

Joan Cabestany
Alberto Prieto
Francisco Sandoval (Eds.)

LNCS 3512

Computational Intelligence and Bioinspired Systems

8th International Work-Conference on
Artificial Neural Networks, IWANN 2005
Vilanova i la Geltrú, Barcelona, Spain, June 2005, Proceedings

 Springer

Commenced Publication in 1973

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison

Lancaster University, UK

Takeo Kanade

Carnegie Mellon University, Pittsburgh, PA, USA

Josef Kittler

University of Surrey, Guildford, UK

Jon M. Kleinberg

Cornell University, Ithaca, NY, USA

Friedemann Mattern

ETH Zurich, Switzerland

John C. Mitchell

Stanford University, CA, USA

Moni Naor

Weizmann Institute of Science, Rehovot, Israel

Oscar Nierstrasz

University of Bern, Switzerland

C. Pandu Rangan

Indian Institute of Technology, Madras, India

Bernhard Steffen

University of Dortmund, Germany

Madhu Sudan

Massachusetts Institute of Technology, MA, USA

Demetri Terzopoulos

New York University, NY, USA

Doug Tygar

University of California, Berkeley, CA, USA

Moshe Y. Vardi

Rice University, Houston, TX, USA

Gerhard Weikum

Max-Planck Institute of Computer Science, Saarbruecken, Germany

Joan Cabestany Alberto Prieto
Francisco Sandoval (Eds.)

Computational Intelligence and Bioinspired Systems

8th International Work-Conference on
Artificial Neural Networks, IWANN 2005
Vilanova i la Geltrú, Barcelona, Spain, June 8-10, 2005
Proceedings

Volume Editors

Joan Cabestany

Universitat Politècnica de Catalunya (UPC), E.T.S.I. Telecomunicación
Departamento de Ingeniería Electrónica
Campus Nord, Edificio C4, C/ Jordi Girona, 1-3, 08034 Barcelona, Spain
E-mail: cabestan@eel.upc.edu

Alberto Prieto

Universidad de Granada, E.T.S.I. Informática
Departamento de Arquitectura de Computadores
C/ Periodista Daniel Saucedo, s/n, 18071 Granada, Spain
E-mail: aprieto@ugr.es

Francisco Sandoval

Universidad de Málaga, E.T.S.I. de Telecomunicación
Departamento de Tecnología Electrónica
Campus Universitario de Teatinos, 29071 Málaga, Spain
E-mail: sandoval@dte.uma.es

Library of Congress Control Number: 2005926938

CR Subject Classification (1998): F.1, F.2, I.2, G.2, I.4, I.5, J.3, J.4, J.1

ISSN 0302-9743

ISBN-10 3-540-26208-3 Springer Berlin Heidelberg New York

ISBN-13 978-3-540-26208-4 Springer Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

Springer is a part of Springer Science+Business Media

springeronline.com

© Springer-Verlag Berlin Heidelberg 2005

Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11494669 06/3142 5 4 3 2 1 0

Preface

We present in this volume the collection of finally accepted papers of the eighth edition of the “IWANN” conference (“International Work-Conference on Artificial Neural Networks”). This biennial meeting focuses on the foundations, theory, models and applications of systems inspired by nature (neural networks, fuzzy logic and evolutionary systems).

Since the first edition of IWANN in Granada (LNCS 540, 1991), the Artificial Neural Network (ANN) community, and the domain itself, have matured and evolved. Under the ANN banner we find a very heterogeneous scenario with a main interest and objective: to better understand nature and beings for the correct elaboration of theories, models and new algorithms. For scientists, engineers and professionals working in the area, this is a very good way to get solid and competitive applications.

We are facing a real revolution with the emergence of embedded intelligence in many artificial systems (systems covering diverse fields: industry, domotics, leisure, healthcare, ...). So we are convinced that an enormous amount of work must be, and should be, still done. Many pieces of the puzzle must be built and placed into their proper positions, offering us new and solid theories and models (necessary tools) for the application and praxis of these current paradigms.

The above-mentioned concepts were the main reason for the subtitle of the IWANN 2005 edition: “*Computational Intelligence and Bioinspired Systems.*” The call for papers was launched several months ago, addressing the following topics:

1. **Mathematical and theoretical methods in computational intelligence.** Complex and social systems; evolutionary and genetic algorithms; fuzzy logic; mathematics for neural networks; RBF structures; self-organizing networks and methods; support vector machines.
2. **Neurocomputational formulations.** Single-neuron modeling; perceptual modeling; system-level neural modeling; spiking neurons; models of biological learning.
3. **Learning and adaptation.** Adaptive systems; imitation learning; reconfigurable systems; supervised, non-supervised, reinforcement and statistical algorithms.
4. **Emulation of cognitive functions.** Decision making; multi-agent systems; multi-sensory integration; natural languages; pattern recognition; perceptual and motor functions (visual, auditory, tactile, etc.); robotics; planning motor control.
5. **Bioinspired systems and neuroengineering.** Embedded neural networks and fuzzy systems; evolvable computing; evolving hardware; microelectronics for neural, fuzzy and bioinspired systems; neural prostheses; retinomorph systems.
6. **Applications.** Biomimetic applications; data analysis and preprocessing; data mining; economics and financial engineering; fuzzy systems for control; the internet; neural networks for control; power systems; signal processing; telecommunication applications; time series and prediction.

After a careful review process of the more than 240 submissions, 150 papers were accepted for publication including the contribution of three invited speakers. In this edition a special emphasis was put on the organization of special sessions. A total of 10 sessions containing 46 papers were accepted for presentation, covering specific aspects like the modelling of neurons, design of neural topologies, applications, etc. This review and selection process was done with the help and cooperation of the members of our International Program Committee.

The organization of this book does not follow the scheme and the order of the main mentioned topics, but is organized in a rational way according to the contents of the accepted papers, going from the more abstract concepts to the concrete and applicable questions and considerations. The result is a 20-chapter volume with the following main parts:

1. Mathematical and Theoretical Methods
2. Evolutionary Computation
3. Neurocomputation-Inspired Models
4. Learning and Adaptation
5. Radial Basis Function Structures
6. Self-organizing Networks and Methods
7. Support Vector Machines
8. Cellular Neural Networks
9. Hybrid Systems
10. Neuroengineering and Hardware Implementations
11. Pattern Recognition
12. Perception and Robotics
13. Applications on Data analysis and Preprocessing
14. Applications on Data Mining
15. Applications on Signal Processing
16. Applications on Image Processing
17. Applications on Forecasting
18. Applications on Independent Component Analysis and Blind Source Separation
19. Applications on Power Systems
20. Other Applications

IWANN 2005 was organized by the Universitat Politècnica de Catalunya, UPC, with the strong cooperation of the Universidad de Granada and the Universidad de Málaga. Sponsorship was obtained from the organizing university, UPC, the Spanish Ministerio de Educación y Ciencia, the AGAUR agency of the Generalitat de Catalunya, and the City Council of Vilanova i la Geltrú.

We would like to express our gratitude to the members of the IWANN Organizing Committee, and to all the people who participated in the event (delegates, invited speakers, special session organizers). The editors would like to address a special mention to the people who helped in the review process as special or additional reviewers.

Finally, we would like to thank Springer, and especially Alfred Hofmann and Anna Kramer, for their continuous support and cooperative work from the very beginning of the IWANN conferences.

June 2005

Joan Cabestany, Universitat Politècnica de Catalunya
Alberto Prieto, Universidad de Granada
Francisco Sandoval, Universidad de Málaga

IWANN 2005 Chairs and Committees

Organizing Committee

Conference Chairs

Joan Cabestany (Univ. Politécnica de Cataluña)

Alberto Prieto (Univ. de Granada)

Francisco Sandoval (Univ. de Málaga)

Technical Program Chairs

Gonzalo Joya (Univ. de Málaga)

Francisco García Lagos (Univ. de Málaga)

Miguel Atencia (Univ. de Málaga)

Publicity and Publication Chairs

Pedro Castillo (Univ. de Granada)

Alberto Guillén (Univ. de Granada)

Francisco Illeras (Univ. de Granada)

Beatriz Prieto (Univ. de Granada)

Registration and Local Arrangements Chairs

Andreu Catalá (Univ. Politécnica de Cataluña)

Cecilio Angulo (Univ. Politécnica de Cataluña)

Xavier Parra (Univ. Politécnica de Cataluña)

Special Sessions Chairs

Christian Jutten (INPG-LIS Grenoble)

Richard Duro (Univ. La Coruña)

Program Committee

Igor Aleksander, Imperial College London, UK

Andreas Andreu, Johns Hopkins University, USA

Antonio Bahamonde, Univ. Oviedo, Gijón, Spain

Sergi Bermejo, Univ. Politécnica de Catalunya, Spain

Gert Cauwenberghs, Johns Hopkins University, USA

Jesus Cid-Sueiro, Univ. Carlos III, Madrid, Spain

Carlos Cotta, Univ. Málaga, Spain

Marie Cottrell, Univ. Paris 1, France

Luiza de Macedo Mourelle, State University of Rio de Janeiro, Brazil

José Dorronsoro, Univ. Autónoma de Madrid, Spain

Richard Duro, Univ. Coruña, Spain

X Organization

Reinhard Eckhorn, Philipps-Universität Marburg, Germany
José Manuel Fernández, Univ. Politécnica de Cartagena, Spain
Kunihiko Fukushima, Osaka Univ., Japan
Dario Floreano, Swiss NSF, EPFL, Switzerland
Patrik Garda, Orsay, France
Karl Goser, Univ. Dortmund, Germany
Manuel Graña, Univ. País Vasco, Spain
Anne Guérin-Dugué, CLIPS, IMAG, Grenoble, France
Alister Hamilton, Univ. Edinburgh, UK
Jeanny Hérault, INP Grenoble, France
Francisco Herrera, Univ. Granada, Spain
Giacomo Indiveri, Institute of Neuroinformatics ETH/UNIZ, Zurich, Switzerland
Pedro Isasi, Univ. Carlos III, Spain
Christian Jutten, INP Grenoble, France
Heinrich Klar, Mikroelektronik, TU Berlin, Germany
Jordi Madrenas, Univ. Politécnica de Catalunya, Spain
Dario Maravall, Univ. P. Madrid, Spain
Bonifacio Martín del Brio, Univ. Zaragoza, Spain
Francesco Masulli, Univ. La Spezia, Genoa, Italy
Juan M. Moreno Aróstegui, Univ. P. de Catalunya, Spain
Alan F. Murray, Edinburgh University, UK
Nadia Nedjah, State University of Rio de Janeiro, Brazil
Julio Ortega, Univ. Granada, Spain
Francisco J. Pelayo, Univ. Granada, Spain
Andrés Perez-Urbe, Univ. of Applied Science of Western Switzerland, Switzerland
Vicenno Piuri, University of Milan, Italy
Angel P. del Pobil, Univ. Jaume I. de Castellón, Spain
Carlos G. Puntonet, Univ. Granada, Spain
Leonardo Reyneri, Politecnico di Torino, Italy
Luigi M. Ricciardi, Univ. di Napoli Federico II, Italy
Ulrich Rückert, Heinz Nixdorf Institute, Univ. of Paderborn, Germany
Ignacio Rojas, Univ. Granada, Spain
Eduardo Ros, Univ. Granada, Spain
Javier Ruiz-del-Solar, Universidad de Chile, Chile
Eduardo Sanchez, LSI, EPFL, Switzerland
Juan V. Sanchez-Andrés, Univ. de La Laguna, Spain
Juan A. Sigüenza, Univ. Autónoma de Madrid, Spain
Jordi Solé-Casals, Univ. de Vic, Spain
Peter Szolgay, Hungarian Academy of Sciences, Hungary
Carme Torras, IRI (CSIC-UPC), Barcelona, Spain
Marley Vellasco, Pontif. Univ. Católica Rio de Janeiro, Brazil
Michel Verleysen, Univ. Cath. de Louvain-la-Neuve, Belgium
Changjiu Zhou, Singapore Polytechnic, Singapore
Barbara Hammer, Univ. of Osnabrück, Germany
Peter Tino, Univ. of Birmingham, UK

Other Reviewers

Ata Kaban
Guillermo Bedoya
Vicenç Parisi
Miguel Sanchez-Marré
Javier Díaz
Antonio Cañas
Héctor Pomares
Manuel Rodriguez
Moisés Salmerón
Eva M. Ortigosa
Catherine Aaron
André Vargas Abs da
Cruz
Miguel Atencia
Chiara Bartolozzi

Jordi Cosp
Javier de Lope
Julián Dorado
José Manuel Ferrández
Francisco Ferrer
Karla Tereza Figueiredo
Leonardo Franco
Raúl Giráldez
Luis González Abril
Luis J. Herrera
José Jerez
Peter Kelly
Elmar W. Lang
Amaury Lendasse

Eric Monte
Dylan Muir
Juan R. Rabuñal
José C. Riquelme
Roberto Ruiz
Vicente Ruiz de Angulo
Francisco Ruiz Vega
Joseph Rynkiewicz
Ricardo Tanscheit
Fabian Theis
José Luis Vázquez
Ahmed Zobaa
Pedro J. Zufiria

Table of Contents

Mathematical and Theoretical Methods

Role of Function Complexity and Network Size in the Generalization Ability of Feedforward Networks <i>Leonardo Franco, José M. Jerez, José M. Bravo</i>	1
Analysis of the Sanger Hebbian Neural Network <i>J. Andrés Berzal, Pedro J. Zufiria</i>	9
Considering Multidimensional Information Through Vector Neural Networks <i>J.L. Crespo, R.J. Duro</i>	17
Combining Ant Colony Optimization with Dynamic Programming for Solving the k -Cardinality Tree Problem <i>Christian Blum, Maria Blesa</i>	25

Evolutionary Computation

A Basic Approach to Reduce the Complexity of a Self-generated Fuzzy Rule-Table for Function Approximation by Use of Symbolic Interpolation <i>G. Rubio, H. Pomares</i>	34
Average Time Complexity of Estimation of Distribution Algorithms <i>C. González, A. Ramírez, J.A. Lozano, P. Larrañaga</i>	42
A Comparison of Evolutionary Approaches to the Shortest Common Supersequence Problem <i>Carlos Cotta</i>	50
Simultaneous Evolution of Neural Network Topologies and Weights for Classification and Regression <i>Miguel Rocha, Paulo Cortez, José Neves</i>	59
Applying Bio-inspired Techniques to the p -Median Problem <i>E. Domínguez, J. Muñoz</i>	67
Optimal Strategy for Resource Allocation of Two-Dimensional Potts Model Using Genetic Algorithm <i>Wing Keung Cheung, Kwok Yip Szeto</i>	75
Memetic Algorithms to Product-Unit Neural Networks for Regression <i>Francisco Martínez-Estudillo, César Hervás-Martínez, Alfonso Martínez-Estudillo, Domingo Ortíz-Boyer</i>	83

Lamarckian Clonal Selection Algorithm Based Function Optimization
Wuhong He, Haifeng Du, Licheng Jiao, Jing Li 91

Neurocomputational Inspired Models

Artificial Neural Networks Based on Brain Circuits Behaviour and Genetic Algorithms
Ana Porto, Alejandro Pazos, Alfonso Araque 99

Modeling Synaptic Transmission and Quantifying Information Transfer in the Granular Layer of the Cerebellum
Egidio D'Angelo, Thierry Nieus, Michele Bezzi, Angelo Arleo, Olivier J.-M.D. Coenen 107

The After-Hyperpolarization Amplitude and the Rise Time Constant of IPSC Affect the Synchronization Properties of Networks of Inhibitory Interneurons
Angelo Di Garbo, Alessandro Panarese, Michele Barbi, Santi Chillemi 115

TiViPE Simulation of a Cortical Crossing Cell Model
Tino Lourens, Emilia Barakova 122

A Model of Spiking-Bursting Neuronal Behavior Using a Piecewise Linear Two-Dimensional Map
Carlos Aguirre, Doris Campos, Pedro Pascual, Eduardo Serrano 130

Real-Time Spiking Neural Network: An Adaptive Cerebellar Model
Christian Boucheny, Richard Carrillo, Eduardo Ros, Olivier J.-M.D. Coenen 136

Modeling Neural Processes in Lindenmayer Systems
Carlos Martín-Vide, Tseren-Onolt Ishdorj 145

Modeling Stimulus Equivalence with Multi Layered Neural Networks
Hiroyuki Okada, Masamichi Sakagami, Hiroshi Yamakawa 153

Instability of Attractors in Auto-associative Networks with Bio-inspired Fast Synaptic Noise
Joaquín J. Torres, Jesús M. Cortés, Joaquín Marro 161

Lookup Table Powered Neural Event-Driven Simulator
Richard R. Carrillo, Eduardo Ros, Eva M. Ortigosa, Boris Barbour, Rodrigo Agís 168

Learning and Adaptation

Joint Kernel Maps
Jason Weston, Bernhard Schölkopf, Olivier Bousquet 176

Statistical Ensemble Method (SEM): A New Meta-machine Learning Approach Based on Statistical Techniques <i>Andrés Yáñez Escolano, Pedro Galindo Riaño, Joaquín Pizarro Junquera, Elisa Guerrero Vázquez</i>	192
Neural Network Modeling by Subsampling <i>Michele La Rocca, Cira Perna</i>	200
Balanced Boosting with Parallel Perceptrons <i>Iván Cantador, José R. Dorronsoro</i>	208
A Reinforcement Learning Algorithm Using Temporal Difference Error in Ant Model <i>SeungGwan Lee, TaeChoong Chung</i>	217
Selection of Weights for Sequential Feed-Forward Neural Networks: An Experimental Study <i>Enrique Romero</i>	225
Exploiting Multitask Learning Schemes Using Private Subnetworks <i>Pedro J. García-Laencina, Aníbal R. Figueiras-Vidal, Jesús Serrano-García, José-Luis Sancho-Gómez</i>	233
Co-evolutionary Learning of Liquid Architectures <i>Igal Raichelgauz, Karina Odinaev, Yehoshua Y. Zeevi</i>	241
Extended Sparse Nonnegative Matrix Factorization <i>Kurt Stadlthanner, Fabian J. Theis, Carlos G. Puntonet, Elmar W. Lang</i> ...	249
Radial Basic Functions Structures	
Using a Mahalanobis-Like Distance to Train Radial Basis Neural Networks <i>J.M. Valls, R. Aler, O. Fernández</i>	257
Robustness of Radial Basis Functions <i>Ralf Eickhoff, Ulrich Rückert</i>	264
Improving Clustering Technique for Functional Approximation Problem Using Fuzzy Logic: ICFA Algorithm <i>A. Guillén, I. Rojas, J. González, H. Pomares, L.J. Herrera, O. Valenzuela, A. Prieto</i>	272
Input Variable Selection in Hierarchical RBF Networks <i>Mohammed Awad, Héctor Pomares, Ignacio Rojas, Luis J. Herrera, Alberto Prieto</i>	280

Approximating I/O Data Using Radial Basis Functions: A New Clustering-Based Approach
Mohammed Awad, Héctor Pomares, Luis Javier Herrera, Jesús González, Alberto Guillén, Fernando Rojas 289

Application of ANOVA to a Cooperative-Coevolutionary Optimization of RBFNs
Antonio J. Rivera, Ignacio Rojas, Julio Ortega 297

Self-organizing Networks and Methods

Characterizing Self-developing Biological Neural Networks: A First Step Towards Their Application to Computing Systems
Hugues Berry, Olivier Temam 306

Cooperative Bees Swarm for Solving the Maximum Weighted Satisfiability Problem
Habiba Drias, Souhila Sadeg, Safa Yahi 318

Deriving Cortical Maps and Elastic Nets from Topology-Preserving Maps
Valery Tereshko 326

Evolution of Cooperating ANNs Through Functional Phenotypic Affinity
F. Bellas, J.A. Becerra, R.J. Duro 333

Robust Growing Hierarchical Self Organizing Map
Sebastián Moreno, Héctor Allende, Cristian Rogel, Rodrigo Salas 341

Support Vector Machines

Web Usage Mining Using Support Vector Machine
Sung-Hae Jun 349

Multi-kernel Growing Support Vector Regressor
D. Gutiérrez-González, E. Parrado-Hernández, A. Navia-Vázquez 357

Cellular Neural Networks

Stability Results for Cellular Neural Networks with Time Delays
Daniela Danciu, Vladimir Răsvan 366

Global Exponential Stability Analysis in Cellular Neural Networks with Time-Varying Coefficients and Delays
Qiang Zhang, Dongsheng Zhou, Xiaopeng Wei, Jin Xu 374

Hybrid Systems

Diversity and Multimodal Search with a Hybrid Two-Population GA: An Application to ANN Development <i>Juan R. Rabuñal, Julián Dorado, Marcos Gestal, Nieves Pedreira</i>	382
Identification of Fuzzy Systems with the Aid of Genetic Fuzzy Granulation <i>Sung-Kwun Oh, Keon-Jun Park, Yong-Soo Kim, Tae-Chon Ahn</i>	391
Clustering-Based TSK Neuro-fuzzy Model for Function Approximation with Interpretable Sub-models <i>Luis Javier Herrera, Héctor Pomares, Ignacio Rojas, Alberto Guillén, Jesús González, Mohammed Awad</i>	399
Genetically Optimized Hybrid Fuzzy Neural Networks with the Aid of TSK Fuzzy Inference Rules and Polynomial Neural Networks <i>Sung-Kwun Oh, Witold Pedrycz, Hyun-Ki Kim, Yong-Kab Kim</i>	407
IG-Based Genetically Optimized Fuzzy Polynomial Neural Networks <i>Sung-Kwun Oh, Seok-Beom Roh, Witold Pedrycz, Jong-Beom Lee</i>	416
Hierarchical Neuro-fuzzy Models Based on Reinforcement Learning for Intelligent Agents <i>Karla Figueiredo, Marley Velasco, Marco Aurélio Pacheco</i>	424

Neuroengineering and Hardware Implementations

Interfacing with Patterned in Vitro Neural Networks by Means of Hybrid Glass-Elastomer Neurovectors: Progress on Neuron Placement, Neurite Outgrowth and Biopotential Measurements <i>Enric Claverol-Tinturé, Xavier Rosell, Joan Cabestany</i>	433
Using Kolmogorov Inspired Gates for Low Power Nanoelectronics <i>Valeriu Beiu, Artur Zawadski, Răzvan Andonie, Snorre Aunet</i>	438
CMOL CrossNets as Pattern Classifiers <i>Jung Hoon Lee, Konstantin K. Likharev</i>	446
Analog VLSI Implementation of Adaptive Synapses in Pulsed Neural Networks <i>Tim Kaulmann, Markus Ferber, Ulf Witkowski, Ulrich Rückert</i>	455
Smart Sensing with Adaptive Analog Circuits <i>Guillermo Zatorre, Nicolás Medrano, Santiago Celma, Bonifacio Martín-del-Brío, Antonio Bono</i>	463

Spiking Neurons Computing Platform <i>Eduardo Ros, Eva M. Ortigosa, Rodrigo Agís, Richard Carrillo, Alberto Prieto, Mike Arnold</i>	471
Inter-spike-intervals Analysis of Poisson Like Hardware Synthetic AER Generation <i>A. Linares-Barranco, M. Oster, D. Cascado, G. Jiménez, A. Civit, B. Linares-Barranco</i>	479
Ultra Low-Power Neural Inspired Addition: When Serial Might Outperform Parallel Architectures <i>Valeriu Beiu, Asbjørn Djupdal, Snorre Aunet</i>	486
An Asynchronous 4-to-4 AER Mapper <i>H. Kolle Riis, Ph. Häfliger</i>	494
Fast Optoelectronic Neural Network for Vision Applications <i>Marta Ruiz-Llata, Horacip Lamela</i>	502
A Computational Tool to Test Neuromorphic Encoding Schemes for Visual Neuroprostheses <i>Christian A. Morillas, Samuel F. Romero, Antonio Martinez, Francisco J. Pelayo, Eduardo Fernández</i>	510
Test Infrastructure for Address-Event-Representation Communications <i>R. Paz, F. Gomez-Rodriguez, M.A. Rodriguez, A. Linares-Barranco, G. Jimenez, A. Civit</i>	518
Automatic Generation of Bio-inspired Retina-Like Processing Hardware <i>Antonio Martínez, Leonardo M. Reyneri, Francisco J. Pelayo, Samuel F. Romero, Christian A. Morillas, Begoña Pino</i>	527
Two Hardware Implementations of Exhaustive Synthetic AER Generation Method <i>F. Gomez-Rodriguez, R. Paz, L. Miro, A. Linares-Barranco, G. Jimenez, A. Civit</i>	534
On the Design of a Parallel Genetic Algorithm Based on a Modified Survival Method for Evolvable Hardware <i>Dong-Sun Kim, Hyun-Sik Kim, Youn-Sung Lee, Duck-Jin Chung</i>	541
A Novel Approach for the Implementation of Large Scale Spiking Neural Networks on FPGA Hardware <i>B. Glackin, T.M. McGinnity, L.P. Maguire, Q.X. Wu, A. Belatreche</i>	552
A Quaternary CLB Design Using Quantum Device Technology on Silicon for FPGA Neural Network Architectures <i>P.M. Kelly, T.M. McGinnity, L.P. Maguire, L.M. McDaid</i>	564

A Dynamically-Reconfigurable FPGA Platform for Evolving Fuzzy Systems <i>Grégory Mermoud, Andres Upegui, Carlos-Andres Peña, Eduardo Sanchez</i>	572
FPGA Implementation of Hopfield Networks for Systems Identification <i>Hafida Boumeridja, Miguel Atencia, Gonzalo Joya, Francisco Sandoval ...</i>	582
An FPGA-Based Adaptive Fuzzy Coprocessor <i>Antonio Di Stefano, Costantino Giaconia</i>	590
Pattern Recognition	
Cascade Ensembles <i>N. García-Pedrajas, D. Ortiz-Boyer, R. del Castillo-Gomariz, C. Hervás-Martínez</i>	598
Ensembles of Multilayer Feedforward: Some New Results <i>Joaquín Torres-Sospedra, Carlos Hernández-Espinosa, Mercedes Fernández-Redondo</i>	604
Layered Network Computations by Parallel Nonlinear Processing <i>Naohiro Ishii, Toshinori Deguchi, Hiroshi Sasaki</i>	612
Fast Classification with Neural Networks via Confidence Rating <i>J. Arenas-García, V. Gómez-Verdejo, S. Muñoz-Romero, M. Ortega-Moral, A.R. Figueiras-Vidal</i>	622
Characterization and Synthesis of Objects Using Growing Neural Gas <i>José García, Francisco Flórez, Juan Manuel García, Antonio Hernández</i>	630
ARGEN + AREPO: Improving the Search Process with Artificial Genetic Engineering <i>Agustín León-Barranco, Carlos A. Reyes-García</i>	637
Perception and Robotics	
Modelling Perceptual Discrimination <i>Janet Aisbett, James T. Townsend, Greg Gibbon</i>	646
Memory Retrieval in a Neural Network with Chaotic Neurons and Dynamic Synapses <i>Zhijie Wang, Hong Fan</i>	654
Neural Network Based 3D Model Reconstruction with Highly Distorted Stereoscopic Sensors <i>Wan-liang Wang, Bing-bing Xia, Qiu Guan, Shengyong Chen</i>	661

Evolutionary Design of a Brain-Computer Interface <i>G. Romero, M.G. Arenas, P.A. Castillo, J.J. Merelo</i>	669
Vision-Based Walking Parameter Estimation for Biped Locomotion Imitation <i>Juan Pedro Bandera Rubio, Changjiu Zhou, Francisco Sandoval Hernández</i>	677
Designing a Control System for an Autonomous Robot Using an Evolutionary Algorithm <i>P.A. Castillo, G. Romero, M.G. Arenas, J.J. Merelo, A. Prieto</i>	685
Pruning Neural Networks for a Two-Link Robot Control System <i>Jie Ni, Qing Song</i>	693
Using PSOMs to Learn Inverse Kinematics Through Virtual Decomposition of the Robot <i>Vicente Ruiz de Angulo, Carme Torras</i>	701
Highly Modular Architecture for the General Control of Autonomous Robots <i>Ricardo A. Téllez, Cecilio Angulo, Diego E. Pardo</i>	709
Complex Behaviours Through Modulation in Autonomous Robot Control <i>J.A. Becerra, F. Bellas, J. Santos, R.J. Duro</i>	717
Applications on Data Analysis and Preprocessing	
Explorative Data Analysis Based on Self-organizing Maps and Automatic Map Analysis <i>Marc Franzmeier, Ulf Witkowski, Ulrich Rückert</i>	725
A Novel Optimization of Profile HMM by a Hybrid Genetic Algorithm <i>Lifang Liu, Hongwei Huo, Baoshu Wang</i>	734
Heuristic Search over a Ranking for Feature Selection <i>Roberto Ruiz, José C. Riquelme, Jesús S. Aguilar-Ruiz</i>	742
Intrinsic Dimensionality Maps with the PCASOM <i>Ezequiel López-Rubio, Juan Miguel Ortiz-de-Lazcano-Lobato, María del Carmen Vargas-González, José Miguel López-Rubio</i>	750
Applications on Data Mining	
The Curse of Dimensionality in Data Mining and Time Series Prediction <i>Michel Verleysen, Damien François</i>	758

Obtaining a Complex Linguistic Data Summaries from Database Based on a New Linguistic Aggregation Operator <i>Zheng Pei, Yajun Du, Liangzhong Yi, Yang Xu</i>	771
Bias and Variance of Rotation-Based Ensembles <i>Juan José Rodríguez, Carlos J. Alonso, Oscar J. Prieto</i>	779
Comparative Assessment of the Robustness of Missing Data Imputation Through Generative Topographic Mapping <i>Iván Olier, Alfredo Vellido</i>	787
Induction of Decision Trees Using an Internal Control of Induction <i>Gonzalo Ramos-Jiménez, José del Campo-Ávila, Rafael Morales-Bueno</i>	795
An Approach to Reduce the Cost of Evaluation in Evolutionary Learning <i>Raúl Giráldez, Norberto Díaz-Díaz, Isabel Nepomuceno, Jesús S. Aguilar-Ruiz</i>	804
Applications on Signal Processing	
Efficient Design of Fixed Point Digital FIR Filters by Using Differential Evolution Algorithm <i>Nurhan Karaboğa, Bahadır Çetinkaya</i>	812
Manifold Constrained Finite Gaussian Mixtures <i>Cédric Archambeau, Michel Verleysen</i>	820
A Comparison of Gaussian Based ANNs for the Classification of Multidimensional Hyperspectral Signals <i>A. Prieto, F. Bellas, R.J. Duro, F. Lopez-Peña</i>	829
Voice Activity Detection Using Higher Order Statistics <i>J.M. Górriz, J. Ramírez, J.C. Segura, S. Hornillo</i>	837
Matched Filter as Pre-processing Tool for Multi-user Detection in DS-CDMA System <i>O. Chakkor, C.G. Puntonet, B. Pino, J.M. Gorriz</i>	845
A Robust Multiple Feature Approach to Endpoint Detection in Car Environment Based on Advanced Classifiers <i>C. Comas, E. Monte-Moreno, J. Solé-Casals</i>	850
Canonical Correlation Analysis Using for DOA Estimation of Multiple Audio Sources <i>Gaoming Huang, Luxi Yang, Zhenya He</i>	857

Applications on Image Processing

Multilayer Perceptrons Applied to Traffic Sign Recognition Tasks <i>R. Vicen-Bueno, R. Gil-Pita, M. Rosa-Zurera, M. Utrilla-Manso, F. López-Ferreras</i>	865
Shape Classification Algorithm Using Support Vector Machines for Traffic Sign Recognition <i>P. Gil-Jiménez, S. Lafuente-Arroyo, S. Maldonado-Bascón, H. Gómez-Moreno</i>	873
A Practical License Plate Recognition System for Real-Time Environments <i>Cemil Oz, Fikret Ercal</i>	881
Matching Algorithm for Hangul Recognition Based on PDA <i>Hyeong-Gyun Kim, Yong-Ho Kim, Jong-Geun Jeong</i>	889
Block LDA for Face Recognition <i>Vo Dinh Minh Nhat, Sungyoung Lee</i>	899
Image Processing with CNN in a FPGA-Based Augmented Reality System for Visually Impaired People <i>F. Javier Toledo, J. Javier Martínez, F. Javier Garrigós, J. Manuel Ferrández</i>	906
A Gradient Descent MRI Illumination Correction Algorithm <i>M. Garcia, E. Fernandez, M. Graña, F.J. Torrealdea</i>	913
Multifractal Analysis of Electroencephalogram Time Series in Humans <i>In-Ho Song, Sang-Min Lee, In-Young Kim, Doo-Soo Lee, Sun I. Kim</i>	921
Face Recognition with Improved Pairwise Coupling Support Vector Machines <i>Huaqing Li, Feihu Qi, Shaoyu Wang</i>	927
Face Recognition System Based on PCA and Feedforward Neural Networks <i>Alaa Eleyan, Hasan Demirel</i>	935
A New Fuzzy Approach for Edge Detection <i>Yasar Becerikli, Tayfun M. Karan</i>	943

Applications on Forecasting

Fault Detection and Prediction of Clocks and Timers Based on Computer Audition and Probabilistic Neural Networks <i>S.Y. Chen, C.Y. Yao, G. Xiao, Y.S. Ying, W.L. Wang</i>	952
Long Term Prediction of Product Quality in a Glass Manufacturing Process Using a Kernel Based Approach <i>Tobias Jung, Luis Herrera, Bernhard Schoelkopf</i>	960

Time Series Forecast with Anticipation Using Genetic Programming <i>Daniel Rivero, Juan R. Rabuñal, Julián Dorado, Alejandro Pazos</i>	968
Multi-modeling: A Different Way to Design Intelligent Predictors <i>Kurosh Madani, Lamine Thiaw, Rachid Malti, Gustave Sow</i>	976
Input and Structure Selection for k -NN Approximator <i>Antti Sorjamaa, Nima Reyhani, Amaury Lendasse</i>	985
Nonlinear Robust Identification with ϵ – GA: FPS Under Several Norms Simultaneously <i>J.M. Herrero, X. Blasco, M. Martínez, C. Ramos</i>	993
Input Selection for Long-Term Prediction of Time Series <i>Jarkko Tikka, Jaakko Hollmén, Amaury Lendasse</i>	1002
Direct and Recursive Prediction of Time Series Using Mutual Information Selection <i>Yongnan Ji, Jin Hao, Nima Reyhani, Amaury Lendasse</i>	1010
Load Forecasting Using Fixed-Size Least Squares Support Vector Machines <i>Marcelo Espinoza, Johan A.K. Suykens, Bart De Moor</i>	1018
TaSe Model for Long Term Time Series Forecasting <i>Luis Javier Herrera, Héctor Pomares, Ignacio Rojas, Alberto Guillén, Olga Valenzuela, Alberto Prieto</i>	1027
Applications on Independent Component Analysis and Blind Source Separation	
Connections Between ICA and Sparse Coding Revisited <i>Susana Hornillo-Mellado, Rubén Martín-Clemente, Juan M. Górriz-Sáez</i>	1035
Analysis of Variance of Three Contrast Functions in a Genetic Algorithm for Non-linear Blind Source Separation <i>F. Rojas, J.M. Górriz, O. Valenzuela</i>	1043
Robust Processing of Microarray Data by Independent Component Analysis <i>Francisco Díaz, Raul Malutan, Pedro Gómez, Victoria Rodellar, Carlos G. Puntonet</i>	1051
Multichannel Blind Signal Separation in Semiconductor-Based GAS Sensor Arrays <i>Guillermo Bedoya, Sergi Bermejo, Joan Cabestany</i>	1059

Clustering of Signals Using Incomplete Independent Component Analysis <i>Ingo R. Keck, Elmar W. Lang, Salua Nassabay, Carlos G. Puntonet</i>	1067
A Hybridization of Simulated Annealing and Local PCA for Automatic Component Assignment Within ICA <i>M. Böhm, K. Stadthanner, E.W. Lang, A.M. Tomé, A.R. Teixeira, F.J. Theis, C.G. Puntonet</i>	1075
An ICA Approach to Detect Functionally Different Intra-regional Neuronal Signals in MEG Data <i>Giulia Barbati, Camillo Porcaro, Filippo Zappasodi, Franca Tecchio</i>	1083
Filtering-Free Blind Separation of Correlated Images <i>Frédéric Vrins, John A. Lee, Michel Verleysen</i>	1091
Robust Blind Image Watermarking with Independent Component Analysis: A Embedding Algorithm <i>Juan José Murillo-Fuentes, Rafael Boloix-Tortosa</i>	1100
Applications on Power Systems	
Adaptive Load Frequency Control with Dynamic Fuzzy Networks in Power Systems <i>Yusuf Oysal, Ahmet Serdar Yilmaz, Etem Koklukaya</i>	1108
Fault Fuzzy Rule Extraction from AC Motors by Neuro-fuzzy Models <i>G.I. Sainz, R. García, M.J. Fuente</i>	1116
Adaptive Power System Stabilizer Using ANFIS and Genetic Algorithms <i>Jesús Fraile-Ardanuy, Pedro J. Zufiria</i>	1124
Use of ANN in a Research Reactor Power Fuzzy Controller <i>Jorge S. Benítez-Read, Da Ruan, Jorge A. Ruiz-Enciso, Régulo López-Callejas, Joel O. Pacheco-Sotelo</i>	1132
Other Applications	
The Use of Bayesian Networks for Subgrouping Heterogeneous Diseases <i>Abdelaziz Ouali, Amar Ramdane Cherif, Marie-Odile Krebs</i>	1140
Graph Partitioning via Recurrent Multivalued Neural Networks <i>Enrique Mérida-Casermeyro, Domingo López-Rodríguez</i>	1149
Dynamical Random Neural Network Approach to a Problem of Optimal Resource Allocation <i>YongJun Zhong, DonChuan Sun, JianJun Wu</i>	1157

Biometric Hand Recognition Using Neural Networks <i>Francisco Martínez, Carlos Orrite, Elías Herrero</i>	1164
Biometric Identification by Means of Hand Geometry and a Neural Net Classifier <i>Marcos Faundez-Zanuy, Guillermo Mar Navarro Mérida</i>	1172
Study of a Committee of Neural Networks for Biometric Hand-Geometry Recognition <i>Marcos Faundez-Zanuy</i>	1180
An Evolutionary Environment for Wind Turbine Blade Design <i>V. Díaz Casás, F. Lopez Peña, A. Lamas, R.J. Duro</i>	1188
Linguistic Properties Based on American Sign Language Recognition with Artificial Neural Networks Using a Sensory Glove and Motion Tracker <i>Cemil Oz, Ming C. Leu</i>	1197
Crack Detection in Wooden Pallets Using the Wavelet Transform of the Histogram of Connected Elements <i>M.A. Patricio, D. Maravall, L. Usero, J. Rejón</i>	1206
A Competitive-Based Method for Determining the Number of Groups: A Clinical Application <i>Antonio Sánchez, Francisco Vico, Santiago Cabello, Francisco Veredas, Isaac López, Yamina Seamari, Javier Farfán, Guillermo García-Herrera</i>	1214
Knowledge Extraction from Neural Networks Using the All-Permutations Fuzzy Rule Base: The LED Display Recognition Problem <i>Eyal Kolman, Michael Margalio</i>	1222
Controlling Force Based on Radial Fuzzy Functions in High-Speed Machining Processes <i>Rodolfo Haber-Guerra, Rodolfo Haber-Haber, José R. Alique</i>	1230
Sequential PN Acquisition Scheme Based on a Fuzzy Logic Controller <i>Rosa María Alsina, Jose Antonio Morán, Joan Claudi Socoró</i>	1238
Fuzzy Logic System for Students' Evaluation <i>José Antonio Montero, Rosa María Alsina, Jose Antonio Morán, Mariona Cid</i>	1246
Author Index	1255

Role of Function Complexity and Network Size in the Generalization Ability of Feedforward Networks

Leonardo Franco^{1,2}, José M. Jerez², and José M. Bravo²

¹ Dept. of Experimental Psychology, University of Oxford,
South Parks Road, Oxford OX1 3UD, UK

Leonardo.Franco@psy.ox.ac.uk

² Departamento de Lenguajes y Ciencias de la Computación,
University of Málaga, 29071 Málaga, Spain

Abstract. The generalization ability of different sizes architectures with one and two hidden layers trained with backpropagation combined with early stopping have been analyzed. The dependence of the generalization process on the complexity of the function being implemented is studied using a recently introduced measure for the complexity of Boolean functions. For a whole set of Boolean symmetric functions it is found that large neural networks have a better generalization ability on a large complexity range of the functions in comparison to smaller ones and also that the introduction of a small second hidden layer of neurons further improves the generalization ability for very complex functions. Quasi-random generated Boolean functions were also analyzed and we found that in this case the generalization ability shows small variability across different network sizes both with one and two hidden layer network architectures.

1 Introduction

Neural networks are nowadays widely use in different applications in pattern recognition and classification tasks due to their ability to learn from examples and to generalize. There exists some general theoretical results about the size of the network needed to implement a desired function [1, 2, 3] but at the time of the implementation the theory is not always accurate. In practice, the problem of selecting a network architecture for a determined application is mainly based on the “trial-and-error” method as no theoretical formula gives clear insight into this problem. Answers to general questions as: Are two hidden layer networks better than one ? or Does larger networks generalize better than smaller ones? are still controversial. A simple and general idea about what network size utilize comes from Occam’s razor: the simpler the solution the better, but it has been shown that very large networks perform sometimes better than smaller ones [3, 4]. The reasons for the previous findings are still unclear and it seems that they arise from properties of the backpropagation algorithm combined with validation

procedures that avoid overfitting [1, 4, 5, 6, 7]. Another results [8] state that what matters to obtain valid generalization is the value of the weights and not the size of the network but this does not solve completely the size dilemma. Moreover, most of the practical results based on simulations concentrate on the use of very few functions, even the most complete studies use less than 30 different functions [6, 7], and also in general the complexity of the analyzed functions is ignored [3, 9].

The complexity of Boolean functions has been studied for a long time, the aim of that being to have a criterion for deciding if a problem is easier to solve or implement than another [10]. Within the area of circuit complexity the complexity of a Boolean function has been defined as the minimum size of a circuit that could implement it, while the size of the circuit is measured as the number of nodes that composed the circuit. Many results have been derived for certain classes of functions, as for symmetric and arithmetic ones, where in general, bounds on the size of the circuits to compute the functions are obtained [10, 11], but all these measures are quite complicate to compute in most cases. In [3], functions were generated with different complexity by selecting a parameter K that controlled the size of the maximum weight in the network. They showed that functions with higher K were more complex, and expected the generalization to be worse as K increases. In this paper, we use a recently introduced measure for the complexity of Boolean functions [9, 12] to analyze how the network architecture affects the generalization ability obtained for different classes of functions grouped according to their complexity.

2 A Measure for the Complexity of Boolean Functions

The measure proposed in [9] is based on the results obtained in [13, 14], where a close relationship between the number of examples needed to obtain valid generalization and the number of neighbouring examples with different outputs was found. The complexity measure $\mathcal{C}[f]$ is obtained from the number of pairs of neighbouring examples having different outputs. The complexity measure used in this paper is defined as:

$$\mathcal{C}[f] = \mathcal{C}_1[f] + \mathcal{C}_2[f], \quad (1)$$

where $\mathcal{C}_i[f]$, $i = 1, 2$ are the terms taking into account pairs of examples at a Hamming distance one and two. The first term can be written as:

$$\mathcal{C}_1[f] = \frac{1}{N_{ex} * N_{neigh}} \sum_{j=1}^{N_{ex}} \left(\sum_{\{l | Hamming(e_j, e_l) = 1\}} (|f(e_j) - f(e_l)|) \right), \quad (2)$$

where the first factor, $\frac{1}{N_{ex} * N_{neigh}}$, is a normalization one, counting for the total number of pairs considered, N_{ex} is the total number of examples equals to 2^N , and N_{neigh} stands for the number of neighbour examples at a Hamming distance

of 1. The second term $C_2[f]$ is constructed in an analogous way. The complexity of the Boolean functions using the measure of Eq. 1 ranges from 0 to 1.5 [9].

3 Simulations Results

To analyze the generalization ability of different architectures and to study how this property change with network size we carried intensive numerical simulations for symmetric and quasi-random generated Boolean functions. All the networks were trained with Backpropagation with momentum. An early stopping procedure was implemented with the aim of avoid overtraining and improve generalization. The validation error was monitored during training and at the end of the maximum number of epochs permitted, the generalization error was taken at the point where the validation error was minimum. A important use of the complexity measure is that permit to analyzed the complexity of the output functions and compared it to the complexity of the target functions, to analyze what functions the networks generalize to. The number of examples available for training was 256 as in all cases we consider input functions of size $N=8$. We divided the examples in a training set containing 128 examples, and into validation and generalization test sets containing 64 examples each one. The learning rate constant was fixed during training and equal to 0.05 and the momentum term was set to 0.3. The maximum number of epochs allowed for training was selected after some initial tests and was selected to be approximated 4 to 5 times larger than the optimal training time (typical values were 600 to 1000). Two sets of functions symmetric and quasi-random functions generated starting from the parity function were considered and the results are described in the subsections below. To analyze the dependence of the generalization ability with the complexity of the functions the functions were grouped in ten different levels of complexity from 0 to 1.0.

3.1 Symmetric Functions

An important class of Boolean functions are the symmetric ones, those with values independent of the order of the input, i.e., the output of the examples depends only on the total number of input bits ON (the number of 1's). The class of symmetric functions contains many fundamental functions like sorting and all types of counting ones, including also the well known parity function [9, 10, 11]. They have been extensively studied and many lower bounds have been obtained for circuits computing them. A general result states that a circuit of size $\mathcal{O}(\sqrt{N})$ gates and depth 3 with polynomial weights is enough to compute all the symmetric functions [11].

We performed simulations using all 512 existing symmetric functions for the case $N = 8$, using neural networks with one and two hidden layers. The number of neurons for the analyzed cases in which the networks have only one hidden layer was $NH1=5, 10, 25, 50, 250, 500$. The best result, in terms of the generalization ability obtained, was by using the architecture with $NH1=250$. In Fig. 1 the generalization ability as a function of the function complexity is shown for

the different networks utilized. The results were computed from five averages taken for every symmetric function. Standard deviations were also computed for every complexity group and it showed that the differences obtained for different architectures (for example between the network with $NH1=250$ $NH2=0$ and the architecture with $NH1=250$ $NH2=10$) were statistically significant for groups of functions with the same complexity. We also computed the average absolute value of the weights in the different layers and the result was that the size of the weights was proportional to the training time, that was also found to be proportional to the function complexity (In [3, 8] similar findings were obtained).

The complexity of the functions to which the networks generalize to was also investigated (i.e. the functions that the trained networks compute at the point of minimum validation error). We found that in almost all cases the output functions were more complex than the target ones.

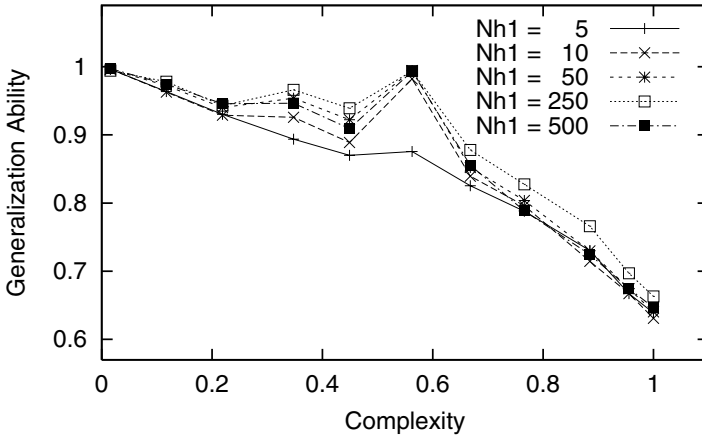


Fig. 1. Generalization ability vs function complexity for all Symmetric Boolean functions with $N=8$ inputs implemented on single hidden layer networks with a number of hidden neurons $NH1= 5,10,25,100,250$ and 500

We also analyzed the generalization ability for cases of architectures having two hidden layers of neurons, first by restricting our study to networks with $NH1=250$, that was the optimal value found for one hidden layer networks. The number of second hidden neurons analyzed were $NH2=5, 10, 25, 50, 250, 500$. The best results were obtained with an architecture 8-250-10-1, that outperformed the architecture 8-250-1. In Fig. 2 the results are shown only for some two-layer architectures (for clarity) in comparison to the best one hidden layer architecture. A further enlargement of the second layer of neurons did not improve the generalization ability. Other cases with a different number of neurons in the first and second layer were considered but the results were worse than the cases already mentioned.

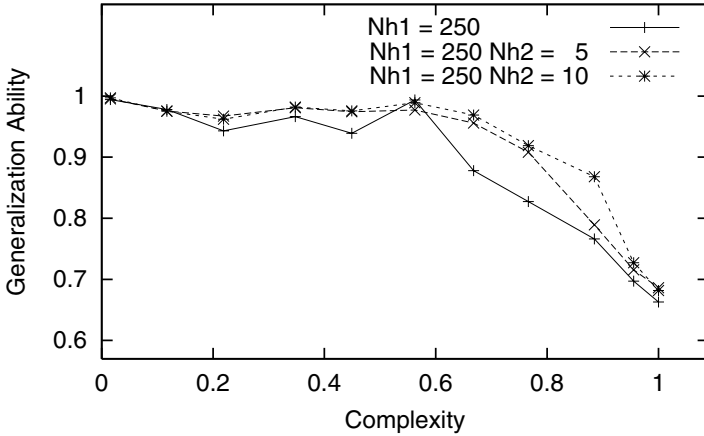


Fig. 2. Generalization ability vs function complexity for all Symmetric Boolean functions with $N=8$ inputs for the cases of two two hidden layer architectures with 250 in the first hidden layer and $NH2=5,10$ in the second hidden layer. For comparison the results for the case of having a single layer with $NH1=250$ is also shown

Table 1. Average generalization ability, training time (epochs) and final training error obtained for different size architectures with one or two hidden layers of neurons constructed to compute all Boolean symmetric functions with $N=8$ inputs

Neurons in 1^{st} hidden layer	Neurons in 2^{nd} hidden layer	Generalization ability	Train. time (epochs)	Final train. error
5	–	0.761	256	0.111
50	–	0.781	167	0.019
250	–	0.803	132	0.060
500	–	0.780	291	0.187
250	5	0.839	192	0.150
250	10	0.852	186	0.136
250	25	0.846	159	0.142

It is possible to observe from Fig. 1 that for networks with a single layer of neurons the use of a large number of neurons up to 250 increases the generalization ability for the whole ranges of complexities. When a second layer of weights is included, the main effect was that the network improved the generalization ability on functions with a complexity larger than 0.6 (See Fig. 2). In table 1, the average generalization ability obtained for some of the architectures used is shown together with the training time (in epochs) until the point in which the minimum validation error was found and also is shown the training error at the end of the training procedure to give an idea of the computational capacity of the network. In a previous study [9], in which the architecture sizes were much restricted, up to 30 neurons in a single hidden layer, we found only a slightly im-

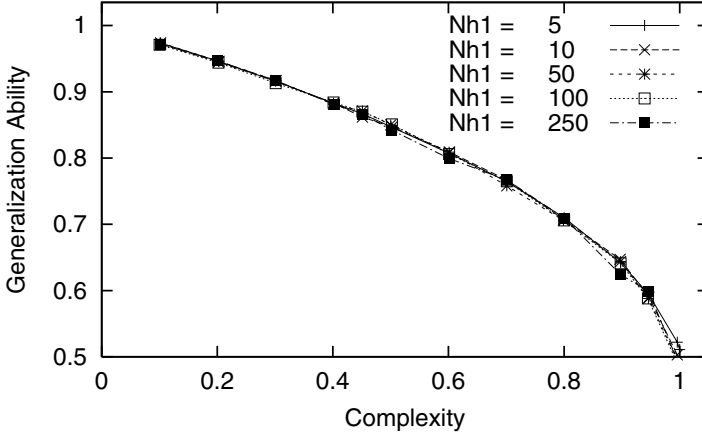


Fig. 3. Random functions implemented on a one hidden layer architecture

Table 2. Average generalization ability, training time (epochs) and final training error obtained for different size architectures with one or two hidden layers of neurons constructed to compute quasi-random Boolean functions generated by modifications of the parity function for $N=8$ inputs

Neurons in 1^{st} hidden layer	Neurons in 2^{nd} hidden layer	Generalization ability	Train. time (epochs)	Final train. error
5	–	0.789	79	0.063
50	–	0.787	58	0.004
250	–	0.785	132	0.060
5	5	0.798	19	0.165
5	10	0.795	44	0.136
5	50	0.792	83	0.129

provement in the generalization ability for the case of very low complex functions and thought that the effect could be explained on the basis that larger networks have more configurations implementing trivial functions. Whether the previous assertion is true is worth but difficult to elucidate and in terms of the generalization ability that can be obtained using Backpropagation combined with early stopping it does not seem to be the case at least for symmetric functions.

3.2 Random Functions

There exists 2^{2^N} Boolean functions of N inputs, making their general study very complicated except for very simple and small cases. Totally random functions are very complex with an average complexity around 1.0 [9].

To analyze a large set of different functions we generate functions with different complexity by modifying with a certain probability the outputs equal to 1 of the parity function, to obtain functions that are asymmetric in the number

of outputs and with a complexity in the range 1.0 to 0. One hundred functions were analyzed for each of 10 levels of complexities in which the functions were grouped, with average complexity from 0.1 to 1.0 in steps of 0.1. We run simulations using networks with a single hidden layer and a number of hidden neurons $NH1$ equal to $NH1=5,10,25,50,100,250,500$ and obtained that the generalization ability for the different complexity groups of functions did not change much with the size of the hidden layer, as can be appreciated from Fig. 3. On average the best one hidden layer architecture was the one with only $NH1=5$ neurons. It was also obtained as in the previous section that the introduction of a second hidden layer of neurons improve the overall generalization ability and the optimal values found were $NH1=5$ and $NH2=5$. In table 2 average results are shown for some of the architectures used. Note from the results in table 2 that the final training error was higher for the optimal architectures, and this might indicate that larger architectures has a major propensity to overfitting for these quasi-random functions while two hidden layer networks seems to suffer less of this problem than the one hidden layer ones.

4 Discussion

We have analyzed the generalization ability of different network architectures studying how the generalization ability is affected by the complexity of the functions being implemented and by the size of the architecture. It is generally assumed that the generalization ability depends on the complexity of the target function but we do not know of previous studies addressing this point except from [3], and we think that this is mainly due to the lack of a simple measure for the complexity of the functions. The use of a recently introduced complexity measure permit us to have a clearer picture about how different architectures perform by permitting the functions under analysis to be grouped according to their complexity [9].

We obtained that large neural networks do not overfit much if trained with the Backpropagation algorithm combined with an early stopping procedure as it was observed in previous studies [3, 4]. For the case of network architectures with a single layer of hidden neurons, the optimum value for the number of neurons was obtained with $NH1=250$ when the set of symmetric Boolean functions was considered. It was further shown that generalization ability improved if a second layer of hidden neurons was used where the optimal number of neurons found was $NH2=10$. In particular, it was observed that the introduction of the second layer of neurons improves the generalization ability of very complex symmetric functions with a complexity between 0.6 and 0.9.

For the case where quasi-random functions were analyzed it was found that the optimal network with a single hidden layer was very small $NH1=5$, but when using larger networks of up to $NH1=500$ the average generalization values were quite similar (See table 2). It was also observed in this case that the introduction of a second layer of hidden neurons further improved the average generalization

ability, but no significant changes were observed in terms of the functions with different complexity.

We are currently extending the measure of complexity to continuous input functions to be able to carry out similar studies with real problems.

References

1. Haykin, S. (1994). *Neural Networks: A Comprehensive Foundation*. Macmillan/IEEE Press.
2. Baum, E.B. & Haussler, D. (1989) What size net gives valid generalization ? *Neural Computation*, *1*, pp. 151-160.
3. Lawrence, S., Giles, C. L., & Tsoi, A. C. (1996). What Size Neural Network Gives Optimal Generalization ? Convergence Properties of Backpropagation. In Technical Report UMIACS-TR-96-22 and CS-TR-3617, Institute for Advanced Computer Studies, Univ. of Maryland.
4. Caruana, R., Lawrence, S., & Giles, C.L. (2001). Overfitting in Neural Networks: Backpropagation, Conjugate Gradient, and Early Stopping. In Leen, T. K., Dietterich, T. G. & Tresp, V. editors, *Advances in Neural Information Processing Systems*, MIT Press, *13*, pp. 402-408.
5. Krogh, A. & Hertz, J.A. (1992) A simple weight decay can improve generalization. In J.E. Moody, S. J. Hanson, & R. P. Lippmann editors, *Advances in Neural Information Processing Systems* Morgan Kaufmann, San Mateo, CA, *4*, pp. 950-957.
6. Prechelt, L. (1998). Automatic Early Stopping Using Cross Validation: Quantifying the Criteria. *Neural Networks*, *11*, pp.761-767.
7. Setiono, R. (2001) Feedforward neural network construction using cross-validation, *Neural Computation*, *13*, pp. 2865-2877.
8. Bartlett, P.L. (1997). For valid generalization the size of the weights is more important than the size of the network. In M.C. Mozer, M. I. Jordan, & T. Petsche, editors, *Advances in Neural Information Processing Systems*, MIT Press, *9*, pp. 134-140 .
9. Franco, L. & Anthony, M. (2004). On a generalization complexity measure for Boolean functions. In *Proceedings of the 2004 IEEE International Joint Conference on Neural Networks*, IEEE Press, pp. 973-978.
10. Wegener, I. (1987) *The complexity of Boolean functions*. Wiley and Sons Inc.
11. Siu, K.Y., Roychowdhury, V.P., & Kailath, T. (1991) Depth-Size Tradeoffs for Neural Computation *IEEE Transactions on Computers*, *40*, pp. 1402-1412.
12. Franco, L. & Cannas, S.A. (2004). Non glassy ground-state in a long-range anti-ferromagnetic frustrated model in the hypercubic cell *Physica A*, *332*, pp. 337-348.
13. Franco, L. & Cannas, S.A. (2000). Generalization and Selection of Examples in Feedforward Neural Networks. *Neural Computation*, *12*, 10, pp. 2405-2426.
14. Franco, L. & Cannas, S.A. (2001). Generalization Properties of Modular Networks: Implementing the Parity Function. *IEEE Transactions on Neural Networks*, *12*, pp. 1306-1313.

Analysis of the Sanger Hebbian Neural Network

J. Andrés Berzal and Pedro J. Zufiria

Departamento de Matemática Aplicada a las Tecnologías de la Información,
ETSI Telecomunicación, Univ. Politécnica de Madrid

jandres.berzal@terra.es

pzz@mat.upm.es

<http://www.mat.upm.es>

Abstract. In this paper, the behavior of the Sanger hebbian artificial neural networks [6] is analyzed. Hebbian neural networks are employed in communications and signal processing applications, among others, due to their capability to implement Principal Component Analysis (PCA). Different improvements over the original model due to Oja have been developed in the last two decades. Among them, Sanger model was designed to directly provide the eigenvectors of the correlation matrix [8]. The behavior of these models has been traditionally considered on a continuous-time formulation whose validity is justified via some analytical procedures that presume, among other hypotheses, an specific asymptotic behavior of the learning gain. In practical applications, these assumptions cannot be guaranteed. This paper addresses the study of a deterministic discrete-time (DDT) formulation that characterizes the average evolution of the net, preserving the discrete-time form of the original network and gathering a more realistic behavior of the learning gain [13]. The dynamics behavior Sanger model is analyzed in this more realistic context. The results thoroughly characterize the relationship between the learning gain and the eigenvalue structure of the correlation matrix.

1 Introduction

Hebbian architectures, as a tool for on-line data processing, are mainly employed in communications, signal processing and fault diagnosis. They compute the Principal Components of the correlation matrix associated with a given stochastic vector which can represent, for instance, the input data to a channel. At the same time, hebbian networks also perform the Karhunen-Loève transform of input data, very suitable for optimal coding and compression purposes. Among the hebbian architectures, Sanger's neural network is the simplest one directly providing the eigenvector structure in an ordered manner. The dynamics of Sanger net has been analyzed on the basis of a fundamental theorem for the study of stochastic approximation algorithms, relating the discrete-time stochastic model with a deterministic continuous-time (DCT) system [7]. Hence, several studies can be found in the literature employing such deterministic continuous-time formulations in order to indirectly interpret Sanger's net dynamics in the

context of average behavior. All these studies of DCT formulations are grounded on restrictive hypotheses so that the fundamental theorem of stochastic approximation can be applied.

On the other hand, few work has been done so far on the study of the original stochastic discrete formulation when some of such hypotheses cannot be assumed [13]. That new perspective leads to a related deterministic discrete time (DDT) formulation providing information on the relationships between the original stochastic discrete system and the DCT formulation. This paper develops in that direction for Sanger net.

First, the DCT associated system is explicitly solved to provide a global analysis. Secondly, the related DDT formulation is derived and its dynamics are analyzed. This DDT system preserves the discrete-time form of the original network and allows a study of the influence of learning gain evolution. It will be shown that, although both DCT and DDT are derived from the same original system, DDT may be interpreted as a numerical discretization of DCT, and it displays a drastically different behavior.

The paper is structured as follows. Section 2 presents the Sanger hebbian neural net original model. Subsection 2.1 shows the main stochastic approximation result that relates the original stochastic formulation of the net with a continuous-time formulation. The alternative deterministic discrete-time version is presented in Subsection 2.2 where the main results concerning its dynamic global behavior are also demonstrated. Subsection 2.3 analyzes the implications of the time scale hypothesis. Concluding remarks are presented in Section 3.

2 Sanger Neural Network

Sanger Neural Network is defined by the linear input-output relationship and the following learning law:

$$y_i = \sum_{j_1=1}^N w_{ij_1} x_{j_1}, \quad w_{il,n+1} = w_{il,n} + \eta_i y_i \left(x_l - \sum_{j_2=1}^i y_{j_2} w_{j_2 l, n} \right),$$

$$1 \leq i \leq M, \quad 1 \leq M \leq N, \quad 1 \leq l \leq N, \quad \eta_i > 0 \quad (1)$$

where y_i , x_l , w_{il} , $w_{il,n}$ and η_i are the output, the inputs, the weights, the weight values at n -th iteration, and the learning rate, respectively, for neuron i in the net and N is the number of inputs to each neuron and M the number of net neurons (see Figure 1).

2.1 Continuous-Time Formulation

The usual stability analysis follows stochastic approximation techniques and it is based on several hypothesis: statistical independence between inputs \mathbf{x} and weights \mathbf{w}_i , time step tends to zero, etc.

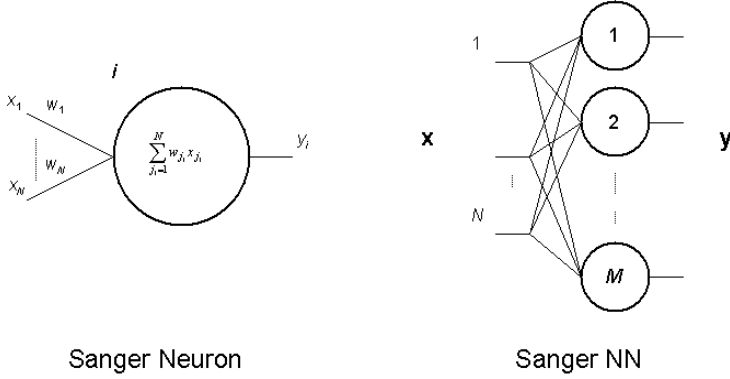


Fig. 1. Functional scheme of NN Sanger neuron and his location in the net with N inputs and M neurons or outputs ($1 \leq M \leq N$)

This approach provides an ODE governing the weights time evolution during the learning phase. For the case of Oja net a Riccati equation is obtained. Here, we present the equation associated with Sanger net:

$$\frac{d\mathbf{w}_i}{dt} = \mathbf{C}\mathbf{w}_i - \sum_{j_1=1}^i (\mathbf{w}_{j_1}^T \mathbf{C}\mathbf{w}_i) \mathbf{w}_{j_1}, \quad \mathbf{C} = E[\mathbf{x}\mathbf{x}^t] \quad (2)$$

Expressing the weights of each neuron on a base of eigenvectors of the input data autocorrelation matrix $\mathbf{w}_i(t) = \sum_{j=1}^N \alpha_{ij}(t) \mathbf{e}_j$ leads to

$$\sum_{j_1=1}^N \frac{d\alpha_{ij_1}}{dt} \mathbf{e}_{j_1} = \sum_{j_2=1}^N \lambda_{j_2} \alpha_{ij_2} \mathbf{e}_{j_2} - \sum_{j_3=1}^i \left(\sum_{j_4=1}^N \alpha_{j_3 j_4} \mathbf{e}_{j_4}^T \sum_{j_5=1}^N \lambda_{j_5} \alpha_{ij_5} \mathbf{e}_{j_5} \right) \sum_{j_6=1}^N \alpha_{j_3 j_6} \mathbf{e}_{j_6} \quad (3)$$

which can be decomposed as

$$\frac{d\alpha_{ip}}{dt} = \lambda_p \alpha_{ip} - \sum_{j_1=1}^i \left(\sum_{j_2=1}^N \alpha_{j_1 j_2} \alpha_{ij_2} \lambda_{j_2} \right) \alpha_{j_1 p} \quad 1 \leq p \leq N. \quad (4)$$

The computation of $\frac{d}{dt} \left(\frac{\alpha_{1p}}{\alpha_{11}} \right)$ leads to a Bernoulli equation which can be solved providing for the first neuron:

$$\alpha_{1p} = \begin{cases} \frac{1}{\pm \sqrt{1 + c_N e^{-2\lambda_1 t} + \sum_{j=2}^N c_{j-1}^2 e^{2(\lambda_j - \lambda_1)t}}}, & p = 1; \\ \frac{c_{1-1} e^{(\lambda_p - \lambda_1)t}}{\pm \sqrt{1 + c_N e^{-2\lambda_1 t} + \sum_{j=2}^N c_{j-1}^2 e^{2(\lambda_j - \lambda_1)t}}}, & 2 \leq p \leq N. \end{cases}$$

Keeping in mind that $\lambda_j - \lambda_1 < 0$, we can always determine $c_j \in \mathbb{R}$ ($1 \leq j \leq N$) for any initial conditions. This implies the asymptotic behavior

$$\lim_{t \rightarrow \infty} \alpha_{1p} = \begin{cases} \pm 1, & p = 1; \\ 0, & 2 \leq p \leq N. \end{cases}$$

which means that

$$\lim_{t \rightarrow \infty} \mathbf{w}_1 = \pm \mathbf{e}_1.$$

For the following neurons a time scale hypothesis is considered (when analyzing i th neuron, smaller index neurons are considered already in steady state regime). Computing $\frac{d}{dt} \left(\frac{\alpha_{ip}}{\alpha_{ii}} \right)$ we also get a Bernoulli equation providing

$$\alpha_{ip} = \begin{cases} \pm \sqrt{\frac{c_{ip} e^{-\lambda_i t}}{1 + \left(2 \sum_{j_1=1}^{i-1} c_{ij_1}^2 \lambda_{j_1} t + c_{iN} \right) e^{-2\lambda_i t} + \sum_{j_2=i+1}^N c_{ij_2-1}^2 e^{2(\lambda_{j_2} - \lambda_i)t}}}, & 1 \leq p < i; \\ \pm \sqrt{\frac{1}{1 + \left(2 \sum_{j_1=1}^{i-1} c_{ij_1}^2 \lambda_{j_1} t + c_{iN} \right) e^{-2\lambda_i t} + \sum_{j_2=i+1}^N c_{ij_2-1}^2 e^{2(\lambda_{j_2} - \lambda_i)t}}}, & p = i; \\ \pm \sqrt{\frac{c_{ip-1} e^{(\lambda_p - \lambda_i)t}}{1 + \left(2 \sum_{j_1=1}^{i-1} c_{ij_1}^2 \lambda_{j_1} t + c_{iN} \right) e^{-2\lambda_i t} + \sum_{j_2=i+1}^N c_{ij_2-1}^2 e^{2(\lambda_{j_2} - \lambda_i)t}}}, & i + 1 \leq p \leq N. \end{cases}$$

Since $\lambda_{j_2} - \lambda_i < 0$, we can also determine $c_{ij} \in \mathbb{R}$ ($1 \leq j \leq N$) for any initial conditions, implying

$$\lim_{t \rightarrow \infty} \alpha_{ip} = \begin{cases} 0, & 1 \leq p \leq i - 1; \\ \pm 1, & p = i; \\ 0, & i + 1 \leq p \leq N. \end{cases}$$

which means that

$$\lim_{t \rightarrow \infty} \mathbf{w}_i = \pm \mathbf{e}_i.$$

Hence, weights \mathbf{w}_i converge to the eigenvectors of the autocorrelation matrix associated with input \mathbf{x} respectively, and the speed of convergence is only determined by the eigenvalue structure of such matrix.

2.2 Discrete-Time Formulation

Following [13] the associated discrete-time formulation is

$$\mathbf{w}_{i,n+1} = \mathbf{w}_{i,n} + \eta_i \left(\mathbf{C} \mathbf{w}_{i,n} - \sum_{j_1=1}^i (\mathbf{w}_{j_1,n}^T \mathbf{C} \mathbf{w}_{i,n}) \mathbf{w}_{j_1,n} \right) \quad (5)$$

A qualitative analysis may be carried out following Lyapunov theory. The fixed points are $\pm \mathbf{e}_k$ ($i \leq k \leq N$), where \mathbf{e}_k is an eigenvector of \mathbf{C} with associated eigenvalue λ_k ($\geq \lambda_{k+1}$). Applying Lyapunov indirect method and considering also the

hypothesis of different time scales between the neurons (when studying stability of i th neuron, neurons with smaller index are assumed to be already in equilibrium), we get

$$\nabla_{\mathbf{w}_{i,n}} \mathbf{w}_{i,n+1} \Big|_{\mathbf{w}_{i,n} = \pm \mathbf{e}_k} = \mathbf{I} + \eta_i \left(\mathbf{C} - \sum_{j_1=1}^{i-1} \mathbf{e}_{j_1} \mathbf{e}_{j_1}^T \mathbf{C} - 2\mathbf{e}_k \mathbf{e}_k^T \mathbf{C} - I(\mathbf{e}_k^T \mathbf{C} \mathbf{e}_k) \right) \quad (6)$$

whose eigenvectors are the same of matrix \mathbf{C} (\mathbf{e}_k) with associated eigenvalues:

$$\lambda_{ikm} = \begin{cases} (1 - \eta_i \lambda_k) & 1 \leq m < i, i \neq 1. \\ (1 - 2\eta_i \lambda_k) & i \leq m \leq N, m = k. i \leq k \leq N. \\ (1 + \eta_i(\lambda_m - \lambda_k)) & i \leq m \leq N, m \neq k. \end{cases} \quad (7)$$

The following table shows the eigenvalues of matrix $\nabla_{\mathbf{w}_{i,n}} \mathbf{w}_{i,n+1} \Big|_{\mathbf{w}_{i,n} = \pm \mathbf{e}_k}$ for the i th neuron in its fixed points $\pm \mathbf{e}_k$ ($i \leq k \leq N$). Analyzing if $|\lambda_{ikm}| < 1$ we conclude that only the pair $\pm \mathbf{e}_i$ is asymptotically stable.

The λ_{ikm} marked with a ‘*’ have modulus larger than 1, for any learning rate η_i , since $\lambda_m - \lambda_k > 0$.

λ_{ikm} , eigenvalue of $\nabla_{\mathbf{w}_{i,n}} \mathbf{w}_{i,n+1} \Big _{\mathbf{w}_{i,n} = \pm \mathbf{e}_k}$	Fixed point k					
	1	...	i	$i+1$...	N
1			$1 - \eta_i \lambda_i$	$1 - \eta_i \lambda_{i+1}$...	$1 - \eta_i \lambda_N$
2			$1 - \eta_i \lambda_i$	$1 - \eta_i \lambda_{i+1}$...	$1 - \eta_i \lambda_N$
...		
i			$1 - 2\eta_i \lambda_i$	$1 + \eta_i(\lambda_i - \lambda_{i+1})^*$...	$1 + \eta_i(\lambda_i - \lambda_N)^*$
$i+1$			$1 + \eta_i(\lambda_{i+1} - \lambda_i)$	$1 - 2\eta_i \lambda_{i+1}$...	$1 + \eta_i(\lambda_{i+1} - \lambda_N)^*$
...		
N			$1 + \eta_i(\lambda_N - \lambda_i)$	$1 + \eta_i(\lambda_N - \lambda_{i+1})$...	$1 - 2\eta_i \lambda_N$

From the table we conclude the asymptotically stable points conditions as follows:

- For eigenvalues with index $m < i$ ($i \neq 1$), where $\lambda_{ikm} = 1 - \eta_i \lambda_k$, the stability condition is
 - $\eta_i < 2/\lambda_k$, and
 - $\lambda_k > 0$ (always true).
- For eigenvalues with index $i \leq m \leq N$ and $m = k$, where $\lambda_{ikm} = 1 - 2\eta_i \lambda_k$ we require
 - $\eta_i < 1/\lambda_k$, and
 - $\lambda_k > 0$ (always true).
- For eigenvalues with index $i \leq m \leq N$ and $m \neq k$ where $\lambda_{ikm} = 1 + \eta_i(\lambda_m - \lambda_k)$ we require
 - $\eta_i < 2/(\lambda_k - \lambda_m)$, and
 - $(\lambda_k - \lambda_m) > 0$ (i.e, $k > m$, if eigenvalues of \mathbf{C} are ordered).

Hence, fixed points $\pm \mathbf{e}_k$ such that $k \geq i + 1$ are unstable (marked with ‘*’ in the Table).

Therefore, we conclude that the only asymptotically stable fixed points for neuron i are $\pm \mathbf{e}_i$ ($k=i$) under the hypotheses that η_i satisfies

$$0 < \eta_i < \min \left\{ \begin{array}{l} \frac{2}{\lambda_i} \quad 1 \leq m \leq i-1, i \neq 1. \\ \frac{1}{\lambda_i} \quad i \leq m \leq N, m = i \\ \frac{2}{\lambda_i - \lambda_m} \quad i \leq m \leq N, m \neq i. \end{array} \right\} = \frac{1}{\lambda_i}, \quad 1 \leq i \leq M. \quad (8)$$

The maximum value for η_i ($< 1/\lambda_i$) implies for each neuron a maximum learning rate (see equation (8)); this value limits the speed of convergence. The optimum value for the learning rate η_i is such that minimizes the absolute value of eigenvalues λ_{ikm} (see equation (7)) within the stability range for the fixed points.

The learning rate η is usually established with a unique value for the whole network such that the stability of every neuron is guaranteed. (η satisfies (8) for all neurons):

$$0 < \eta \leq \min \{ \eta_i \} < 1/\lambda_1, \quad 1 \leq i \leq M.$$

Obviously this way we do not have an optimum convergence rate for each neuron since they have different associated optimal rates.

2.3 Time Scales Hypothesis

In order to apply the time scales hypothesis, some conditions can be imposed to the eigenvalues associated with the different neurons. For instance, the relationship

$$\max \{ |\lambda_{iim}| \} < \min \{ |\lambda_{i+1i+1m}| \} \quad (9)$$

would determine different time scales between the local dynamics of i -th neuron and $i + 1$ -th neuron around their respective asymptotically stable equilibrium points. Hence, equation (9) provides additional restrictions for the allowed ranges of η_i and η_{i+1} .

The computation of the maximum and minimum values

$$\max \{ |\lambda_{iim}| \} = \begin{cases} 1 - \eta_i (\lambda_i - \lambda_{i+1}), & 0 < \eta_i < \frac{2}{3\lambda_i - \lambda_{i+1}}, \\ -1 + 2\eta_i \lambda_i, & \frac{2}{3\lambda_i - \lambda_{i+1}} \leq \eta_i < \frac{1}{\lambda_i}. \end{cases} \quad (10)$$

$$\min \{ |\lambda_{i+1i+1m}| \} = \begin{cases} 1 - 2\eta_{i+1} \lambda_{i+1}, & 0 < \eta_{i+1} < \frac{1}{2\lambda_i}, \\ -1 + 2\eta_{i+1} \lambda_{i+1}, & \frac{1}{2\lambda_i} \leq \eta_{i+1} < \frac{2}{3\lambda_i}, \\ 1 - \eta_{i+1} \lambda_{i+1}, & \frac{2}{3\lambda_i} \leq \eta_{i+1} < \frac{1}{\lambda_i}. \end{cases} \quad (11)$$

And recursively applying equation (9), we get the more restrictive conditions

$$0 < \eta_i < \begin{cases} \frac{1}{\lambda_i} & i = 1. \\ \frac{1}{2} \left(\frac{\lambda_{i-1}}{\lambda_i} - 1 \right) \eta_{i-1} & 2 \leq i \leq M. \end{cases} \quad (12)$$

For each neuron we can select an optimal rate η_{io} to maximize the speed of convergence, minimizing the maximum of $|\lambda_{im}|$

$$\eta_{io} = \frac{2}{3\lambda_i - \lambda_{i+1}}, \quad (13)$$

but this condition may not be compatible with (9). In fact, we could select an optimal rate for the first neuron ($i = 1$), the rest of the rates being imposed by (9).

In case that we want to use a unique learning rate η , the time scale condition will be satisfied if the eigenvalues of matrix \mathbf{C} satisfy:

$$\eta_i = \eta < 1/\lambda_1 \Leftrightarrow \frac{\lambda_i}{\lambda_{i+1}} \geq 3, \quad 1 \leq i \leq M. \quad (14)$$

Hence, if the eigenvalues of \mathbf{C} are close to each other ($\lambda_i/\lambda_{i+1} \ll 3$), we have to gradually decrease the learning rate as the index of the neuron increases following equation (12). On the other hand, if the eigenvalues are far apart, we can even increase such learning rate.

3 Concluding Remarks

Hebbian neural networks have been proven to be very useful for the real time implementation of Principal Component Analysis in an adaptive manner. Among them, Sanger Neural Network is the simplest one providing the eigenvalues of the autocorrelation matrix in an ordered way. So far, only a rigorous analysis of the first neuron had been considered. Here, the analysis of all the neurons has been addressed.

The application of the standard approach for the analysis of Sanger Neural Network leads to a Continuous-Time system which can be globally solved under some time scale asymptotic hypotheses. The solutions of this system conclude the existence of a pair of (almost globally) asymptotically stable equilibria. This optimistic result would guarantee the convergence of the net from any initial condition. Nevertheless, it does not take into account real aspects such as the existence of a lower bound in the learning rate.

The Discrete-Time system proposed here, preserving real conditions and relaxing some of the usual hypothesis, provides a more realistic analysis of Sanger Neural Network. The influence of the learning rate in the convergence of the weights is clearly delineated via a local analysis, based also on a time scale hypothesis. The results show that the size of the learning rate is strongly related to the spectrum of the input covariance matrix in order to preserve stability and good convergence properties. Different design alternatives are evaluated and an upper bound for such learning rate is provided, which is very useful for implementation purposes.

Acknowledgments

This work has been partially supported by project BFM2003-02960 of the Programa Nacional de Promocin General del Conocimiento. MCyT, and project MTM2004-05316 of the Plan Nacional de I+D+i, MEyC, Spain.

References

1. J. A. Berzal and P. J. Zufiria. Algorithms and Implementation Architectures for Hebbian Neural Networks. Proceedings of the International World Conference on Artificial Neural Networks, Springer-Verlang, Berlin Heidelberg 2001. Granada, Spain, 13-15 June 2001 (IWANN'01), 166-173.
2. J. A. Berzal and P. J. Zufiria. Linearized Training of Hebbian Neural Networks: Application to Multispectral Image Processing. Proceedings of the International Conference on Engineering Applications of Neural Networks. Gibraltar, 10-12 June 1998 (EANN'98), 1-8.
3. J. A. Berzal, P. J. Zufiria and J. Cires. Analysis of Hebbian Neural Networks and their Application to Video Sequence Compression. Proceedings of the International Conference on Engineering Applications of Neural Networks. Estocolmo, Sweden, 16-18 June 1997 (EANN'97), 127-136.
4. J. A. Berzal, P. J. Zufiria and L. Rodríguez. Implementing the Karhunen-Loeve Transform via Improved Neural Networks. Proceedings of the International Conference on Engineering Applications of Neural Networks. London, UK, 15-17 June 1996 (EANN'96), 375-378.
5. T. Chen, Y. Hua and W.-Y. Yan. Global Convergence of Oja's Subspace Algorithm for Principal Component Extraction. IEEE Transactions on Neural Networks. January 1998, 58-67.
6. K. I. Diamantaras and S. Y. Kung. Principal Component Neural Networks: Theory and Applications. John Wiley & Sons, Inc, 1994.
7. H. J. Kushner and G. G. Yin, Stochastic Approximation Algorithms and Applications, New York, Springer-Verlag, 1997.
8. T. D. Sanger. Optimal unsupervised learning in a single-layer linear feedforward neural network. Neural Networks. 1992, vol. 2, 459-473.
9. A. Weingessel and K. Hornik. Local PCA Algorithms. IEEE Transactions on Neural Networks. November 2000, 1242-1250.
10. W.-Y. Yan, U. Helmke and J. B. Moore. Global Analysis of Oja's Flow for Neural Networks. IEEE Transactions on Neural Networks. September 1994, 674-683.
11. P. J. Zufiria, J. A. Berzal, M. A. Martínez and J. M. Fernández. Neural Network Processing of Satellite Data for the Nowcasting and Very Short Range Forecasting. Proceedings of the International Conference on Engineering Applications of Neural Networks. Varsovia, Polonia, 13-15 September 1999 (EANN'99), 241-246.
12. P. J. Zufiria and J. A. Berzal. Satellite Data Processing for Meteorological Nowcasting and Very Short Range Forecasting Using Neural Networks. Intelligent Data Analysis, Iospress. 2001, vol. 5, n. 1 , 3-21.
13. Zufiria, P. J.: On the Discrete-Time Dynamics of the Basic Hebbian Neural-Network Node, IEEE Transactions on Neural Networks. November 2002, 1342-1352.

Considering Multidimensional Information Through Vector Neural Networks

J.L. Crespo and R.J. Duro

Grupo de Sistemas Autónomos,
Universidade da Coruña, Spain
{jcm, richard}@udc.es

Abstract. This paper proposes a neural network structure as well as an adaptation of the backpropagation algorithm for its training that provides a way to consider multidimensional information directly in its original space. Traditionally, when inputting multidimensional information to artificial neural networks, its components are fed individually through different inputs and basically processed separately throughout the network. In the present structure, the multidimensional information, in the form of vectors is processed as such in the network, thus preserving in a simple way all the multidimensional neighbourhood relationships. The projection into the dimensionality of the output space is also carried out within the network. This procedure allows for a simpler processing of multidimensional signals such as multi or hyperspectral cubes as used in remote sensing or colour signals in images, which is the example we present as a test for the algorithm.

1 Introduction

Most artificial network structures are well suited for the processing of scalar values at their inputs. In fact, even though feature vectors are often introduced at the inputs of the network for many classification tasks, the internal processing is actually carried out on the components of the vector, that is, without taking any notice of the fact that they are vectors that determine a point in multidimensional space. This makes the tasks of establishing boundaries between categories the more difficult as the networks must establish boundaries for each one of the components individually. To facilitate this task several structures have been proposed from different points of view, depending on the network element on which the authors concentrated. For example, in the case of RBFs [1], the activation function shape is chosen for its properties to best fit data. A good review of different activation functions designed for solving particular problems can be found in [6]. In the case of Gaussian Synapse Multi Layer Perceptron (GSMLP) [2], the network element over which the authors act is the synapse, obtaining a synapse that plays the role of an active filter, this results in a network that can solve certain types of hard signal processing problems.

In this article we will introduce another type of network where a new topological variation is applied in order to make the information processing more natural when

considering multidimensional classification tasks. This is a type of problem that presents some peculiarities arising when high dimensionalities of the input space are considered, as commented in [7]. The network is called the Vectorial Input Higher Order Network (VIHON). The main modification with respect to other networks occurs in the input stage: Every point at the input can be a multidimensional element - vector- and the information processing is carried out in such a way that its preserves its vectorial identity, without dispersing its components throughout the network as in other topologies. In other words, we want to process the vectors as vectors and not their components as scalars.

We could find some similarities between this network and some approaches that consider the use of vectorial information within the networks. But the approaches are different, we have found that the main efforts in this field are focused on the development of non Euclidean input space networks, like hyperboloid [3], or complex numbers [4][5]. These projects are aimed at solving certain types of problems where Euclidean norms and definitions are not valid or are very inadequate. In our case we will reconsider problems where the Euclidean norm is appropriate but where neighbourhood information in the dimensionality of the original space is important in order to establish the different categories.

2 Network Description

The following figure displays a graphic diagram of the feed forward structure of the network proposed here, the VIHON network.

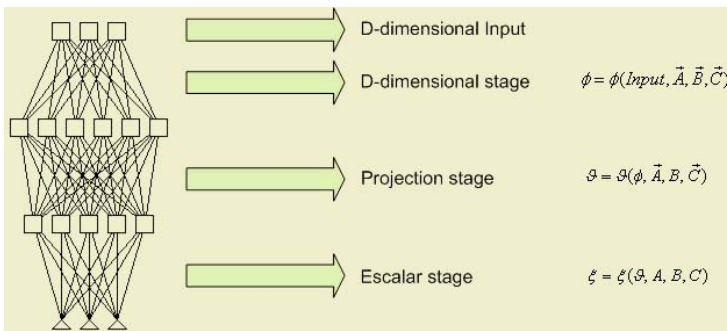


Fig. 1. Structure of the network

The elements we must consider are:

- **Inputs:** Every input of this network is composed of a certain number of elements (that constitute the network window) and every one of these elements is a vector. It is very important to emphasize the fact that this is not only a 2-D generalization of the input of a multilayer network, because in VIHON networks the vectorial character of every input is preserved until a

certain amount of processing has been carried out. The vector data are considered as an entity.

- In the first synapse and processing layer we still have a set of Gaussian synapses, where every component of the vector is processed using a d -dimensional extension of a gaussian function.
- Second synapse and processing layer: It is made up of synapses that are constant deviation d -dimensional gaussian functions –where the constant deviation is necessary in order to allow the projection onto the decision space-, being d the dimension of the input space. This is a very important stage because it is in this layer where the projection task necessary for the information in the dimensionality of the input space to be transformed into the dimensionality of the output space is carried out. In this projection lies the main differential fact of this type of network and it will be discussed later.
- Third synapse layer and output stage: From this point until the end, the network structure is similar to that of the GSMLP, with 1-D Gaussian synapses acting as adaptive filters for scalar inputs, once the projection has been made. The output is controlled by means of a sigmoid neuron.

The operation of the VIHON is similar to a typical GSMLP, except for a number of aspects. First, synapses and activation functions in the first layer are the multidimensional extension for a Gaussian shape and multidimensional extension of a sigmoid activation function. Second, there is a point in the information path across the network in which the projection of the information is carried out. The projection is a critical stage in this network in the sense that this is the differential fact in its operation: we will reduce the dimensionality of the data only after a first processing stage has been carried out in the original space, so that the initial neighbourhood information is preserved, the data in the feature vector are not cut into independently processed pieces. This permits the information carried by the data associated with feature characteristics of the problem to be jointly considered.

3 Training Algorithm of VIHON Network

A training algorithm for this type of network has been developed. It is a variation of classical backpropagation taking into account the multidimensional Gaussian synapses and the fact that the information being processed are vectors. We have chosen the BP algorithm as starting point not only because it is easy to implement, but also because it has a smooth behavior in relation to the error surfaces. A more detailed description of the algorithm can be found elsewhere [8], here we are just going to provide a summary of the main points considering a generic d -dimensional input space.

The three feed forward equations of the network, corresponding to the output, hidden layer and input layers can be written as follows:

$$O_k = \mathfrak{S} \left[\sum_j h_j A_{jk} e^{-B_{jk}(h_j - C_{jk})^2} \right]$$

$$h_j = \mathfrak{S} \left[\sum_l T_l \bar{A}_{lj} e^{-B_{lj}(T_l - \bar{C}_{lj})^2} \right]$$

$$T_l^d = \mathfrak{S} \left[\sum_m I_m^d A_{ml} e^{-B_{ml}^d (I_m^d - C_{ml}^d)^2} \right]$$

If we consider a standard expression for the total error when presenting an input to the network and comparing the output obtained to the desired one:

$$E_{tot} = \sum_k \frac{1}{2} (O_k - T_k)^2$$

Then, following a similar approach to traditional backpropagation and taking into account that the vectorial character of the first layer and that the synapses contain Gaussians with three parameters, center, variance and amplitude (C, B and A), the gradients for the output layer functions, which are one dimensional can be expressed as:

$$\left\{ \begin{array}{l} \frac{\partial O_{Net_k}}{\partial A_{jk}} = h_j e^{B_{jk}(h_j - C_{jk})^2} \\ \frac{\partial O_{Net_k}}{\partial B_{jk}} = h_j A_{jk} (h_j - C_{jk})^2 e^{B_{jk}(h_j - C_{jk})^2} \\ \frac{\partial O_{Net_k}}{\partial C_{jk}} = -2A_{jk} B_{jk} (h_j - C_{jk}) h_j e^{B_{jk}(h_j - C_{jk})^2} \end{array} \right.$$

For the hidden layer we have

$$\frac{\partial h_{Net_j}}{\partial A_{lj}^d} = (T_l^d) e^{B_{lj}(T_l^d - C_{lj}^d)^2}$$

$$\frac{\partial h_{Net_j}}{\partial B_{lj}} = \sum_d (T_l^d A_{lj}^d) \sum_d \left((T_l^d - C_{lj}^d)^2 \right) e^{B_{lj} \left(\sum_d (T_l^d - C_{lj}^d)^2 \right)}$$

$$\frac{\partial h_{Net_j}}{\partial C_{lj}^d} = -2 \sum_d (T_l^d A_{lj}^d) B_{lj} (T_l^d - C_{lj}^d) e^{B_{lj} \sum_d \left((T_l^d - C_{lj}^d)^2 \right)}$$

And finally, for the input layer terms can be expressed as matrices of the form.

$$\frac{\partial T_{Net_l}^d}{\partial A_{ml}^d} = I_m^d e^{B_{ml}^d (I_m^d - C_{ml}^d)^2}$$

$$\frac{\partial T_{Net_l}^d}{\partial B_{ml}^d} = I_m^d A_{ml}^d (I_m^d - C_{ml}^d)^2 e^{B_{ml}^d (I_m^d - C_{ml}^d)^2}$$

$$\frac{\partial T_{Net_l}^d}{\partial C_{ml}^d} = -2I_m^d A_{ml}^d (I_m^d - C_{ml}^d) B_{ml}^d e^{B_{ml}^d (I_m^d - C_{ml}^d)^2}$$

Thus, training a VIHON network is similar to training a multilayer perceptron using backpropagation. The only difference is that now the gradient equations are those stated in this section.

In the following section we introduce an example to test the behaviour of VIHONS and compare them to other types of networks.

4 Application Example: Detection of Spatial and Associated Color Information Patterns over Various Backgrounds

When seeking an example to test VIHON networks in terms of their multidimensional behaviour, certain requirements had to be taken into account. First, the input set must have an intrinsic vectorial character, in the sense that the feature space and problem we are working with requires the preservation of vectorial neighbourhood relationships. Secondly, we require a problem that presents a level of difficulty that allowed us to exclude the use of simple gaussian synapse based networks in order to solve it. Additionally, we sought a multidimensional example where the dimensionality allowed for the intuitive understanding of the operation of the network. Consequently, we chose a problem that involved the detection of spatial patterns with colour information: Traffic sign recognition. In fact, to appropriately classify traffic signs one must take into account neighbourhoods in terms of the colors and neighbourhoods in terms of the spatial patterns involved.

The test and training sets were built out of stripes of traffic sign images that presented variations on the shapes -round shapes, square, rhomboidal...- edge color -red, blue, white, green...- and main color of the sign -white, green, red...etc.- and also the inner region of the sign contained certain types of color and shape information. The examples were sorted randomly on the stripe. The final stripes are swept from left to right both in the training and test phase. In figure 1 we present examples of training and test sets.

Our objective in the experiments was to obtain a network that could detect a given category and the definition of the categories involved several aspects in order to ensure that the network could solve multi criterium recognition problem types. For example, sign shape associated with edge color.



Fig. 2. Examples of part of the stripes used in the experiments

The number of samples in one stripe was between 60 and 120, every sample is a 25 x 25 pixel square image. The stripes were swept in one pixel steps from left to right. Three different experiments were carried out: discrimination between red edge - round shape signs and the rest, discrimination of green as main color - square shape and discrimination of yellow inner region - romboidal shape.

The first attempt we made at obtaining the classification or identification of spatial and colour patterns was to use a standard Gaussian Synapse based ANN and its training algorithm [2], but we did not obtain any positive result. As a second approach, we made use of a set of 3 GSMLP networks working in parallel and processing each one of the three color channels separately. The results were again very poor, as shown in figure 2. Taking a look at what was happening, we noticed that neighbourhood relationships in the RGB space, that is, 3D neighbourhood relations had to be taken into account in order to obtain the appropriate classification and that was very difficult for networks working over the three dimensions separately. That is, we required some type of structure that preserved vectorial neighborhood relations until some degree of processing has been carried out, and this was accomplished through the use of VIHON networks.

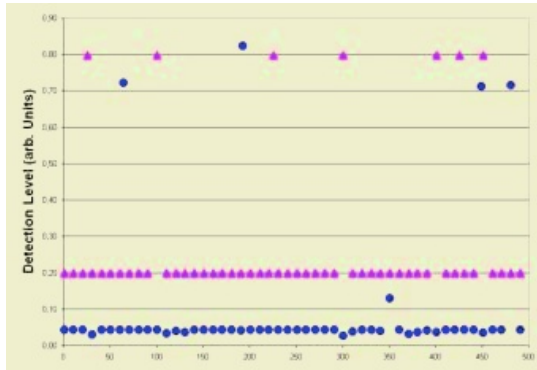


Fig. 3. The graph displays the desired output as triangles and the output obtained as circles for the best results obtained in the experiments in which we considered the three colour channels separately

When we addressed the problem using VIHON networks and the algorithm presented above the results were much better. The resulting networks consisted of 625 input vectors (corresponding to the 25 x 25 pixel window size), each vector was 3

dimensional (RGB), two hidden layers with 20 neurons and one output corresponding to the probability of the input pattern being part of the class. For every one of the training runs, a number of training steps of around 50 were necessary. The results were very encouraging, having obtained a percentage of good detections between 89.5 and 93.5 % and a percentage of false positives with a minimum of 0.9 and a maximum of 6 % depending on the stripes. Some detection results for this case are presented in figure 3. We show the output value of the network as a circle and the desired value as a triangle. It is clear that in this case all of the targets are detected if we consider a 0.5 discrimination level. There seem to be, however, some false positives. But this is not really so, we have to remember that the image is being swept at 1 pixel intervals and thus a few frames have very similar information as shown in the right part of the figure.

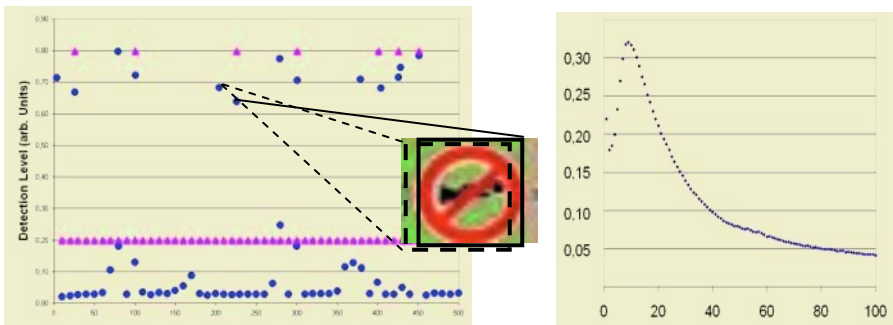


Fig. 4. (left) Results obtained for a portion of the test set in one of the experiments. We have marked as triangles the desired output and the obtained output as circles. We can see that for every desired detected pattern we obtain a positive answer by the network. (Right) the MSE for the training process

The experiments show how the detections were quite appropriate. As indicated before, when a GSBP network was applied to the same problem the results obtained were quite deficient. The problem in this second case had to do with the difficulty of the boundaries that had to be established over all the dimensions of the problem as there was no association during processing of the components of the vectors involved. In the case of the VIHON networks this problem did not exist, as the color information for each pixel was treated as a single vector with 3 components and, consequently, all the neighbourhood relations obtainable on this type of color space could be used to the best advantage during processing. This was carried out in the first layers and after them the projection into 1-d space was carried out in order to obtain the classification value.

5 Conclusions

In this article we present and apply a new type of neural network and training algorithm that is based on the direct consideration of vectorial information in order to

make good use of the neighborhood information from the input space to obtain better and easier classification decisions. This is achieved by means of the introduction of synapses and nodes in the network that directly work with the information in the input space dimension and only after this layer is the information projected into the classification space dimension. The results presented are very encouraging as they provide a way to easily use neighbourhood information in highly dimensional spaces such as those found in remote sensing applications when considering multi or hyperspectral signals.

Acknowledgements

This work was partially funded by Xunta de Galicia under project PGIDIT02PXIB10501PR and the MCYT of Spain under project VEM2003-20088-C04-01.

References

1. Fiesler, E. and Beale, R. (eds. in chief). Handbook of Neural Computation. pp: B1.7:2 – B1.7:3. IOP Publishing Ltd. and Oxford University Press, 1997.
2. Duro, R.J., Crespo, J.L. and Santos, J. Training Higher Order Gaussian Synapses. Foundations and Tools for Neural Modelling. Lecture Notes in Computer Science. Vol. 1606. pp: 537 – 545. Springer-Verlag, Germany, 1999.
3. Buchholz, S. and Sommer, G. A Hyperbolic Multilayer Perceptron Proceedings of the International Joint Conference on Artificial Neural Networks IJCNN2000. Vol. 2. pp: 129 – 133. Como (Italy), 2000.
4. Buchholz, S. and Sommer, G.. Quaternionic spinor MLP. Proceedings of the 8th European Symposium on Artificial Neural Networks ESANN2000. pp: 377 – 382. Bruges (Belgium), 2000.
5. Neural Networks in Multidimensional Domains. Number 234 in LNCIS. Springer-Verlag, 1998
6. Duch, W. and Jankowski, N. Survey of Neural Transfer Functions. Neural Computer Surveys. Num. 2, pp: 163 – 213. 1999
7. Verleysen, M., Francois, D., Simon, G. and Wertz, V.. On the Effects of Dimensionality on Data Analysis with Neural Networks. In Artificial Neural Nets Problem solving Methods. J. Mira, J. R. Alvarez (eds.). Vol. 2. pp: 105 – 112. Springer – Verlag. Berlin Heidelberg, 2003.
8. Red Neuronal Artificial Vectorial basada en Sinapsis Gaussianas de alto Orden. GSA Internal Report. Ref: 01/06-2003

Combining Ant Colony Optimization with Dynamic Programming for Solving the k -Cardinality Tree Problem*

Christian Blum and Maria Blesa

ALBCOM, Dept. Llenguatges i Sistemes Informàtics,
Universitat Politècnica de Catalunya, E-08034 Barcelona, Spain
{cblum, mjblesa}@lsi.upc.edu

Abstract. Research efforts in metaheuristics have shown that an intelligent incorporation of more classical optimization techniques in metaheuristics can be very beneficial. In this paper, we combine the metaheuristic ant colony optimization with dynamic programming for the application to the NP-hard k -cardinality tree problem. Given an undirected graph G with node and/or edge weights, the problem consists of finding a tree in G with exactly k edges such that the sum of the weights is minimal. In a standard ant colony optimization algorithm, ants construct trees with exactly k edges. In our algorithm, ants may construct trees that have more than k edges, in which case we use a recent dynamic programming algorithm to find—in polynomial time—the best k -cardinality tree embedded in the bigger tree constructed by the ants. We show that our hybrid algorithm improves over the standard ant colony optimization algorithm and, for node-weighted grid graph instances, is a current state-of-the-art method.

1 Introduction

The k -cardinality tree (KCT) problem—also referred to as the k -minimum spanning tree (k -MST) problem, or just the k -tree problem—is an NP-hard [13] combinatorial optimization problem which generalizes the well-known minimum weight spanning tree problem. In this paper we deal with a generalized problem version in which the given graph G can have both node and edge weights. More formally, let $G = (V, E)$ be a graph with a weight function $w_E : E \rightarrow \mathbb{N}$ on the edges and a weight function $w_V : V \rightarrow \mathbb{N}$ on the nodes. We denote by \mathcal{T}_k the set of all k -cardinality trees (i.e., trees with exactly k edges) in G . Then, the problem consists of finding a k -cardinality tree $T_k \in \mathcal{T}_k$ that minimizes

* This work was supported by the Spanish CICYT project no. TIC-2002-04498-C05-03 (TRACER), and by the “Juan de la Cierva” program of the Spanish Ministry of Science and Technology of which Christian Blum is a post-doctoral research fellow.

$$f(T_k) = \left(\sum_{e \in E(T_k)} w_E(e) \right) + \left(\sum_{v \in V(T_k)} w_V(v) \right), \quad (1)$$

where $E(T)$ and $V(T)$ denote the edges of the tree T and its nodes, respectively.

The edge-weighted version of the KCT problem was first tackled by exact approaches [14, 8, 17] and heuristics [12, 11, 8]. Soon, the research focused on the development of more appealing metaheuristics: two evolutionary computation approaches [1, 4], three tabu search methods [2, 15, 4], different variations of variable neighborhood search (VNS) [18] and two ant colony optimization (ACO) approaches [7, 4]. Two sets of benchmark instances exist: one was introduced for the empirical evaluation of the VNS-based approaches in [18], and the other one for the metaheuristics proposed in [4]. The variable neighborhood decomposition search (VNDS) algorithm proposed in [18] is the state-of-the-art method for the first set, and the ACO algorithm proposed in [7] is so for the second set.

Less results are known for the node-weighted KCT problem. Greedy-based heuristics were proposed in [12], and the first metaheuristic approaches were presented in [5]. The only existing benchmark set for the node-weighted KCT, together with the currently best metaheuristic (a VNDS), are introduced in [6].

Our contribution. A polynomial-time dynamic programming (DP) algorithm for finding optimal k -cardinality trees in bigger edge-weighted trees was proposed in [16]. This algorithm was used in well-working heuristics for the edge-weighted KCT problem [12]. Recently, we extended this algorithm to be applied to the general (edge and/or node-weighted) KCT problem [3]. We show how a standard ACO algorithm (see [4]) can be improved by applying the DP algorithm in [3] in the following way: Instead of producing k -cardinality trees, the ants produce trees with more than k edges. To these trees we then apply the dynamic programming algorithm in order to obtain the best k -cardinality trees embedded in them. Our experimental results show that our algorithm improves over the standard ACO algorithm for the KCT problem. Moreover, our algorithm is able to improve current state-of-the-art results for node-weighted grid graph instances.

The remainder of the paper is organized as follows. In Section, 2 we describe our hybrid algorithm. In Section 3, we present the experimental evaluation, before we conclude this work in Section 4.

2 The ACO-DP Algorithm

ACO [9, 10] emerged in the early 90's as a novel nature-inspired metaheuristic for the solution of combinatorial optimization problems. The inspiring source of ACO is the foraging behaviour of real ants. When searching for food, ants initially explore the area surrounding their nest in a random manner. As soon as an ant finds a food source, it carries some of the food found to the nest. During the return trip, the ant deposits a chemical pheromone trail on the ground. The quantity of pheromone deposited, which may depend on the quantity and quality of the food, will guide other ants to the food source. The indirect communication established via the pheromone trails allows the ants to find shortest paths

Algorithm 1 ACO-DP for the KCT problem

```

INPUT: a node and/or edge-weighted graph  $G$ , a cardinality  $k < |V| - 1$ , and
a tree size  $l$  with  $k \leq l \leq |V| - 1$ 
 $T_k^{bs} \leftarrow \text{NULL}$ ,  $T_k^{rb} \leftarrow \text{NULL}$ 
 $cf \leftarrow 0$ ,  $bs\_update \leftarrow \text{FALSE}$ 
forall  $e \in E$  do  $\tau_e \leftarrow 0.5$  end forall
while termination conditions not satisfied do
  for  $j = 1$  to  $n_a$  do
     $T_l^j \leftarrow \text{ConstructTree}(\mathcal{T}, l)$ 
    if  $(l > k)$  then  $T_k^j \leftarrow \text{DynamicTree}(T_l^j)$  end if
  end for
   $T_k^{ib} \leftarrow \text{argmin}\{f(T_k^1), \dots, f(T_k^{n_a})\}$ 
   $\text{Update}(T_k^{ib}, T_k^{rb}, T_k^{bs})$ 
   $\text{ApplyPheromoneValueUpdate}(cf, bs\_update, \mathcal{T}, T_k^{ib}, T_k^{rb}, T_k^{bs})$ 
   $cf \leftarrow \text{ComputeConvergenceFactor}(\mathcal{T}, T_k^{rb})$ 
  if  $cf \geq 0.99$  then
    if  $bs\_update = \text{TRUE}$  then
      forall  $e \in E$  do  $\tau_e \leftarrow 0.5$  end forall
       $T_k^{rb} \leftarrow \text{NULL}$ 
       $bs\_update \leftarrow \text{FALSE}$ 
    else
       $bs\_update \leftarrow \text{TRUE}$ 
    end if
  end if
end while
OUTPUT:  $T_k^{bs}$ 

```

between their nest and food sources. This behaviour of real ants in nature is exploited in ACO in order to solve discrete optimization problems using artificial ant colonies.

We combine a standard ACO algorithm [4] with the dynamic programming algorithm in [3], and denote the resulting algorithm as ACO-DP. Algorithm 1 captures the framework of this new hybrid approach. In the pheromone model used, the set of pheromone trail parameters \mathcal{T} contains a parameter \mathcal{T}_e (with pheromone value τ_e) for each $e \in E$. After the initialization of the variables T_k^{bs} (i.e., the best-so-far solution), T_k^{rb} (i.e., the restart-best solution), and cf (i.e., the convergence factor), all the pheromone values are set to 0.5. At every iteration, each of the n_a ants construct probabilistically an l -cardinality tree. If $l > k$, the dynamic programming algorithm from [3] is applied to extract the best k -cardinality tree embedded in the l -cardinality tree. Finally, before the next iteration starts, some of the solutions are used for updating the pheromone values. The details of the methods in this framework are explained in the following.

– **ConstructTree**(\mathcal{T}, l): Henceforth, we denote the two nodes that are connected by an edge e with $v_{e,1}$ and $v_{e,2}$. To build an l -cardinality tree, an ant

starts from an edge e that is chosen probabilistically in proportion to the values $\tau_e / (w_E(e) + w_V(v_{e,1}) + w_V(v_{e,2}))$. At each construction step an ant extends its current tree by adding a node and an edge such that the result is again a tree. Let, at an arbitrary construction step, \mathcal{N} be the set of nodes that fulfill the above condition. For each $v \in \mathcal{N}$ let \mathcal{N}_v be the set of edges that have v as an end-point, ant that have their other end-point—denoted by $v_{e,o}$ —in the current tree. If an ant chooses $v \in \mathcal{N}$ to be added to the current tree, edge $e_{\min} \in \mathcal{N}_v$ that minimizes $w_E(e) + w_V(v_{e,o})$ is also added to the current tree. It remains to be specified how an ant chooses a node $v \in \mathcal{N}$: In $\text{det}\%$ of the cases, an ant chooses the node v that minimizes $w_E(e_{\min}) + w_V(v)$. In $100 - \text{det}\%$ of the cases, v is chosen probabilistically in proportion to $w_E(e_{\min}) + w_V(v)$.

– **DynamicTree(T_l^j)**: This procedure applies the dynamic programming algorithm for node and/or edge-weighted trees in [3] to an l -cardinality tree T_l^j (where j denotes the tree constructed by the j th ant). The algorithm returns the best k -cardinality tree T_k^j embedded in T_l^j .

– **Update($T_k^{ib}, T_k^{rb}, T_k^{bs}$)**: In this procedure T_k^{rb} and T_k^{bs} are set to T_k^{ib} (i.e., the iteration-best solution), if $f(T_k^{ib}) < f(T_k^{rb})$ and $f(T_k^{ib}) < f(T_k^{bs})$, respectively.

– **ApplyPheromoneUpdate($cf, bs_update, \mathcal{T}, T_k^{ib}, T_k^{rb}, T_k^{bs}$)**: In the same way as described in [4], our ACO-DP algorithm may use three different solutions for updating the pheromone values: (i) the iteration-best solution T_k^{ib} , (ii) the restart-best solution T_k^{rb} and, (iii) the best-so-far solution T_k^{bs} . Their influence depends on the convergence factor cf , which provides an estimate about the state of convergence of the system. To perform the update, first an update value ξ_e for every pheromone trail parameter $\mathcal{T}_e \in \mathcal{T}$ is computed:

$$\xi_e \leftarrow \kappa_{ib} \cdot \delta(T_k^{ib}, e) + \kappa_{rb} \cdot \delta(T_k^{rb}, e) + \kappa_{bs} \cdot \delta(T_k^{bs}, e) ,$$

where κ_{ib} is the weight of T_k^{ib} , κ_{rb} the weight of T_k^{rb} , and κ_{bs} the weight of T_k^{bs} such that $\kappa_{ib} + \kappa_{rb} + \kappa_{bs} = 1.0$. The δ -function is the characteristic function of the set of edges in the tree, i.e., for each k -cardinality tree T_k ,

$$\delta(T_k, e) = \begin{cases} 1 & : e \in E(T_k) \\ 0 & : \text{otherwise} \end{cases}$$

Then, the following update rule is applied to all pheromone values τ_e :

$$\tau_e \leftarrow \min \{ \max \{ \tau_{\min}, \tau_e + \rho \cdot (\xi_e - \tau_e) \}, \tau_{\max} \} ,$$

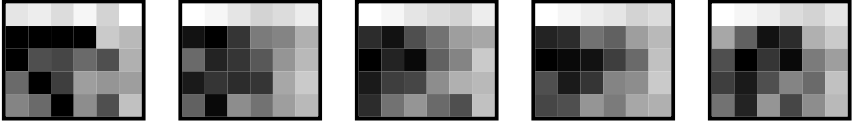
where $\rho \in (0, 1]$ is the evaporation (or learning) rate. The upper and lower bounds $\tau_{\max} = 0.99$ and $\tau_{\min} = 0.01$ keep the pheromone values always in the range $(\tau_{\min}, \tau_{\max})$, thus preventing the algorithm from converging to a solution. After tuning, the values for ρ , κ_{ib} , κ_{rb} and κ_{bs} are chosen as shown in Table 1.

– **ComputeConvergenceFactor(\mathcal{T}, T_k^{rb})**: This function computes, at each iteration, the convergence factor as

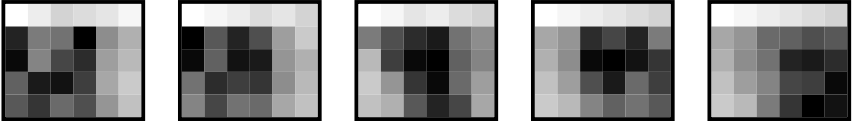
$$cf \leftarrow \frac{\sum_{e \in E(T_k^{rb})} \tau_e}{k \cdot \tau_{\max}} ,$$

Table 1. The schedule used for values ρ , κ_{ib} , κ_{rb} and κ_{bs} depending on cf (the convergence factor) and the Boolean control variable bs_update

	$bs_update = \text{FALSE}$			$bs_update = \text{TRUE}$
	$cf < 0.7$	$cf \in [0.7, 0.9)$	$cf \geq 0.9$	
ρ	0.05	0.1	0.15	0.15
κ_{ib}	2/3	1/3	0	0
κ_{rb}	1/3	2/3	1	0
κ_{bs}	0	0	0	1



(a) Tuning results for 20x20 random node-weighted grid graphs. From left to right, the results for cardinalities 40, 80, 120, 160, and 200 are shown



(b) Tuning results for edge-weighted graphs with 500 nodes. From left to right, the results for cardinalities 50, 100, 150, 200, and 250 are shown

Fig. 1. Tuning results. In each of the five matrices in (a) and (b), the columns correspond to the 6 values of det , and the rows to the 5 values of l (e.g., the matrix position (2, 2) corresponds to the gray value of the tuple ($det = 80, l = k + s$))

where τ_{\max} is again the upper limit for the pheromone values. The convergence factor cf can therefore only assume values between 0 and 1. The closer cf is to 1, the higher is the probability to produce the solution T_k^{rb} .

3 Experimental Results

We implemented ACO-DP in C++, and run experiments on a PC with a 3 GHz Intel Pentium IV processor and 1 Gb memory. In the previous section we specified the values of all parameters of ACO-DP except for two crucial ones: det , the percentage of deterministic steps during the tree construction, and l , the size of the trees constructed by the ants. For their determination, we conducted a parameter

tuning by running ACO-DP for all combinations of $det \in \{75, 80, 85, 90, 95, 99\}$ and $l \in \{k, k + s, k + 2s, k + 3s, k + 4s\}$, where $s = (|V| - 1 - k)/4$.

To obtain instances for tuning, we randomly generated 10 20x20-grid node-weighted graph instances, and applied ACO-DP exactly once for each (det, l) combination and for each cardinality $k \in \{40, 80, 120, 160, 200\}$. This gives a value averaged over the 10 graphs for each (det, l, k) triple. For each k , we then ranked the resulting 30 values and translated them into gray scale: the best of the 30 values received gray value 1.0 (i.e., black), and the worst received gray value 0.0 (i.e., white). These results, expressed in gray values, are shown in Figure 1(a). We did the same for the 10 edge-weighted graphs with 500 nodes from the benchmark set proposed in [18] (for cardinalities $k \in \{50, 100, 150, 200, 250\}$). The results are shown in Figure 1(b).

Both for node-weighted grid graphs and for edge-weighted graphs, the tuning results show that a setting $l > k$ is always better, which means that ACO-DP always improves on the standard ACO algorithm. With respect to the tuning results, we chose the setting ($det = 85, l=k+2s$) for the application of ACO-DP to node-weighted grid graphs, and the setting ($det = 95, l = k + 2s$) for the application to edge-weighted graphs. With these settings, we applied our algorithm to some of the existing benchmarks.¹

Results for the node-weighted benchmark. We applied ACO-DP to all node-weighted grid graph instances from [6], i.e., 10 30x30 instances, another 10 of 40x40, and another 10 with 50x50 nodes. The results are shown in Table 2. The first table column indicates the graph type, and the second one indicates the tested cardinality. We compared our results to the state-of-the-art algorithm VNDS in [6] and to the heuristic DynamicTree (Prim) in [3]; the latter constructs a spanning tree of the given graph, and applies the dynamic programming algorithm to it. For these approaches, together with our ACO-DP approach, the table shows the average value obtained for each (n, k) combination (note that each algorithm was applied exactly once to each of the 10 instances of a graph type). The best value for each (n, k) combination is given in bold font. For ACO-DP we additionally provide the relative deviations (in %) from VNDS (headed by $d_{vnds}/\%$), and from DynamicTree (Prim) (headed by $d_{dtp}/\%$). The last table row provides the averages over these relative deviations. The machine used for running VNDS is about 7 times slower than our machine. Therefore, we used as time limits for ACO-DP one seventh of the time limits of VNDS in [6].

The results show that ACO-DP is on average 1.28 % better than VNDS, and 2.51 % better than DynamicTree (Prim). They also show that ACO-DP seems in general to have advantages for smaller cardinalities, and that ACO-DP seems to become better—in comparison to VNDS—with growing problem instance size.

¹ Concerning the computational overhead, note that with $l = k + 2s$ the time needed for constructing a solution for our smallest cardinality takes about 6 times longer than with $l = k$. While for our biggest cardinality the time needed for constructing a solution is only about 0.3 times longer than with $l = k$.

Table 2. Results for node-weighted grid graphs in [6]. As time limits for ACO-DP we used one seventh of the time limits of VNDS as given in [6]

n	k	VNDS	DynamicTree (Prim)	ACO-DP		
		q_{vnds}	q_{dtp}	q_{aco}	$d_{vnds}/\%$	$d_{dtp}/\%$
30x30	100	8571.9	8612.2	8203.7	-4.2954	-4.7432
	200	17994.4	18491.6	17850.1	-0.8019	-3.4691
	300	28770.9	29488.6	28883.9	0.3927	-2.0506
	400	42114	42618.6	42331.9	0.5174	-0.6727
	500	59266.4	59662	59541.7	0.4645	-0.2016
40x40	150	18029.9	18495.3	17527.1	-2.7887	-5.2348
	300	38965.9	39220.4	37623.8	-3.4442	-4.0708
	450	61290.1	62118.3	60417	-1.4245	-2.7388
	600	86422.3	87935.4	86594.7	0.1994	-1.5246
	750	117654	119303	118570	0.7785	-0.6140
50x50	250	37004	38007.1	35995.2	-2.7261	-5.2934
	500	81065.8	80247.9	77309.9	-4.6331	-3.6611
	750	128200	128224	125415	-2.1723	-2.1908
	1000	182220	184103	181983	-0.1300	-1.1516
	1250	250962	253116	253059	0.8355	-0.0224
$\bar{d}/\%$					-1.2818	-2.5093

Table 3. Results for edge-weighted graphs in [18]. As time limits for ACO-DP we used one tenth of the time limits of VNDS as given in [18]

n	k	VNDS	DynamicTree (Prim)	ACO-DP		
		q_{vnds}	q_{dtp}	q_{aco}	$d_{vnds}/\%$	$d_{dtp}/\%$
1000	100	5828	5841.9	5827.2	-0.0137	-0.2516
	200	11893.7	11927.5	11910.1	0.1378	-0.1458
	300	18196.6	18225.6	18217.4	0.1143	-0.0449
	400	24734	24766.4	24757.8	0.0962	-0.0347
	500	31561.8	31593.3	31613.8	0.1647	0.0648
2000	200	23538.3	23543.6	23479	-0.2519	-0.2743
	400	48027.3	48086.2	48030.4	0.0064	-0.1160
	600	73277.8	73394.2	73392.9	0.1570	-0.0017
	800	99491.2	99623.2	99801.4	0.3117	0.1788
	1000	126485	126916	127325	0.6639	0.3222
3000	300	35186.1	35203.5	35160.3	-0.0733	-0.1227
	600	71634.7	71729.2	71862.8	0.3184	0.1862
	900	109463	109631	110094	0.5764	0.4223
	1200	148826	148997	149839	0.6803	0.5649
	1500	189943	190080	191627	0.8867	0.8141
$\bar{d}/\%$					0.2516	0.1041

Results for the edge-weighted benchmark. We also applied ACO-DP to some of the edge-weighted graph instances from [18], i.e., 10 instances with 1000, 2000 and 3000 nodes, respectively. The results are shown in Table 3 in the same way as outlined for Table 2. Note that also a VNDS (a different one) is the current state-of-the-art algorithm for these instances. The machine that was used to run VNDS is about 10 times slower than our machine. Therefore, we used as time limits for ACO-DP one tenth of the time limits of VNDS in [18].

The results show that in contrast to the node-weighted grid graph case, ACO-DP seems not to reach the state-of-the-art results; ACO-DP can improve them only for small cardinalities. This might be due to the much higher density of the edge-weighted graph instances (w.r.t. the node-weighted grid graph instances). Note that the difference in quality is minimal; on average ACO-DP is only 0.25% worse than VNDS, and 0.1% worse than DynamicTree (Prim).

4 Conclusions

A successful combination of ACO and dynamic programming has been proposed for solving a general (edge and/or node-weighted) version of the NP-hard k -cardinality tree problem. The results obtained indicate that the hybrid algorithm outperforms the standard ACO algorithm. Furthermore, for node-weighted grid graphs, the hybrid algorithm outperforms the current state-of-the-art VNDS method. Further experiments have shown that the results obtained depend on topological properties of the graphs (e.g., density, d -regularity, etc.) rather than the tackled problem version (i.e., edge or node weighted).

References

1. M. Blesa, P. Moscato, and F. Xhafa. A memetic algorithm for the minimum weighted k -cardinality tree subgraph problem. In *Proc. of the 4th Metaheuristic Conf.*, volume 1, pages 85–90, 2001.
2. M. Blesa and F. Xhafa. A C++ implementation of tabu search for k -cardinality tree problem based on generic programming and component reuse. In *Net.ObjectDays 2000 Tagungsband*, pages 648–652, Erfurt, Germany, 2000.
3. C. Blum. Revisiting dynamic programming for finding optimal subtrees in trees. Tech. Rep. LSI-04-57, Universitat Politècnica Catalunya, Spain, 2004. Submitted.
4. C. Blum and M. Blesa. New metaheuristic approaches for the edge-weighted k -cardinality tree problem. *Computers & Operations Research*, 32:1355–1377, 2005.
5. C. Blum and M. Ehrgott. Local search algorithms for the k -cardinality tree problem. *Discr. Appl. Math.*, 128:511–540, 2003.
6. J. Brimberg, D. Urošević, and N. Mladenović. Variable neighborhood search for the vertex weighted k -cardinality tree problem. *Eur. J. of Op. Res.*, 2005. In press.
7. T. N. Bui and G. Sundarraj. Ant system for the k -cardinality tree problem. In K. Deb et al., editor, *Proc. of the Genetic and Ev. Comp. Conference-GECCO 2004*, volume 3102 of LNCS, pages 36–47. Springer, 2004.
8. S. Y. Cheung and A. Kumar. Efficient quorumcast routing algorithms. In *Proceedings of INFOCOM'94*, Los Alamitos, USA, 1994. IEEE Society Press.

9. M. Dorigo. *Optimization, Learning and Natural Algorithms* (in Italian). PhD, Dipartimento di Elettronica, Politecnico di Milano, Italy, 1992.
10. M. Dorigo, V. Maniezzo, and A. Coloni. Ant System: Optimization by a colony of cooperating agents. *IEEE TSMC – Part B*, 26:29–41, 1996.
11. M. Ehrgott and J. Freitag. K.TREE / K.SUBGRAPH: A program package for minimal weighted k -cardinality-trees and -subgraphs. *Eur. J. of Op. Res.*, 1:214–225, 1996.
12. M. Ehrgott, J. Freitag, H. W. Hamacher, and F. Maffioli. Heuristics for the k -cardinality tree and subgraph problem. *Asia-Pac. J. of Op. Res.*, 14:87–114, 1997.
13. M. Fischetti, H. W. Hamacher, K. Jörnsten, and F. Maffioli. Weighted k -cardinality trees: Complexity and polyhedral structure. *Networks*, 24:11–21, 1994.
14. J. Freitag. *Minimal k -cardinality trees* (in German). Master’s thesis, Department of Mathematics, University of Kaiserslautern, Germany, 1993.
15. K. Jörnsten and A. Løkketangen. Tabu search for weighted k -cardinality trees. *Asia-Pac. J. of Op. Res.*, 14:9–26, 1997.
16. F. Maffioli. Finding a best subtree of a tree. Tech. Rep. 91.041, Politecnico di Milano, Dipartimento di Elettronica, Italy, 1991.
17. R. Uehara. The number of connected components in graphs and its applications. IEICE Tech. Rep. COMP99-10, Komazawa University, Japan, 1999.
18. D. Urošević, J. Brimberg, and N. Mladenović. Variable neighborhood decomposition search for the edge weighted k -cardinality tree problem. *Comp. & Op. Res.*, 31:1205–1213, 2004.

A Basic Approach to Reduce the Complexity of a Self-generated Fuzzy Rule-Table for Function Approximation by Use of Symbolic Interpolation

G. Rubio and H. Pomares

Department of Computer Architecture and Computer Technology,
University of Granada (18071), Spain

Abstract. There are many papers in the literature that deal with the problem of the design of a fuzzy system from a set of given training examples. Those who get the best approximation accuracy are based on TSK fuzzy rules, which have the problem of not being as interpretable as Mamdany-type Fuzzy Systems. A question now is posed: How can the interpretability of the generated fuzzy rule-table base be increased? A possible response is to try to reduce the rule-base size by generalizing fuzzy-rules consequents which are symbolic functions instead of fixed scalar values or polynomials, and apply symbolic interpolations techniques in fuzzy system generation. A first approximation to this idea is presented in this paper for 1-D functions.

1 Introduction

The problem of estimating an unknown function f from samples of the form (\vec{x}^k, z^k) ; $k=1,2,\dots,K$; with $z^k = f(\vec{x}^k) \in \mathbb{R}$ and $\vec{x} \in \mathbb{R}^N$ (i.e. function approximation from a finite number of data points), has been and is still a fundamental issue in a variety of scientific and engineering fields. Inputs and outputs can be continuous and/or categorical variables. This paper is concerned with continuous output variables, thus considering regression or function approximation problems [4].

Generally, there are three ways to solve the function approximation problem from a set of numerical data:

1. by building a mathematical model for the function to be learned
2. by building a model-free system
3. by seeking human experts' advice

One limitation of the first method is that accurate mathematical models for complex non-linear systems either do not exist or can only be derived with great difficulty. Therefore, the theory of traditional equation-based approaches is well developed and successful in practice only for linear and simple cases [4].

Recently, model-free systems, such as artificial neural networks or fuzzy systems, have been proposed to avoid the knowledge-acquisition bottleneck [1], [2].

Fuzzy systems provide an attractive alternative to the “black boxes” characteristic of neural network models, because their behavior can be easily explained by a human being.

Many fuzzy systems that automatically derive fuzzy IF-THEN rules from numerical data have been proposed in the bibliography to overcome the problem of knowledge acquisition [2], [6]. An important study in this context was carried out in [8].

The approaches presented in [3],[7],[6] need a fixed structure for the rules. However, the distribution of the membership functions (shape and location) has a strong influence on the performance of the systems. It is usually difficult to define and tune the membership functions and rules. These limitations have justified and encouraged the creation of intelligent hybrid systems that overcome the limitations of individual techniques. Genetic algorithms (GA's) and artificial neural networks (ANN's) offer a possibility to solve this problem [2], [1], [5].

This paper proposes a generalization based on the learning method proposed in [4] to automatically obtain the optimum structure of a fuzzy system and derive fuzzy rules and membership functions from a given set of training data, using a hybridization between fuzzy systems and traditional equation-based approaches using symbolic interpolation. We will propose and study a basic application of the idea for one dimensional (1-D) continuous functions. Our aim is to obtain an analytical partitioned description of a 1-D function domain, using symbolic interpolation to determine some possible analytical equations for each partition while, at the same time, the number and definition of each partition is optimized. Each partition of the input domain is associated with a trapezoidal membership function that can be intuitively interpreted without effort.

2 Statement of the Problem

We consider the problem of approximating a continuous single-input single-output function to clarify the basic ideas of our approach, since the extension of the method to a multiple-input is straightforward. Let us consider a set D of desired input-output data pairs, derived from an unknown 1-D function or system F. Each vector datum can be expressed as $(x, y)^k$ and $k=1,2,\dots,K$. Our fuzzy system comprises a set of n IF-THEN fuzzy rules having the following form:

$$IF\ x\ is\ X_i\ THEN\ y = f_i(x) \quad (1)$$

where $i=1..n$ with n being the number of membership functions of the input variable and $f_i(x)$, $j=1..n$, is a analytical equation associated to rule (partition of x domain). Using the above notation, the output of our fuzzy system can be expressed as follows:

$$\tilde{F}(x^k; R, C) = \frac{\sum_{i=1}^n f_i(x) \cdot U_i(x)}{\sum_{i=1}^n U_i(x)} \quad (2)$$

where an explicit statement is made of the dependency of the fuzzy output, not only on the input vector, but also on the matrix of rules R and on all the

parameters that describe the membership functions C. The problem considered in this paper may be stated in a precise way as that of finding a configuration C and generating a set of fuzzy rules from a data set D of K input-output pairs, such that the fuzzy system correctly approximates the unknown function F. The function to be minimized is the sum of squared errors:

$$J(R, C) = \sum_{k \in D} (F(x^k) - \tilde{F}(x^k; R, C))^2. \tag{3}$$

The index selected to determine the degree of accuracy of the obtained fuzzy approximation is the Normalized Root-Mean-Square Error (NRMSE) defined as:

$$NRMSE = \sqrt{\frac{\bar{e}^2}{\sigma_y^2}} \tag{4}$$

where σ_y^2 is the mean-square value of the output data, and \bar{e}^2 is the mean-square error between the obtained and the desired output. This index is independent of scale factors or number of data.

3 Proposed Approach

In this section we present the basics of the algorithm we have implemented and studied. Figure 1 shows a flowchart describing the structure of the algorithm.

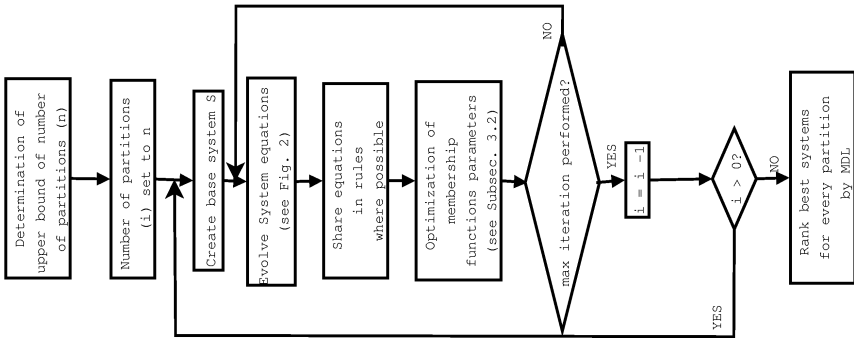


Fig. 1. Algorithm’s flowchart

Before starting the algorithm, we must find an upper bound to the number of partitions for the input variable. For this purpose, we compute the number of local maxima and minima of the underlying function, with the idea that in the worst case, linear functions can be use to approximate the segments in between. Throughout the algorithm, the number of partitions will be optimized starting from this maximal value. As can be seen from Fig. 1, the algorithm performs a search for the best system in a top-down fashion, i.e. starting from the biggest

number of partitions, it tries to optimize both the location of each partition and the function used in the consequent of each fuzzy rule. After generating every system, it is analyzed to try to share equations between rules in order to reduce the number of rules. The procedure is to try to apply for each rule, the consequents (equations) of rules of adjacent partitions and use the best one. The procedure iterates until no new change can be performed. Finally, we optimize the membership functions parameters (see subsection 3.2) using the Levenberg-Marquardt algorithm and calculate the NRMSE for the final system.

For a given number of partitions, the algorithm in charge of making such optimization is presented in figure 2.

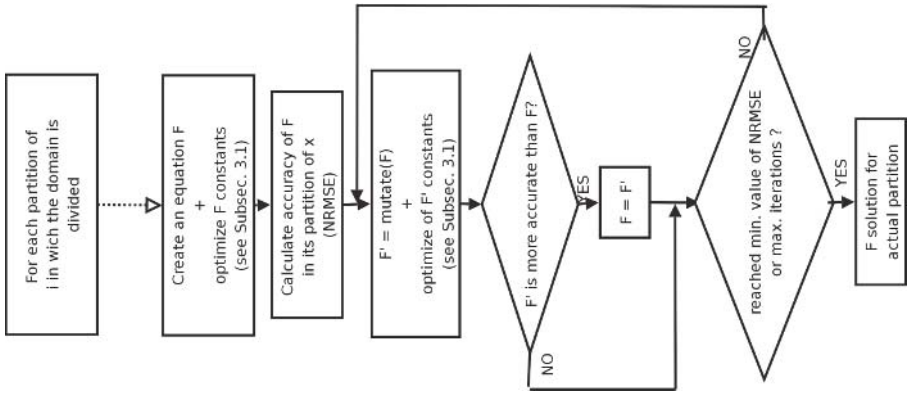


Fig. 2. Local search algorithm for optimization of equation structure

Figure 2 represents the sub-algorithm that is the core of our method. It considers each one of the i partitions at a time. For each partition, an equation F is generated and its constants optimized as explained in subsection 3.1; then iteratively we generate mutated forms of F , optimizing its constants and we accept or reject it if it is better than the actual F . The algorithm iterates until a minimum value of NRMSE or a maximum number of iterations performed are not reached.

3.1 Representation of Rules Consequents: Symbolic Equations

In order to have the least limited possible set of usable functions as consequents and use symbolic integration techniques, we have represented them as a vector of elements that represent an equation in Reverse Polish Notation (RPN) with a set of possible nodes:

1. Variable: x .
2. Constant: a real value.
3. Binary operators: $+$, $-$, $*$, $'$ safe $/$ ' . We have defined 'a safe b ' as $\frac{a}{1+b^2}$, the reason is to prevent divisions by zero.

4. Unary operators: - (negation), cos, sin, tanh, atan, ‘safe $\ln(a)$ ’, ‘limited e^a ’. We have defined ‘safe $\ln(a)$ ’ as $\ln(1+a^2)$, the reason is to prevent ∞ values; and ‘limited e^a ’ as e^{-a^2} , the reason is to prevent strong function value variations.

Nodes that are ‘variable’ or ‘operator’ have associated 2 implicit constants: 1 multiplicative and another additive. Note that these constants are distinct than the ‘constant’ node type. This large number of constants by equation gives a lot of flexibility to the procedure and helps in the optimization stage (for example, the input and output domains can be implicitly normalized in an automatic way).

To limit the complexity of the results we have quoted the size of the equations to 10 elements in the executions.

With this representation, we can apply “mutation” operators that make possible to use local search algorithms to equations in order to optimize their structure itself. A mutation consist in 75% of probability of choosing one of the nodes in the equation tree and substitute all the subtree beyond by another one of the same length; and 25% of probability of substituting the equation by a new one.

3.2 Membership Functions of the System

Intuitively, the results of the algorithm should be an indication of the kind of functions that seem to follow the output of the function to model, in some intervals of the input variable; but we should not forget the “transitions” between them. For these, trapezoidal membership functions can be of much help to implement that idea.

As is well known, we can define a trapezoidal membership function over the interval $[0,1]$ using 4 parameters, $0 \leq z1 \leq c1 \leq c2 \leq z2 \leq 1$. In our approach, we have also restricted the membership functions making them share some parameters in order to insure a soft union of functions parts, in the form determined by their indexes. For example, for $t + 1$ partitions, we have:

$$\begin{aligned}
 U_0(x) &= \begin{cases} 1 & 0 \leq x < c_{0,2} \\ \frac{c_{1,1}-x}{c_{1,1}-c_{0,2}} & c_{0,2} \leq x < c_{1,1} \\ 0 & \textit{otherwise} \end{cases} \\
 U_i(x) &= \begin{cases} \dots \\ \frac{x-c_{i-1,2}}{c_{i,1}-c_{i-1,2}} & c_{i-1,2} \leq x < c_{i,1} \\ 1 & c_{i,1} \leq x < c_{i,2} \\ \frac{c_{i+1,1}-x}{c_{i+1,1}-c_{i,2}} & c_{i,2} \leq x < c_{i+1,1} \\ 0 & \textit{otherwise} \end{cases} \quad (5) \\
 U_t(x) &= \begin{cases} \dots \\ \frac{x-c_{t-1,2}}{c_{t,1}-c_{t-1,2}} & c_{t-1,2} \leq x < c_{t,1} \\ 1 & c_{t,1} \leq x < 1 \\ 0 & \textit{otherwise} \end{cases}
 \end{aligned}$$

4 Simulation Results

To see how the proposed approach works and to facilitate its understanding, in this Section we use an artificial example generated combining the following three functions (see Fig. 3)

$$\begin{aligned}
 f_1(x) &= e^{-5x} \cdot \sin(2\pi x) \\
 f_2(x) &= \ln(1 + x^2) + 0.2 \\
 f_3(x) &= 0.25 * \sin(5\pi x + \pi/2) + 0.6
 \end{aligned}
 \tag{6}$$

using trapezoidal membership functions as

$$\begin{aligned}
 f(x) &= f_1(x)U(x, 0, 0, 0.13, 0.33) + \\
 &f_2(x)U(x, 0.13, 0.33, 0.47, 0.67) + f_3(x)U(x, 0.47, 0.57, 1, 1).
 \end{aligned}
 \tag{7}$$

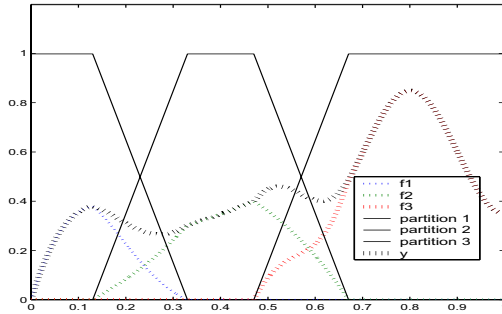


Fig. 3. Membership functions, original functions and the combination of them used in the example

We decided to parse the function with 1000 homogenously distributed points, without noise, and applied the proposed algorithm with the results given by Tables 1 and 2:

Table 1. Statistical results obtained by the proposed methodology for the example

N of Partitions	Best NRMSE	MEAN NRMSE	VAR NRMSE	AVG of final N of rules
5	0.060	0.066	0.000	5.000
4	0.064	0.078	0.000	3.000
3	0.094	0.134	0.000	3.000
2	0.129	0.178	0.034	2.000
1	0.034	0.113	0.008	1.000

Table 2. Best result obtained by the algorithm

Nodes	x	x	cos	*	x	tanh	x	'safe ln'	*	+
constants										
Multiplicative	-2.002	-7.362	0.601	-0.481	-6.830	0.055	9.027	-0.078	1.809	-4.968
Additive	0.722	1.302	-0.398	-1.094	1.933	1.149	-5.939	1.801	-1.761	2.640

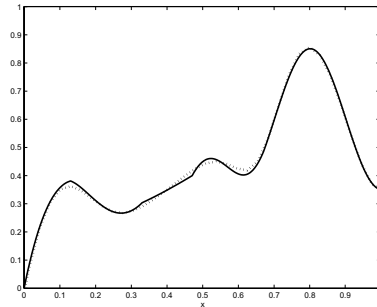


Fig. 4. Output (dotted line) of the best solution found by the algorithm

Table 3. Number of fuzzy rules needed to get an NRMSE similar to the one obtained by the proposed algorithm

N of Rules	NRMSE
40	0.061
35	0.062
30	0.063
25	0.065
20	0.071
15	0.088
10	0.149

It is interesting to note that the solution with just one partition (therefore with a single expression) has been the one with the best performance result. The coded equation generated for this case was the one presented in Table 2.

In order to compare with other works, we have applied an adaptation of the method presented in [4] to obtain fuzzy systems with NRMSE similar to the one generated by our method. Table 3 shows these results, where we can see that very complex (grid-based) fuzzy systems are needed for this case in order to have an accuracy which is still worse than that obtained by the proposed approach.

Thus, applying symbolic integration to obtain structured fuzzy rules can reduce the complexity of the fuzzy system, in the sense that less number of rules are needed (in some cases, in a very dramatic proportion). But the consequents themselves are a bit complicated to interpret; it's a part that needs further

research but we think that this is a very interesting approach to the reduction of the fuzzy rule base.

5 Conclusions

This paper has dealt with the problem of reducing the size of the rule base in the design of a fuzzy system from a set of given training examples. For that purpose, we have generalized the fuzzy-rules consequents allowing them to be symbolic functions instead of fixed scalar values or polynomials, and applying symbolic interpolations technics in the fuzzy system generation. The results indicate that this is a possible good application of symbolic interpolation to reduce the fuzzy rules base complexity, but it's a little far from being effective to increase the interpretability of the whole system. In future works, the method will be extend to multiple-input, single-input functions, using Gaussian membership functions, more practical for n-D problems. We will also try to define a theory about symbolic equation creation and operation for human understandability, with the idea to endow the equations generated with as much relevant meaning as possible.

References

1. M.Funabashi, A.Maeda, "Fuzzy and neural hybrid expert systems: synergetic AI", IEEE Expert, pp.32-40,1995.
2. J.S.R.Jang, C.T.Sun, E.Mizutani, "Neuro-Fuzzy and soft computing", Prentice Hall, ISBN 0-13-261066-3, 1997.
3. B.Kosko, Neural Networks and Fuzzy Systems. Englewood Cliffs, NJ; Prentice-Hall, 1992.
4. H.Pomares, I.Rojas, J.Ortega, J.Gonzalez, A. Prieto "A Systematic Approach to a Self-Generating Fuzzy Rule-Table for Function Approximation", IEEE Trans. on Syst. Man. and Cyber., vol.30, no.3, June 2000.
5. I.Rojas, J.J.Merelo, J.L.Bernier, A.Prieto, "A new approach to fuzzy controller designing and coding via genetic algorithms", IEEE International Conference on Fuzzy Systems, Barcelona, pp.1505-1510, July 1997.
6. R.Rovatti, R.Guerrieri, "Fuzzy sets of rules for system identification", IEEE Trans.on Fuzzy Systems, vol.4, no.2, pp.89-102, 1996.
7. T.Sudkamp, R.J.Hammell, "Interpolation, Completion, and Learning Fuzzy Rules", IEEE Trans. on Syst. Man and Cyber., vol.24, no.2, February, 1994, pp.332-342.
8. L.X.Wang, J.M.Mendel, "Generating fuzzy rules by learning from examples", IEEE Trans. On Syst. Man and Cyber, vol.22, no.6, November/December, pp.1414-1427, 1992.

Average Time Complexity of Estimation of Distribution Algorithms

C. González, A. Ramírez, J.A. Lozano, and P. Larrañaga

Department of Computer Science and Artificial Intelligence,
University of the Basque Country, San Sebastián, Spain
{ccpgomoc, ccploalj, ccplamup}@si.ehu.es
juerware@hotmail.com

Abstract. This paper presents a study based on the empirical results of the average first hitting time of Estimation of Distribution Algorithms. The algorithms are applied to one example of linear, pseudo-modular, and unimax functions. By means of this study, the paper also addresses recent issues in Estimation of Distribution Algorithms: (i) the relationship between the complexity of the probabilistic model used by the algorithm and its efficiency, and (ii) the matching between this model and the relationship among the variables of the objective function. After analyzing the results, we conclude that the order of convergence is not related to the complexity of the probabilistic model, and that an algorithm whose probabilistic model mimics the structure of the objective function does not guarantee a low order of convergence.

1 Introduction

The most important questions concerning Evolutionary Algorithms (EAs) are how efficiently an EA will optimize a given objective function and which classes of objective functions can be optimized efficiently. Time complexity is a key issue in the analysis of EAs. It shows how efficiently an algorithm can face a large problem.

Recently, a new kind of EAs called Estimation of Distribution Algorithms (EDAs) [1] has appeared. In recent research on EDAs [2, 3], important questions related to the first hitting time have been raised. Does the first hitting time decrease when the complexity of the probabilistic model used by the algorithm increases? Is the algorithm's capability of detecting and exploiting the (in)dependencies of the objective function related to efficiency? In order to deepen our knowledge on these questions, we offer empirical results of the average first hitting time for some instances of EDAs: UMDA, TREE and EBNA_{BIC} (see Section 2) applied to one example of linear, pseudo-modular and unimax functions. Our aim is to compare them.

The rest of this paper is organized as follows. Section 2 introduces EDAs and the instances that will be used in this work. The functions studied are introduced in Section 3. Section 4 explains the experiments carried out and presents the results. Finally, we draw conclusions in Section 5.

2 Estimation of Distribution Algorithms

Estimation of Distribution Algorithms (EDAs) [1, 4] are based on GAs and constitute an example of stochastic heuristics based on populations of individuals, each of which encodes a possible solution to the optimization problem. These populations evolve in successive generations as the search progresses, organized the same way most EA heuristics are. To generate new individuals EDAs estimate and sample the joint probability distribution of the individuals selected. In Figure 1 a pseudocode for a general EDA can be seen. Unfortunately, the bottleneck of EDAs lies in estimating this joint probability distribution. To avoid this problem, several authors have proposed different algorithms where simplified assumptions concerning the conditional (in)dependencies between the variables of the joint probability distribution are made. The particular algorithms used in this paper have been chosen to provide one instance of each level of complexity of the probabilistic model.

EDA

$D_0 \leftarrow$ Generate M individuals (the initial population) randomly

Repeat for $l = 1, 2, \dots$ until the stopping criterion is met

$D_{l-1}^{Se} \leftarrow$ Select $N \leq M$ individuals from D_{l-1} according to the selection method

$p_l(\mathbf{x}) = p(\mathbf{x}|D_{l-1}^{Se}) \leftarrow$ Estimate the joint probability distribution for an individual to be one of the individuals selected

$D_l \leftarrow$ Sample M individuals (the new population) from $p_l(\mathbf{x})$

Fig. 1. Pseudocode for the EDA approach

UMDA. The Univariate Marginal Distribution Algorithm (UMDA) was proposed by Mühlenbein [4]. UMDA uses the simplest model to estimate the joint probability distribution of the selected individuals at each generation, $p_l(\mathbf{x})$. This joint probability distribution is factorized as a product of independent univariate marginal distributions, which are usually calculated by maximum likelihood estimation.

TREE. Baluja and Davies [5] proposed an algorithm called COMIT. This algorithm uses a probabilistic model that considers second-order statistics. The dependency structure between the variables forms a tree. The tree structure of the probability distribution of the individuals selected at each generation is estimated using the algorithm proposed by Chow and Liu. The parameters, given the structure, are calculated by maximum likelihood estimation. In the original work, COMIT applied a local optimizer to each individual generated. We have eliminated this step in order to carry out a fair comparison. In other words, this paper refers to TREE as an algorithm that does not apply a local optimizer.

EBNA. The Estimation of Bayesian Networks Algorithm (EBNA) was introduced by Larrañaga et al. [6]. This algorithm allows statistics of unrestricted

order in the factorization of the joint probability distribution. This distribution is encoded by a Bayesian network that is learned from the database containing the individuals selected at each generation. Formally, a Bayesian network is a pair, (S, θ) , representing a graphical factorization of a probability distribution. The structure, S , is a directed acyclic graph which reflects the set of conditional (in)dependencies among the variables, while θ is a set of parameters for the local probability distributions associated with each variable.

In EBNA learning the probabilistic model at each generation of the algorithm means learning a Bayesian network from the individuals selected. There are different strategies to learn the structure of a Bayesian network. Here the ‘score + search’ method is used. In this method, given a database, D , and a Bayesian network whose structure is denoted by S , a value which evaluates how well the Bayesian network represents the probability distribution of D is assigned. Different EBNA algorithms can be obtained by using different scores. In this work we have used the BIC score (based on penalized maximum likelihood).

After defining the score, we have to set a search process to find the Bayesian network that maximizes the score given the set of individuals selected. As we need to find an adequate model structure as quickly as possible, a simple algorithm which returns a good structure, even if not optimal, is preferred. An interesting algorithm with these characteristics is Algorithm B. Algorithm B is a greedy search which starts with an arc-less structure and, at each step, adds the arc with the maximum score improvement. The algorithm finishes when there is no arc whose addition improves the score. The parameters of the Bayesian network are calculated by maximum likelihood.

3 The Functions Used

We optimize particular cases of three problem classes of pseudo-Boolean functions: linear, pseudo-modular and unimax functions. To briefly analyze each objective function we set a neighborhood structure: the neighbor solutions $N(\mathbf{x})$ of a given solution $\mathbf{x} \in \{0, 1\}^n$ are composed of \mathbf{x} and all points at Hamming distance 1.

The particular **linear function** analyzed in this work is $f(\mathbf{x}) = \sum_{i=1}^n i \cdot x_i$, $x_i \in \{0, 1\}$. Clearly $(1, \dots, 1)$ is the only global maximum for this function, and the value of $f(\mathbf{x})$ at this point is $\frac{n(n+1)}{2}$. It should be stressed that this function can be optimized variable by variable. Taking this fact into account, a suitable probabilistic model that corresponds to the relationship between the variables of this function assumes that all the variables are independent. Therefore, UMDA seems to be a good candidate to optimize it.

The **pseudo-modular function** that we have used is $f(\mathbf{x}) = \sum_{i=1}^n \prod_{j=1}^i x_j$, with $x_j \in \{0, 1\}$. For this fitness function the only optimal solution is $(1, \dots, 1)$, and the value of $f(\mathbf{x})$ at this point is n . The variables of this function present a chain of pairwise dependencies, where x_i depends on x_{i-1} , $i = 1, \dots, n-1$. This fact suggests that TREE is a suitable model to optimize it.

In the experiments carried out, we have used a well-known **unimax function**: the long path function [7]. It is important to stress that it only makes sense if n is odd. The optimal point in the long path problem is $(1, 1, 0, \dots, 0)$ and the function value is 0. In long path problems, the relationship between the variables of the problem are not evident, and the $EBNA_{BIC}$ algorithm, therefore, seems to be the most adequate one to optimize these functions.

4 Experimental Results

In order to find the average first hitting time of the EDAs introduced in Section 2, we have carried out experiments using those algorithms to maximize the different functions in Section 3. The empirical results for each algorithm and each objective function have been fitted to curves (in the least squares sense) in terms of problem size, using “*Mathematica*” program. We have measure the “goodness” of each fit with the coefficient of determination and the mean squared error. After comparing those quantities for various fits we have chosen the fit with greater coefficient of determination and lower mean squared error. Finally, all this information has been graphically presented.

The population size, M , was fixed to avoid dependency of the first hitting time on this parameter (as done in previously published works [8]), thus we fixed M to the individual size n . The stopping condition is the same for each algorithm: they stop when they find the optimum for the first time. The best individuals are selected (truncation selection).

Next, in order to set up the number of individuals selected and the type of elitism used, we illustrate how a new population is created. Once the population D_{l-1} is created, the $n/2$ best individuals of D_{l-1} are used to estimate the joint probability distribution $p_l(\mathbf{x})$. After that, $n - 1$ individuals are sampled from $p_l(\mathbf{x})$, obtaining D_l^* . Finally the new population D_l is created by selecting the n best individuals from $D_{l-1} \cup D_l^*$. This way, we make sure that the best individual in population D_{l-1} will not get lost. To ensure that the algorithm goes through the optimum, the maximum likelihood estimation of parameters is modified via

EDA

$D_0 \leftarrow$ Generate $M = n$, individuals (initial population) randomly
 Define: $f_{max} = \max \{f(\mathbf{x}) : \mathbf{x} \in \{0, 1\}^n\}$
Repeat for $l = 1, 2, \dots$ until $f_{D_l} = f_{max}$
 $D_{l-1}^{Se} \leftarrow$ Select the $n/2$ best individuals from D_{l-1}
 $p_l(\mathbf{x}) = p(\mathbf{x}|D_{l-1}^{Se}) \leftarrow$ Estimate the joint probability distribution for
 an individual to be one of the individuals selected, using
 Laplace correction when estimating the the parameters
 $D_l^* \leftarrow$ Sample $n - 1$ individuals from $p_l(\mathbf{x})$
 $D_l \leftarrow$ Select the n best individuals from $D_{l-1} \cup D_l^*$
 Define $f_{D_l} = \max \{f(\mathbf{x}) : \mathbf{x} \in D_l\}$

Fig. 2. Pseudocode for the EDA approach used in the experiments

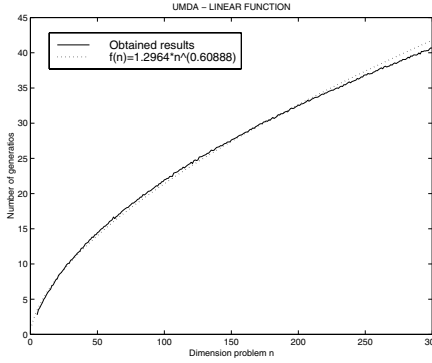


Fig. 3. UMDA - Linear F

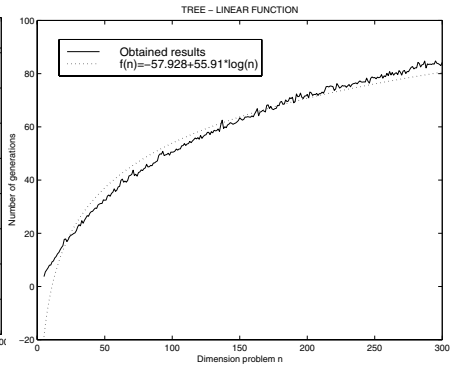


Fig. 4. TREE - Linear F

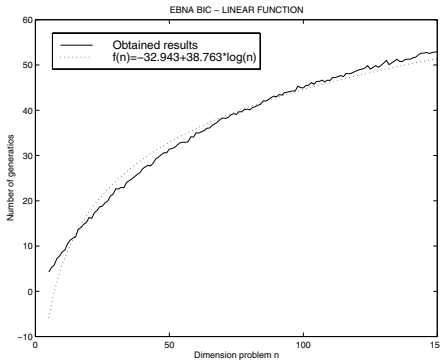


Fig. 5. EBNA_{BIC} - Linear F

	Fitting Curve	Order
UMDA	$1.2964 \cdot n^{0.60888}$	$O(n^\varepsilon)$ $0.6 < \varepsilon < 1$
TREE	$-57.928 + 55.91 \log(n)$	$O(\log n)$
EBNA_{BIC}	$-32.943 + 38.763 \log n$	$O(\log n)$

Fig. 6. Fitting curves for the different EDAs in the optimization of the linear function and their order

Laplace correction. A pseudocode for a general algorithm used in the experiments can be seen in Figure 2.

We run each algorithm 1,000 times for each objective function and each problem dimension, each time recording the generation in which the optimum was reached for the first time. Due to the computational cost associated with the learning of a Bayesian network at each iteration, the problem dimension of the EBNA algorithm is lower than in the rest. The problem dimensions used for the UMDA and TREE algorithms ranged from 5 to 300 in the linear function, from 5 to 150 in the pseudo-modular function, and from 5 to 65 in the unimax function. For EBNA_{BIC} they ranged from 5 to 150 in the linear and pseudo-modular functions and from 5 to 65 in the unimax function.

4.1 Summarizing the Results

After obtaining the results for each algorithm and each objective function, we have found a simple formula that approximates these numerical results, fitting a curve through the data obtained. Furthermore, we can offer a general idea con-

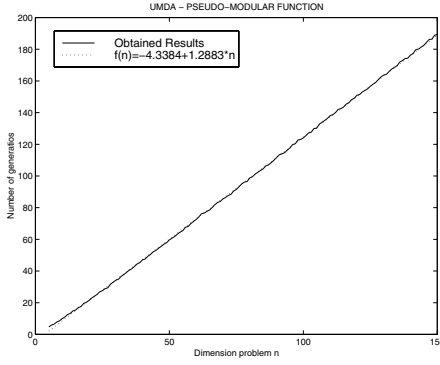


Fig. 7. UMDA - Pse. Mod. F

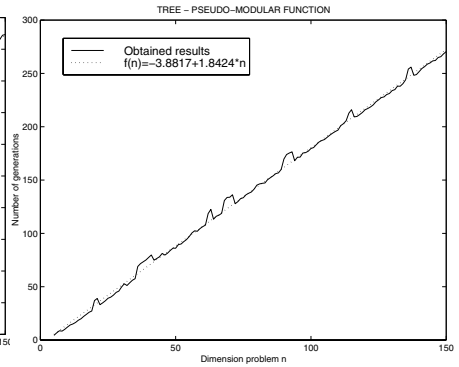
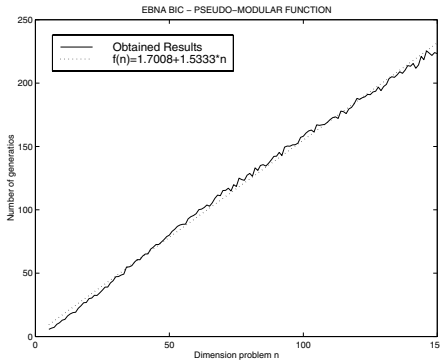


Fig. 8. TREE - Pse. Mod. F

Fig. 9. EBNA_{BIC} - Pse. Mod. F

	Fitting Curve	Order
UMDA	$-4.3384 + 1.2883 \cdot n$	$O(n)$
TREE	$-3.8817 + 1.8424 \cdot n$	$O(n)$
EBNA _{BIC}	$1.7008 + 1.5333 \cdot n$	$O(n)$

Fig. 10. Fitting curves for the different EDAs in the optimization of the pseudo-modular function and their order

cerning the expected time complexity for each algorithm. We show four figures for each function, the last of them being a table. The first three figures show the results obtained for an EDA when optimizing the objective function. The table entries are: (i) the curve that fits the results obtained for each EDA and (ii) the order of the fitting curve.

Linear Function. The results in the linear function can be seen in Figures 3 to 6. They show that UMDA has the best behavior ($O(n^\epsilon)$, $0.6 < \epsilon < 1$). In this case, the probabilistic model of UMDA, that seemed to be the one that best corresponded with the relationship between the variables of the problem, obtains the best average first hitting time.

Pseudo-Modular Function. The results for the pseudo-modular function can be seen in Figures 7 to 10. As can be seen in Figure 10, all the algorithms have the same linear ($O(n)$) behavior. If we take into account the fitting curves, UMDA's results appear slightly better than the others. Here the correspondence between

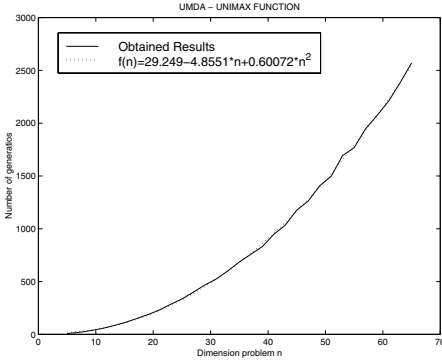


Fig. 11. UMDA - Unimax F

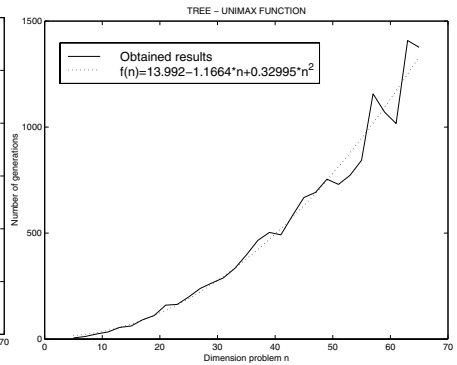


Fig. 12. TREE - Unimax F

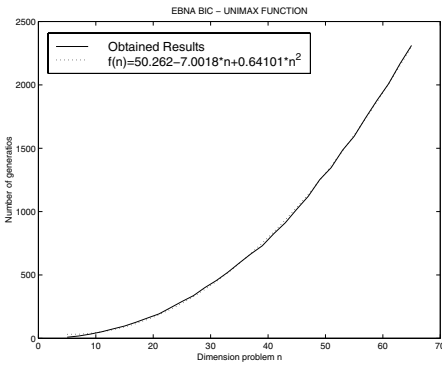


Fig. 13. EBNA_{BIC} - Unimax F

	Fitting Curve	Order
UMDA	$29.249 - 4.8551 \cdot n + 0.67002 \cdot n^2$	$O(n^2)$
TREE	$13.999 - 1.1664 \cdot n + 0.32995 \cdot n^2$	$O(n^2)$
EBNA _{BIC}	$50.262 - 7.0018 \cdot n + 0.64101 \cdot n^2$	$O(n^2)$

Fig. 14. Order of convergence and fitting curves for the different EDAs in the optimization of the unimax function

the probabilistic model used by the algorithm and the relationship between the variables of the problem does not imply a better average first hitting time.

Unimax Function. The results for the unimax function are given in Figures 11 to 14. As can be seen in Figure 14 all the algorithms have the same quadratic behavior ($O(n^2)$). Taking into account the fitting curves, TREE seems to have better behavior than the others. Again, the correspondence between the probabilistic model used by the algorithm and the relationship between the variables of the problem does not imply a better average first hitting time.

5 Conclusion

Based on empirical results, this work offers new information on the first hitting time of some EDAs applied to a number of objective functions. Two main conclusions have been reached. On one hand, the average optimization time in the experiments is not related to the complexity of the probabilistic model used by

the algorithm. A greater complexity of the probabilistic model of the EDA does not imply a greater efficiency or a low order of the first hitting time. On the other hand, EDAs whose probabilistic models reflect the relationship between the variables of the problem do not obtain a better order of the first hitting time.

In order to improve the statistical analysis carried out, in future research we will (i) run the algorithm more than 1,000 times, (ii) record at each generation the standard deviation and (iii) enlarge the range of n (to provide a more sensible fit).

Acknowledgments

This work was supported by the University of the Basque Country under grant 9/UPV/EHU 00140.226-15334/2003, and also by the Basque Government under grant Saiotek S-PE04UN25.

References

1. Larrañaga, P., Lozano, J.A.: *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Kluwer Academic Publishers (2002)
2. Pelikan, M., Sastry, K., Goldberg, D.E.: Scalability of the Bayesian Optimization Algorithm. *International Journal of Approximate Reasoning* **31** (2002) 221–258
3. Mühlenbein, H., Mahng, T.: Evolutionary Optimization and the Estimation of Search Distributions with Applications to Graph Bipartitioning. *International Journal of Approximate Reasoning* **31** (2002) 157–192
4. Mühlenbein, H., Paaß, G.: From Recombination of Genes to the Estimation of Distributions I. Binary Parameters. In Voigt, H.M., Ebeling, W., Rechenberg, I., Schwefel, H.P., eds.: *Parallel Problem Solving from Nature, PPSN-IV*. (1996) 178–187
5. Baluja, S., Davies, S.: *Using Optimal Dependency Trees for Combinatorial Optimization: Learning the Structure of the Search Space*. Technical Report CMU-CS-97-107, Carnegie Mellon University (1997)
6. Larrañaga, P., Etxeberria, R., Lozano, J.A., Peña, J.M.: Combinatorial Optimization by Learning and Simulation of Bayesian Networks. In Boutilier, C., Goldszmidt, M., eds.: *Proceedings of Uncertainty in Artificial Intelligence, UAI-2000*, Morgan Kaufmann (2000) 343–352
7. Horn, J., Goldberg, D.E., Deb, K.: Long Path Problems. In Davidor, Y., Schwefel, H.P., Männer, R., eds.: *Parallel Problem Solving from Nature, PPSN III*, Berlin and Heidelberg: Springer (1994) 149–158
8. He, J., Yao, X.: Drift Analysis and Average Time Complexity of Evolutionary Algorithms. *Artificial Intelligence* **127** (2001) 57–85

A Comparison of Evolutionary Approaches to the Shortest Common Supersequence Problem

Carlos Cotta

Dept. Lenguajes y Ciencias de la Computación, ETSI Informática,
University of Málaga, Campus de Teatinos, 29071 - Málaga, Spain
ccottap@lcc.uma.es

Abstract. The Shortest Common Supersequence problem is a hard combinatorial optimization problem with numerous practical applications. Several evolutionary approaches are proposed for this problem, considering the utilization of penalty functions, GRASP-based decoders, or repairing mechanisms. An empirical comparison is conducted, using an extensive benchmark comprising problem instances of different size and structure. The empirical results indicate that there is no single best approach, and that the size of the alphabet, and the structure of strings are crucial factors for determining performance. Nevertheless, the repair-based EA seems to provide the best performance tradeoff.

1 Introduction

The Shortest Common Supersequence (SCS) problem is a classical problem from the realm of string analysis. Roughly speaking, the SCS problem amounts to finding a minimal-length sequence S of symbols such that every string in a certain set L can be generated from S by removing some symbols of the latter. The resulting combinatorial problem is enormously interesting, not only from the point of view of Theoretical Computer Science, but also from an applied perspective. Indeed, it has applications in planning, data compression, and bioinformatics among other fields [1, 2, 3].

Unfortunately, the SCS problem has been shown to be hard under various formulation and restrictions [4, 5, 6] (a summary of these hardness results is provided in Sect. 2). This way, although exact approaches have been proposed to tackle this problem (see e.g., [7]), these are impractical for even moderate-size problem instances. Hence, heuristic approaches are in order. In this sense, greedy approaches have been popular. For example, one can cite the Majority Merge (MM) algorithm, and related variants [8]. However, these heuristics are not the ultimate solvers for this problem due to their myopic functioning. More sophisticated techniques such as evolutionary algorithms (EAs) can be used to overcome the limitations of greedy techniques.

This work will analyze and compare four different evolutionary approaches to the SCS problem, involving either a direct search in the space of supersequences, or using auxiliary search spaces and more complex decoding mechanisms for obtaining high-quality solutions.

2 The Shortest Common Supersequence Problem

Let us start by formally defining the SCS problem in its decisional version:

SHORTEST COMMON SUPERSEQUENCE PROBLEM

Instance: A set L of m strings $\{s_1, \dots, s_m\}$ of arbitrary length over an alphabet Σ (i.e., $s_i \in \Sigma^*$, for $1 \leq i \leq m$), and a positive integer k .

Question: Does there exist a string $s \in \Sigma^*$, $|s| \leq k$, such that s is a supersequence¹ of every $s_i \in L$?

Having defined the problem, let us now consider its computational complexity.

2.1 Complexity Results for the SCS Problem

The SCS problem can be shown to be NP-hard, even if strong constraints are posed on L , or on Σ . For example, it is NP-hard in general when all s_i have length two [2], or when the alphabet size $|\Sigma|$ is two [5]. It must be noted that—despite being important—NP-hard results are usually over-stressed; in fact, there are many problems that can be efficiently solved in practice, yet they are NP-hard.

Parameterized complexity [9] tries to deal with this issue, providing a more sensible characterization of hardness. The key idea is to isolate hardness (i.e., non-polynomial behavior) within a certain set of parameters. This way, if these parameters are kept fixed, the problem can be efficiently² solved for large problem sizes. VERTEX COVER is a good example of this situation: it is NP-hard, but it can be solved in linear time in the number of vertices, when the size of the vertex cover sought is kept fixed. Problems such as VERTEX COVER for which this hardness-isolation is possible are termed *fixed-parameter tractable* (FPT). Non-FPT problem will fall under some class in the W -hierarchy. Hardness for the parameterized class $W[1]$ is the current measure of intractability.

Several parameterizations are possible for the SCS problem. Firstly, the maximum length k of the supersequence sought can be taken as a parameter. If the alphabet size is constant, or another parameter, then the problem turns in this case to be FPT, since there are at most $|\Sigma|^k$ supersequences. However, this is not very useful in practice because $k \geq \max |s_i|$. If the number of strings m is used as parameter, then SCS is $W[1]$ -hard, and remains so even if $|\Sigma|$ is taken as another parameter [3], or is constant [6]. Failure of finding FPT results in this latter scenario is particularly relevant since the alphabet size in biological problems is fixed (e.g., there are just four nucleotides in DNA). Furthermore, notice that absence of FPT algorithms implies the non-existence of fully polynomial time approximation schemes (FPTAS) for the corresponding problem.

¹ s is a supersequence of r if $r = \epsilon$ (where ϵ is the empty string) or if $s = \alpha s'$, $r = \alpha' r'$ and s' is a supersequence of r' ($\alpha = \alpha'$) or s' is a supersequence of r ($\alpha \neq \alpha'$).

² In time $O(f(k)n^c)$, where k is the parameter, n is the problem size, f is an arbitrary function of k only, and c is a constant independent of k and n .

2.2 Heuristics for the SCS Problem

The hardness results mentioned in the previous subsection motivate the utilization of heuristic approaches for tackling the SCSP. One of the most popular algorithms for this purpose is Majority Merge (MM). This is a greedy algorithm that constructs a supersequence incrementally by adding the symbol most frequently found at the front of strings in L , and removing these symbols from the corresponding strings. More precisely:

Heuristic MM ($L = \{s_1 \cdots, s_m\}$)

1. **let** $s \leftarrow \epsilon$
2. **do**
 - (a) **for** $\alpha \in \Sigma$ **do let** $\nu(\alpha) \leftarrow \sum_{s_i = \alpha s'_i} 1$
 - (b) **let** $\beta \leftarrow \max^{-1}\{\nu(\alpha) \mid \alpha \in \Sigma\}$
 - (c) **for** $s_i \in L, s_i = \beta s'_i$ **do let** $s_i \leftarrow s'_i$
 - (d) **let** $s \leftarrow s\beta$
- until** $\sum_{s_i \in L} |s_i| = 0$

The myopic functioning of MM makes it incapable of grasping the global structure of strings in L though. In particular, MM misses the fact that the strings can have different lengths [8]. This implies that symbols at the front of short strings will have more chances to be removed, since the algorithm has still to scan the longer strings. For this reason, it is less urgent to remove those symbols. In other words, it is better to concentrate in shortening longer strings first. This can be done by assigning a weight to each symbol, depending of the length of the string in whose front is located. Branke *et al.* [8] propose to use precisely this string length as weight, i.e., step 2a in the previous pseudocode would be modified to have $\nu(\alpha) \leftarrow \sum_{s_i = \alpha s'_i} |s'_i|$. This modified heuristic will be termed weighted MM (WMM).

Several other heuristics were also defined in [8] on the basis of WMM. For example, one of them has $\nu(\alpha) \leftarrow |\text{WMM}(\{s_1|_\alpha, \dots, s_m|_\alpha\})|$ (to be minimized), where $s|_\alpha$ is the string obtained by removing α from the front of s ; ties are broken by maximizing $|\text{WMM}(\{s_i \mid s_i = \alpha s'_i\})|$. A more interesting heuristic results from the combination of EAs and WMM. In this heuristic, the EA is used to evolve weights for each character of every string. These weights are utilized within step 2a, modifying the influence of symbols in each string. This is done by multiplying the WMM weight with the evolved weight, i.e., $\nu(\alpha) \leftarrow \sum_{s_i = \alpha s'_i} w_{p_{\alpha i}, i} |s'_i|$, where $p_{\alpha i}$ is the position of the current front symbol α in the original string s_i . In a further refinement, the EA is used also to evolve a basic value to be added to each weight before evaluation.

This EA approach is similar to the EA used in [10] for the multidimensional knapsack problem, in which a greedy heuristic was used to generate solutions, and weights were evolved in order to modify the value of objects (thus making the underlying heuristic take different decisions). Next section will explore alternative EA definitions in which the search is conducted in search spaces different to this weight space.

3 Evolutionary Approaches to the SCS Problem

Clearly, one of the difficulties faced by an EA when applied to the SCS problem is the existence of feasibility constraints, i.e., an arbitrary string $s \in \Sigma^*$, no matter its length, is not necessarily a supersequence of strings in L . Typically, these situations can be solved in three ways: (i) allowing the generation of infeasible solutions and penalizing accordingly, (ii) using a repairing mechanism for mapping infeasible solutions to feasible solutions, and (iii) defining appropriate operators and/or problem representation to avoid the generation of infeasible solutions. These three possibilities will be explored below.

Let us firstly consider the simplest version, namely the use of a penalty function. The idea is to have an EA evolving strings in Σ^* , using the supersequence length as the quality measure for feasible solutions, and adding an extra penalty term for infeasible solutions. This has been implemented as follows: let s be the tentative solution provided by the EA, and let $L = \{s_1, \dots, s_m\}$ be the target strings; then, the fitness (to be minimized) is

$$fitness(s, L) = \begin{cases} 0 & \text{if } \forall i : s_i = \epsilon \\ 1 + fitness(s', L|_\alpha) & \text{if } \exists i : s_i \neq \epsilon \text{ and } s = \alpha s' \\ |MM(L)| & \text{if } \exists i : s_i \neq \epsilon \text{ and } s = \epsilon \end{cases} \quad (1)$$

This algorithm will be termed Penalty EA, and relies in MM for providing a heuristic assessment on how much longer a string should be to account for uncovered string suffixes in L . An obvious variant consists of injecting the actual sequence returned by MM back to the candidate solution. Thus, the sequence is repaired, resulting in a feasible solution. More precisely, the repairing function $\rho : \Sigma^* \times (\Sigma^*)^m \rightarrow \Sigma^*$ can be described as follows:

$$\rho(s, L) = \begin{cases} s & \text{if } \forall i : s_i = \epsilon \\ \rho(s', L) & \text{if } \exists i : s_i \neq \epsilon \text{ and } \nexists i : s_i = \alpha s'_i \text{ and } s = \alpha s' \\ \alpha \rho(s', L|_\alpha) & \text{if } \exists i : s_i = \alpha s'_i \text{ and } s = \alpha s' \\ MM(L) & \text{if } \exists i : s_i \neq \epsilon \text{ and } s = \epsilon \end{cases} \quad (2)$$

As it can be seen, this repairing function not only completes s in order to have a valid supersequence, but also removes intermediate symbols that are not present at the front of any string at a certain step. Thus, it also serves the purpose of local improver. This algorithm is termed Repair EA.

The third approach is to have the EA handling uniquely feasible solutions. In this case, we have considered the utilization of an auxiliary space, and a *smart* decoder in order to perform the mapping to the sequence space. This decoder can be based on either MM or WMM, and borrows some ideas from greedy randomized adaptive search procedures (GRASP). This latter metaheuristic also relies on an underlying greedy heuristic, and proceeds iteratively by selecting at each step an attribute of the solution from a *candidate list* [11]. This selection can be typically done by using a qualitative criterion (i.e., a candidate is selected among the best k elements in the candidate list, k being a parameter), or a quantitative criterion (i.e., a candidate is selected among the elements whose quality is within a certain range).

One of the potential problems of the basic GRASP procedure described before relies on the selection of the parameter for selecting an attribute value from the candidate list. As shown in [12], using a single fixed value for this parameter may hinder finding high-quality solutions. Several options are possible to solve this problem. On one hand, a learning-based strategy termed *reactive* GRASP was proposed [13]. On the other hand, the utilization of EAs to evolve the sequence of selection decisions was presented in [14]. This latter approach is precisely considered here. To be precise, the EA is used to evolve a sequence of integers $\langle \delta_1, \dots, \delta_n \rangle$, $\delta_i \in [1..|\Sigma|]$, where $n = \sum_{i=1}^m |s_i|$. At each step of the decoding process, a ranked list of the potential symbols to be added to the supersequence is constructed using either the MM or the WMM criterion; the value δ_i indicates that the δ_i -th best symbol is chosen at the i -th step. Notice that the construction of the supersequence will in general be accomplished in many less steps than n .

4 Experimental Validation

The experiments have been done with a steady-state EA ($popsiz = 100$, $p_X = .9$, $p_m = 1/n$, $maxevals = 100,000$), using binary tournament selection, uniform crossover, and random-substitution mutation. Three different sets of problem instances have been considered in the experimentation. The first one is composed of random strings with different lengths. To be precise, each instance is composed of eight strings, four of them with 40 symbols, and the remaining four with 80 symbols. Each of these strings is randomly built, using an alphabet Σ . Four subsets of instances have been defined using different alphabet sizes, namely $|\Sigma| = 2, 4, 8$, and 16. For each alphabet size, five different instances have been generated.

The second set of instances comprises strings whose structure is deceptive for greedy heuristics such as MM or WMM. These are the following (cf. [8]):

- $L_1 = \{9 \times a^{40}, 4 \times ba^{39}, 2 \times bba^{38}, 1 \times bbba^{37}\}$. The optimal solution for this instance is to firstly remove the three b 's, and then remove the forty a 's. It thus has length 43.
- $L_2 = \{9 \times a^{40}, 4 \times b^{13}a^{27}, 2 \times b^{26}a^{14}, 1 \times b^{39}a\}$. This instance is similar to L_1 , and again the optimal solution is to firstly remove the b 's, i.e., $b^{39}a^{40}$, a solution of length 79.
- $L_3 = \{8 \times a^{20}b^{20}, 8 \times b^{20}c^{20}\}$. The optimal solution in this case is $a^{20}b^{20}c^{20}$.

Finally, a more realistic benchmark consisting of strings with a common origin has been considered. A DNA sequence from a SARS coronavirus strain has been retrieved from a genomic database³, and has been taken as supersequence; then, different sequences are obtained from this supersequence by scanning it from left to right, and skipping nucleotides with a certain fixed probability. In these experiments, the length of the supersequence is 158, the gap probability is 10%, 15%, or 20% and the number of so-generated sequences is 10.

³ <http://gel.ym.edu.tw/sars/genomes.html>, accession AY271716.

Table 1. Results of the different heuristics on 8 random strings (4 of length 40, and 4 of length 80), for different alphabet sizes $|\Sigma|$. The results of MM and WMM are averaged over 150 executions, and the results of the EAs are averaged over 30 runs. In all cases, the results are further averaged over five different problem instances

$ \Sigma $	MM		WMM		Penalty EA	
	best	mean \pm std.dev.	best	mean \pm std.dev.	best	mean \pm std.dev.
2	115.0	120.46 \pm 2.16	114.8	116.13 \pm 0.89	119.0	123.61 \pm 2.52
4	164.2	175.47 \pm 5.02	157.8	161.85 \pm 2.81	200.2	223.31 \pm 12.18
8	227.0	249.33 \pm 6.92	210.4	219.61 \pm 4.90	366.8	445.94 \pm 23.48
16	309.2	333.57 \pm 9.08	282.8	296.07 \pm 5.31	538.2	569.74 \pm 11.73
$ \Sigma $	Repair EA		GRASP-EA(MM)		GRASP-EA(WMM)	
	best	mean \pm std.dev.	best	mean \pm std.dev.	best	mean \pm std.dev.
2	111.2	112.58 \pm 0.75	113.0	116.95 \pm 1.97	110.8	113.31 \pm 1.49
4	151.6	155.17 \pm 1.85	160.6	167.47 \pm 3.11	151.6	157.81 \pm 2.98
8	205.4	213.47 \pm 3.97	217.6	228.18 \pm 4.22	204.0	211.01 \pm 3.05
16	267.0	281.81 \pm 5.88	286.2	297.53 \pm 4.94	271.2	278.15 \pm 2.97

Table 2. Results of the different heuristics on the deceptive problem instances. The results of MM and WMM are averaged over 150 executions, and the results of the EAs are averaged over 30 runs

	MM		WMM		Penalty EA	
	best	mean \pm std.dev.	best	mean \pm std.dev.	best	mean \pm std.dev.
L_1	157	157.00 \pm 0.00	92	92.00 \pm 0.00	43	43.00 \pm 0.00
L_2	121	121.00 \pm 0.00	91	91.00 \pm 0.00	79	79.00 \pm 0.00
L_3	68	77.73 \pm 2.98	79	79.46 \pm 0.50	60	60.07 \pm 6.42
	Repair EA		GRASP-EA(MM)		GRASP-EA(WMM)	
	best	mean \pm std.dev.	best	mean \pm std.dev.	best	mean \pm std.dev.
L_1	43	43.00 \pm 0.00	43	43.00 \pm 0.00	43	43.00 \pm 0.00
L_2	79	79.00 \pm 0.00	79	79.00 \pm 0.00	79	79.00 \pm 0.00
L_3	60	60.00 \pm 0.00	60	60.00 \pm 0.00	60	60.00 \pm 0.00

First of all, the results for random strings are shown in Table 1. WMM performs better than MM, and both the repair-based EA and the WMM-based EA outperform the remaining algorithms. The MM-based EA is capable of beating MM, and produce similar results to those of WMM; however, it cannot compete with the repair-based EA or the WMM-based EA. Notice also the poor results of the plain Direct EA (it cannot even produce a feasible solution for $|\Sigma| = 16$).

It is interesting to note the U-shaped behavioral pattern of the EAs with respect to the basic heuristics. This is illustrated in Fig. 1, that shows the ratio between the percentage of improvement of the EAs (with respect to MM), and the percentage of improvement of WMM (also with respect to MM). The pattern for the MM-based EA and the WMM-based EA is the same (save for the scale factor) with a minimum at $|\Sigma| = 4$; however, this pattern is shifted for the

Table 3. Results of the different heuristics on the strings from the SARS DNA sequence for different gap probabilities. The results of MM and WMM are averaged over 150 executions, and the results of the EAs are averaged over 30 runs

gap%	MM		WMM		Penalty EA	
	best	mean \pm std.dev.	best	mean \pm std.dev.	best	mean \pm std.dev.
10%	158	158.00 \pm 0.00	158	158.00 \pm 0.00	350	428.00 \pm 26.83
15%	158	158.00 \pm 0.00	158	158.00 \pm 0.00	358	397.87 \pm 20.86
20%	160	227.77 \pm 25.75	231	234.41 \pm 1.88	339	377.30 \pm 18.87
gap%	Repair EA		GRASP-EA(MM)		GRASP-EA(WMM)	
	best	mean \pm std.dev.	best	mean \pm std.dev.	best	mean \pm std.dev.
10%	158	158.00 \pm 0.00	163	181.60 \pm 14.55	163	171.97 \pm 8.42
15%	158	158.00 \pm 0.00	199	218.70 \pm 8.83	191	212.60 \pm 9.04
20%	165	180.80 \pm 15.73	205	222.47 \pm 8.88	212	222.27 \pm 5.37

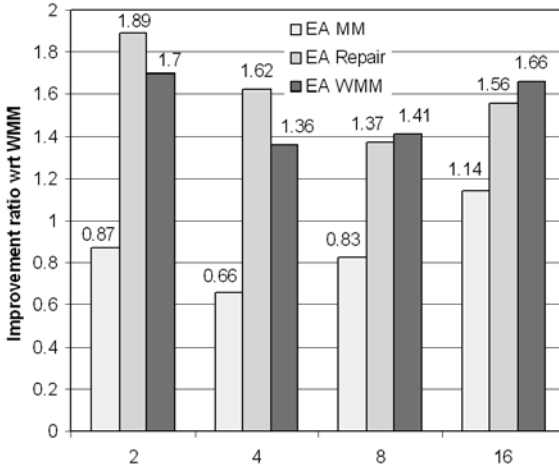


Fig. 1. Improvement ratios of the different EAs with respect to WMM for different alphabet sizes

repair-based EA that has its minimum at $|\Sigma| = 8$. At any rate, the performance of the repair-based EA is here very similar to that of the WMM-based EA.

The results for the deceptive problem instances are shown in Table 2. As expected, both MM and WMM are fooled by the problem structure. However, the EAs are capable of finding consistently the optimal solutions without major difficulties. This emphasizes the important role played by the evolutionary search in this problem.

Finally, the results for the strings from the SARS DNA sequence are shown in Table 3. The basic heuristics perform quite well for low gap probability (i.e., for larger strings). However, neither the MM-based EA nor the WMM-based EA can match this performance (recall that $|\Sigma| = 4$, that is, the lower performance point for these EAs). For the larger gap probability there is a remarkable performance

drop of the basic heuristics, and GRASP-based EAs can catch up with these. The repair-based EA offers the best results throughout the three problem instances, performing significantly better than the remaining algorithms.

5 Conclusions

Four different EAs have been proposed and compared for the SCS problem. The main goal has been to determine which of the typical constraint-handling procedures is more appropriate for this problem. The experimental results seem to indicate that the best performance tradeoff is provided by the repair-based EA: it behaves much better than the other EAs in some problem instances, and similarly to these in the remaining ones. The GRASP-based EAs are located at the next performance level, the WMM-based EA performing better than the MM-based EA. Future work will be directed to test other underlying heuristics for decoding solutions. The repair-based EA seems to be the adversary with which such new EAs should be confronted.

Acknowledgements. This work is partially supported by Spanish MCyT and FEDER under contract TIC2002-04498-C05-02.

References

1. Foulser, D., Li, M., Yang, Q.: Theory and algorithms for plan merging. *Artificial Intelligence* **57** (1992) 143–181
2. Timkovsky, V.: Complexity of common subsequence and supersequence problems and related problems. *Cybernetics* **25** (1990) 565–580
3. Hallet, M.: An integrated complexity analysis of problems from computational biology. PhD thesis, University of Victoria (1996)
4. Bodlaender, H., Downey, R., Fellows, M., Wareham, H.: The parameterized complexity of sequence alignment and consensus. *Theoretical Computer Science* **147** (1994) 31–54
5. Middendorf, M.: More on the complexity of common superstring and supersequence problems. *Theoretical Computer Science* **125** (1994) 205–228
6. Pietrzak, K.: On the parameterized complexity of the fixed alphabet shortest common supersequence and longest common subsequence problems. *Journal of Computer and System Sciences* **67** (2003) 757–771
7. Fraser, C.: Subsequences and Supersequences. PhD thesis, University of Glasgow, Department of Computer Science (1995)
8. Branke, J., Middendorf, M., Schneider, F.: Improved heuristics and a genetic algorithm for finding short supersequences. *OR-Spektrum* **20** (1998) 39–45
9. Downey, R., Fellows, M.: *Parameterized Complexity*. Springer-Verlag (1998)
10. Cotta, C., Troya, J.: A hybrid genetic algorithm for the 0-1 multiple knapsack problem. In Smith, G., Steele, N., Albrecht, R., eds.: *Artificial Neural Nets and Genetic Algorithms*, Wien New York, Springer-Verlag (1998) 251–255
11. Feo, T., Resende, M.: Greedy randomized adaptive search procedures. *Journal of Global Optimization* **6** (1995) 109–133

12. Prais, M., Ribeiro, C.: Parameter variation in GRASP procedures. *Investigación Operativa* **9** (2000) 1–20
13. Prais, M., Ribeiro, C.: Reactive GRASP: an application to a matrix decomposition problem in TDMA traffic assignment. *INFORMS Journal on Computing* **12** (2000) 164–176
14. Cotta, C., Fernández, A.: A hybrid GRASP-evolutionary algorithm approach to golomb ruler search. In Yao, X., et al., eds.: *Parallel Problem Solving From Nature VIII*. Volume 3242 of *Lecture Notes in Computer Science.*, Berlin, Springer-Verlag (2004) 481–490

Simultaneous Evolution of Neural Network Topologies and Weights for Classification and Regression^{*}

Miguel Rocha¹, Paulo Cortez², and José Neves¹

¹ Dep. Informática, Universidade do Minho, 4710-057 Braga, Portugal
{mrocha, jneves}@di.uminho.pt

² Dep. Sistemas de Informação, Univ. do Minho, 4800-058 Guimarães, Portugal
pcortez@dsi.uminho.pt
<http://www.dsi.uminho.pt/~pcortez>

Abstract. *Artificial Neural Networks (ANNs)* are important *Data Mining (DM)* techniques. Yet, the search for the optimal *ANN* is a challenging task: the architecture should learn the input-output mapping without overfitting the data and training algorithms tend to get trapped into local minima. Under this scenario, the use of *Evolutionary Computation (EC)* is a promising alternative for *ANN* design and training. Moreover, since *EC* methods keep a pool of solutions, an *ensemble* can be build by combining the best *ANNs*. This work presents a novel algorithm for the optimization of *ANNs*, using a direct representation, a structural mutation operator and Lamarckian evolution. Sixteen real-world classification/regression tasks were used to test this strategy with single and ensemble based versions. Competitive results were achieved when compared with a heuristic model selection and other *DM* algorithms.

Keywords: Supervised Learning, Multilayer Perceptrons, Evolutionary Algorithms, Ensembles.

1 Introduction

Artificial Neural Networks (ANNs) denote a set of connectionist models inspired in the behavior of the human brain. In particular, the *Multilayer Perceptron (MLP)* is the most popular *ANN* architecture, where *neurons* are grouped in *layers* and only *forward connections* exist [3]. This provides a powerful base-learner, with advantages such as nonlinear mapping and noise tolerance, increasingly used in the *Data Mining (DM)* and *Machine Learning (ML)* fields due to its good behavior in terms of predictive knowledge [8].

The interest in *MLPs* was stimulated by the advent of the *Backpropagation* algorithm in 1986 and since then, several fast variants have been proposed (e.g., *RPROP*) [9]. Yet, these training algorithms minimize an error function by tuning the modifiable parameters of a fixed architecture, which needs to be set a

^{*} This work was supported by the ALGORITMI research center and the *FCT* project POSI/EIA/59899/2004.

priori. The *MLP* performance will be sensitive to this choice: a small network will provide limited learning capabilities, while a large one will induce generalization loss (i.e., overfitting). Thus, the correct design of the *MLP* topology is a complex and crucial task, commonly addressed by trial-and-error procedures (e.g. exploring different number of hidden nodes), in a *blind* search strategy, which only goes through a small set of possible configurations. More elaborated methods have also been proposed, such as *pruning* [14] and *constructive* [5] algorithms, although these perform *hill-climbing*, being prone to local minima. In addition, the gradient-based procedures used for the *MLP* training are not free from getting trapped into local minima when the error surface is rugged, being also sensitive to parameter settings and to the network initial weights.

An alternative is to optimize both the structure and weights by using *Evolutionary Computation (EC)*, which performs a global multi-point (or *beam*) search, quickly locating areas of high quality, even when the search space is very complex. The combination of *EC* and *ANNs*, called *Evolutionary Neural Networks (ENNs)*, is a suitable candidate for topology design, due to the error surface features [16]: the number of nodes/connections is unbounded; the mapping from the structure to its performance is indirect; changes are discrete; and similar topologies may present different performances. Moreover, since *EC* performs a global search, it is expected to overcome local minima and reach the optimal set of weights. *Ensembles* are another promising *DM/ML* research field, where several models are combined to produce an answer [2]. Often, it is possible to build ensembles that are better than individual learners. One interesting way to build *ANN* ensembles is based on the use of *heterogeneous topologies*, where a family of *MLPs* with distinct structures (and therefore complexities) are combined [11]. Since *ENNs* use a population of different neural structures, this strategy can be easily adapted to *ENNs* with no computational effort increase.

In this work, a novel *ENN* is presented for the simultaneous optimization of *MLPs*, where a direct representation is used. New topologies are achieved by applying a structural mutation, which adds or deletes connections or weights. On the other hand, connection weights are optimized through Lamarckian evolution that uses a random mutation and a local learning algorithm (RPROP). This technique will be tested in classification and regression problems, using both single *ANN* and ensemble based models. Then, the results will be compared with a heuristic *ANN* selection procedure, as well with other *DM/ML* methods.

The paper is organized as follows. First, a description is given on the datasets used (Section 2.1). Then, the neural and evolutionary models are presented (Sections 2.2 and 2.3). In Section 3 the experiments performed are described and the results analyzed. Finally, closing conclusions are drawn in Section 4.

2 Materials and Methods

2.1 Classification and Regression Datasets

This work endorses two important *DM/ML* problems: *classification* and *regression* tasks. The former requires a correct association between input attributes

Table 1. A summary of the data sets used

Task	Inputs			Examples Classes	
	Num.	Bin.	Nom.		
Balance	4	0	0	625	3
Bupa	6	0	0	345	2
Car	0	0	6	1728	4
Cmc	5	3	1	1473	3
Dermatology	34	0	0	366	6
Ionosphere	34	0	0	351	2
Sonar	60	0	0	104	2
Yeast	7	1	0	1484	10
Abalone	7	0	1	4177	\Re
Auto-mpg	5	0	2	398	\Re
Autos	17	3	5	205	\Re
Breast-cancer	1	4	4	286	\Re
Heart-disease	6	3	4	303	\Re
Housing	12	1	0	506	\Re
Servo	2	0	2	167	\Re
WPBC	32	0	0	194	\Re

and a class label (e.g., classifying cells for cancer diagnosis). The latter deals with a functional approximation between n -dimensional input vectors and m -dimensional output ones (e.g., stock market prediction).

Eight classification and eight regression datasets were selected from the UCI *ML* repository [13]. The main features are listed in Table 1, namely: the number of numeric (**Num.**), binary (**Bin.**) and nominal (**Nom.**, i.e. discrete with 3 or more labels) input attributes, as well as the number of examples and classes. The regression tasks are identified by the symbol \Re (last eight rows).

2.2 Neural Networks

The *MLPs* used in this study make use of biases, sigmoid activation functions and one hidden layer with a variable number of nodes. A different approach was followed for the regression tasks, since outputs may lie out of the logistic output range $([0, 1])$. Hence, shortcut connections and *linear* functions were applied on the output neuron(s), to scale the range of the outputs (Fig. 1).

Before feeding the *MLPs*, the data was preprocessed with a *1-of-C* encoding, one binary variable per class, applied to the nominal attributes and all inputs were rescaled within the range $[-1, 1]$. For example, the **safety** attribute from the task **car** was encoded as: *low* \rightarrow (1 -1 -1), *med* \rightarrow (-1 1 -1) and *high* \rightarrow (-1 -1 1). Regarding the outputs, the discrete variables were normalized within the range $[0, 1]$ (using also a *1-of-C* encoding for the nominal attributes). Therefore, the predicted class is given by the nearest class value to the node’s output, if one single node is used (binary variable), otherwise the node with the highest output value is considered. On the other hand, regression problems will be modeled by one real-valued output, which directly represents the *dependent* target variable.

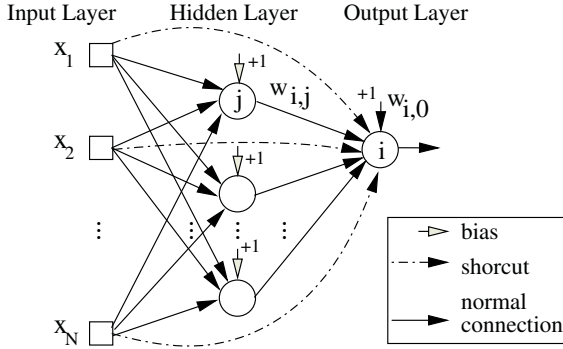


Fig. 1. A fully connected *Multilayer Perceptron* with one output neuron, bias and shortcuts

Two distinct accuracy measures were adopted: the *Percentage of Correctly Classified Examples (PCCE)*, used in classification tasks; and the *Normalized Root Mean Squared Error (NRMSE)*, applied in the regression ones. These measures are given by the equations:

$$\begin{aligned}
 PCCE &= \sum_{i=1}^E 1, \text{ if } (T_i = P_i) / E \times 100 (\%) \\
 RMSE &= \sqrt{\sum_{i=1}^E (T_i - P_i)^2 / E} \\
 NRMSE &= \frac{RMSE}{\sum_{i=1}^E T_i / E} \times 100 (\%)
 \end{aligned} \tag{1}$$

where E denotes the number of examples; P_i , T_i the predicted and target values for the i -th example.

In order to provide a basis for comparison with the *ENN*, an *Heuristic* approach (*HNN*) to model selection was defined by a simple trial-and-error procedure, where fully connected *MLPs*, with a number of hidden nodes ranging from 0 to 20, are trained. For each *MLP*, the initial weights were randomly set within the range $[-1, 1]$. Next, the *RPROP* algorithm [9] was selected for training, due to its faster convergence and stability, being stopped after a maximum of 500 epochs or when the error slope was approaching zero. Then, the topology with the lowest validation error (computed over non training data) is selected. The trained *MLPs* will also be used to build an *Ensemble (HNNE)*, where the output is given by the average over all 21 *MLPs*.

2.3 Evolutionary Neural Network

In the past, evolutionary approaches have been proposed for training connection weights, optimizing neural topologies and evolving both architectures and weights [16]. Yet, in order to train an *ANN*, an a priori architecture needs to be set. On the other hand, evolving neural structures without weight information will make harder the fitness evaluation due to the noisy fitness evaluation problem: different random initial weights may produce distinct performances. Hence,

it seems natural to use the global search advantages of the *EC* to simultaneously evolve topologies and weights [17].

In the present work, a *Simultaneous Evolutionary Neural Network (SENN)* algorithm with a *direct* representation is embraced, where the genotype is the whole *MLP*. The population size contains P individuals and the initial population is created by choosing structures with a random number of hidden nodes (between 0 and H). Then, each possible connection is set with a probability of 50%. Next, the connection weights are randomly initialized within the range $[-1.0; 1.0]$. Regarding the genetic recombination, the crossover operator was discarded since previous experiments [10] revealed no gain in its use, probably due to the *permutation* problem; i.e., several genomes may encode the same *ANN*. Thus, the evolutionary algorithm uses two different mutation operators (Fig. 2) with equal probabilities (50%): a *structural mutation* [12], which works by adding/deleting a random number (from 1 to M) of nodes or connections; and a *macro mutation*, which replaces a random number of weights (from 1 to M) by a new randomly generated value within the range $[-1.0, 1.0]$.

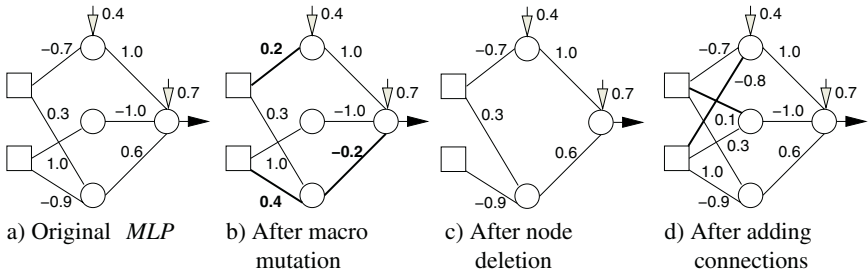


Fig. 2. Example of the application of the mutation operators

This algorithm will also be combined with a local optimization procedure, under a *Lamarckian* evolution setting [1]. In each generation, L epochs of the *RPROP* learning algorithm are applied to each individual (*MLP*) in the population, using the examples in the training set. In past work [10], this Lamarckian approach (with macro mutation) to training outperformed eight *evolutionary algorithms* (using different crossovers and mutations) and gradient-based algorithms (e.g. Backpropagation and *RPROP*).

The *fitness* function is based in the *RMSE* (Eq. 1) computed over a validation set. The *selection* procedure is done by converting the fitness value into its ranking. Then, a roulette wheel scheme is applied, being used a substitution rate of 50%. Finally, the algorithm is stopped after G generations. The *SENN Ensemble (SENNE)* will be built using the best G individuals obtained during the evolutionary process, being the output computed as the average of the *MLPs*.

3 Results and Discussion

The *ANN/EC* experiments were conducted using a software package developed in *JAVA* by the authors. The other *DM/ML* techniques were computed using the *WEKA* software package with its default parameters [15]:

- *J48* – a classification decision tree based on the C4.5 algorithm;
- *M5P* – a regression decision tree (M5 algorithm);
- *IB5* – a *5-Nearest Neighbor*;
- *KStar* – an instance based algorithm; and
- *SVM* – a *Support Vector Machine*.

For each model, 10 runs of a 5-fold cross-validation process [4] (stratified in the classification tasks) were executed. This means that in each of these 50 experiments, 80% of the data is used for learning and 20% for testing.

With the pure *ANN* approaches, the learning data was divided into *training* (50% of the original dataset) and *validation* sets (30%). A different strategy was used for the *SENN*, since the simultaneous evolution of weights and topologies is very sensitive to overfitting. Thus, the validation set is divided into: a *fitness* set (15%), used for the fitness evaluation, and a *model selection* set (15%), used to select the best individual (or individuals when building an ensemble). The *SENN* parameters were set to $P = 20$, $H = 10$, $L = 50$, $M = 5$ and $G = 20$. Tables 2 and 3 show the average errors of the 10 runs for each model and task. The last row of each table averages the global performance of each learning strategy.

When comparing the classification results, the *ANN* learning models (last four columns) are competitive, outperforming the other *ML* algorithms. In effect, the few exceptions are the **dermatology** and **sonar** tasks where the *SVM* and *KStar* get the best results. Regarding the neural approaches, the *SENN* excels the *HNN* with a 1.1% difference in the average performance. Moreover, the ensemble approaches (*HNNE* and *SENNE*) obtain better results when compared with the single based versions, with improvements of 1.4% and 1.2%. Overall, the *SENNE* obtains the best predictive accuracy, being the best choice in 5 tasks.

Table 2. The classification results (*PCCE* values, in %)

Task	J48	IB5	KStar	SVM	HNN	SENN	HNNE	SENNE
Balance	78.1	87.6	88.3	87.7	94.8	96.1	95.7	96.7
Bupa	64.8	60.7	65.9	58.0	68.4	68.8	68.5	69.0
Car	91.3	92.3	87.1	93.5	97.4	98.5	98.3	98.8
Cmc	51.2	47.2	49.6	48.4	50.6	53.8	52.1	54.5
Dermatology	95.7	96.6	94.5	97.4	95.1	95.5	95.9	96.5
Ionosphere	89.4	84.6	84.0	87.9	88.9	89.9	92.3	91.9
Sonar	72.5	80.5	85.2	76.7	79.9	79.3	80.9	83.6
Yeast	56.0	57.1	53.1	56.6	58.2	59.9	59.9	60.0
Mean	74.9	75.8	76.0	75.8	79.1	80.2	80.5	81.4

Table 3. The regression results (*NRMSE* values, in %)

Task	M5P	IB5	KStar	SVM	HNN	SENN	HNNE	SENNE
Abalone	24.3	25.3	24.8	25.0	23.2	23.2	23.2	22.9
Auto-mpg	11.8	15.1	14.6	13.5	14.2	12.8	12.2	12.2
Autos	13.3	21.4	21.6	13.8	14.6	15.2	14.2	13.6
Breast-cancer	40.8	40.8	43.6	44.0	42.6	42.4	47.4	40.6
Heart-disease	21.2	21.0	25.5	21.5	21.9	23.3	22.2	21.7
Housing	18.4	22.2	18.0	22.6	18.4	17.1	16.6	15.8
Servo	50.4	60.4	67.6	70.5	60.8	43.4	49.6	38.3
Wpbc	73.2	73.6	98.2	71.7	75.4	75.7	80.3	71.7
Mean	33.0	35.0	39.2	35.3	33.9	31.6	33.2	29.6

For the regression tasks, the decision tree (*M5P*) is quite competitive, outperforming all non evolutionary approaches. As before, the *SENN* excels the *HNN* (2.3% improvement) and the ensembles behave better, with enhancements of 0.7% and 2.0%. Thus, the best alternative is the evolutionary ensemble (*SENNE*), followed by its single based version (*SENN*).

Similar work has been reported in the literature, namely the EPNet system [6], which obtained interesting results. However, this approach was only applied to five UCI datasets where the best results are obtained by low complexity *MLPs* (in some cases linear models). It is not surprising that, since EPNet heavily promotes simple models, good results were obtained for these cases. In this work, the majority of the problems demanded *MLPs* with a much higher number of hidden nodes, where is it believed that the EPNet system would not excel.

4 Conclusions

In this work, a *Simultaneous Evolutionary Neural Network (SENN)* algorithm is proposed, aiming at the optimization of the neural structure and weights. This approach was enhanced by considering ensembles, which combine the best *ANNs* obtained by the *SENN* approach. The results obtained in several real-world classification and regression tasks confirm the competitive *SENN* performances, when compared with a heuristic trial-and-error design procedure (*HNN*) and with other *DM/ML* algorithms.

Another advantage presented by the *SENN* is the reduced computational effort, when compared to an evolutionary algorithm that performs only topology optimization [12]. Indeed, with the current setup, the computational burden is similar to the one required by the *HNN*. Overall, the hybrid EC/*ANN* ensemble (*SENNE*) presents the best predictive accuracy while requiring no extra computation, thus being the advised choice. In future work, it is intended to test similar techniques with other *ANNs* (e.g., *Recurrent Neural Networks*). Furthermore, more elaborated ensembles could be considered, by designing fitness functions which reward specialization [7].

References

1. P. Cortez, M. Rocha, and J. Neves. A Lamarckian Approach for Neural Network Training. *Neural Processing Letters*, 15(2):105–116, April 2002.
2. T. Dietterich. Ensemble methods in machine learning. In J. Kittler and F. Roli, editors, *Multiple Classifier Systems, LNCS 1857*, pages 1–15. Springer, 2001.
3. S. Haykin. *Neural Networks - A Comprehensive Foundation*. Prentice-Hall, New Jersey, 2nd edition, 1999.
4. R. Kohavi. A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, Montreal, Quebec, Canada, August 1995.
5. T. Kwok and D. Yeung. Constructive algorithms for structure learning in feedforward neural networks for regression problems: A survey. *IEEE Transactions on Neural Networks*, 8(3):630–645, May 1999.
6. Y. Liu and X. Yao. Evolving Modular Neural Networks Which Generalize Well. In *Proc. of the 1997 IEEE Intern. Confer. on Evolutionary Computation, Indianapolis*, pages 670–675, New York, 1997. IEEE Press.
7. Y. Liu, X. Yao, and T. Higuchi. Evolutionary Ensembles with Negative Correlation Learning. *IEEE Transactions on Evolutionary Computation*, 4(4):380–387, 2000.
8. J.R. Quinlan. *Comparing Connectionist and Symbolic Learning Methods*, pages 445–456. MIT Press, Cambridge, Massachusetts, 1994.
9. M. Riedmiller. Supervised Learning in Multilayer Perceptrons - from Backpropagation to Adaptive Learning Techniques. *Computer Standards and Interfaces*, 16, 1994.
10. M. Rocha, P. Cortez, and J. Neves. Evolutionary Neural Network Learning. In F. Pires and S. Abreu, editors, *Progress in Artificial Intelligence, EPIA 2003 Proceedings, LNAI 2902*, pages 24–28, Beja, Portugal, December 2003. Springer.
11. M. Rocha, P. Cortez, and J. Neves. Ensembles of Artificial Neural Networks with Heterogeneous Topologies. In *Proceedings of the 4th Symposium on Engineering of Intelligent Systems (EIS2004)*. ICSC Academic Press, March 2004.
12. M. Rocha, P. Cortez, and J. Neves. Evolutionary Design of Neural Networks for Classification and Regression. In *ICANNGA Proceedings*, Coimbra, Portugal, March, 2005. Springer.
13. C. Soares. Is the UCI Repository Useful for Data Mining? In F. Pires and S. Abreu, editors, *Progress in Artificial Intelligence, EPIA 2003 Proceedings, LNAI 2902*, pages 209–223, Beja, Portugal, 2003. Springer.
14. G. Thimm and E. Fiesler. Evaluating pruning methods. In *Proc. of the Int. Symp. on Artificial Neural Networks*, pages 20–25, Taiwan, December 1995.
15. I. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, USA, 2000.
16. X. Yao. Evolving Artificial Neural Networks. In *Proc. of the IEEE*, 87(9): 1423–1447, September 1999.
17. X. Yao and Y. Liu. A New Evolutionary System for Evolving Artificial Neural Networks. *IEEE Transactions on Neural Networks*, 8(3):694–713, 1997.

Applying Bio-inspired Techniques to the p -Median Problem

E. Domínguez and J. Muñoz

Department of Computer Science,
E.T.S.I.Informática, University of Malaga,
Campus Teatinos s/n, 29071 – Malaga, Spain
{enriqued, munozp}@lcc.uma.es

Abstract. Neural networks (NNs) and genetic algorithms (GAs) are the two most popular bio-inspired techniques. Criticism of these approaches includes the tendency of recurrent neural networks to produce infeasible solutions, the lack of generalize of the self-organizing approaches, and the requirement of tuning many internal parameters and operators of genetic algorithms. This paper proposes a new technique which enables feasible solutions, removes the tuning phase, and improves solutions quality of typical combinatorial optimization problems as the p -median problem. Moreover, several biology inspired approaches are analyzed for solving traditional benchmarks.

1 Introduction

Solving NP-hard combinatorial optimization problems has been a core area in research for many communities in engineering, operations research and computer science. The interdisciplinary features of most NP-hard optimization problems have caused a large amount of papers from many researches that have proposed numerous and different algorithms to overcome the many difficulties of such problems. Several researches used algorithms based on the model of organic evolution as an attempt to solve hard optimization problems [1]. Due to their representation scheme for search points, genetic algorithms [2] are one of the most easily applicable representatives of evolutionary algorithms.

The idea of using neural networks to provide solutions to NP-hard optimization problems have been pursued for over decades. Hopfield and Tank [3] showed that the travelling salesman problem (TSP) could be solved using a Hopfield neural network. This technique requires minimization of an energy function containing several terms and parameters. Due to this technique was shown to often yield infeasible solutions, researchers tried to either modify the energy function or optimally tune the numerous parameters involved so that the neural network would converge to a feasible solution.

Both GAs and NNs are addressed in this work as competed paradigm for solving combinatorial optimization problems. There exist many NP-hard combinatorial optimization problems, although in this paper we selected the well-known

p -median problem as a preliminary study of the two bio-inspired techniques for solving NP-hard combinatorial optimization problems.

The outline of the paper is organized as follows: Section 2 presents the problem and a primarily integer programming formulation. In section 3 the analyzed bio-inspired techniques are described. The experimental results are showed in section 4, and the paper concludes summarizing our findings in section 5.

2 Problem Formulation

The p -median problem has been studied during decades and it is one of the most popular and well-known facility location problems. The first reference to this problem is found in the work of Hakimi (1964) [4] who proposed such optimization problem. The p -median problem concerns the location of p facilities (medians) in order to minimize the total weighted distances from each point (population center or customer) to its nearest facility. Kariv and Hakimi (1979) [5] showed that the p -median problem on a general network is NP-hard. Revelle and Swain [6] provided an integer programming formulation for the discrete p -median problem, which is given below

Minimize

$$\sum_{i=1}^n \sum_{j=1}^n d_{ij} x_{ij} \quad (1)$$

Subject to :

$$\sum_{j=1}^n x_{ij} = 1 \quad i = 1, \dots, n \quad (2)$$

$$\sum_{j=1}^n x_{jj} = p \quad (3)$$

$$x_{ij} \leq x_{jj} \quad i = 1, \dots, n; j = 1, \dots, n \quad (4)$$

where

n is the number of demand points

p is the number of facilities or medians

d_{ij} is the distance (cost) between the point i and the facility j

$$x_{ij} = \begin{cases} 1 & \text{if the point } i \text{ is assigned to the facility } j \\ 0 & \text{otherwise} \end{cases}$$

$$x_{jj} = \begin{cases} 1 & \text{if the point } j \text{ is a facility} \\ 0 & \text{otherwise} \end{cases}$$

The restrictions (2) prevent that all demand points are assigned to a facility. The constraints (3) establish the number of facilities (medians) to locate, and the last conditions (4) assure that a demand point is assigned to an opened facility.

A number of heuristic solution techniques have been developed in an effort to solve large-scale instances to near-optimality with a reasonable computational

effort. Initially, these proposed procedures were based on mathematical programming relaxation [7], dual problem [8] and branch-and-bound techniques [9]. Although one of the oldest algorithms presented by Teitz and Bart (1968) [10] is the most popular and practical heuristic for this problem due to simplicity and easy implementation. While these algorithms were used for solving small-scale instances of the p -median problem, modern heuristic solutions are required for solving large-scale instances, since these instances are intractable due to the necessary computational effort. Thus, some modern heuristics as tabu search [11], dynamic programming [12], heuristic concentration [13], variable neighborhood search [14], neural networks [15, 16], and genetic algorithms [17] have been applied to solve the p -median problem.

3 Bio-inspired Techniques for the p -Median Problem

In this section we present the bio-inspired techniques analyzed in this paper. We present a first subsection on GAs, and a second one on NNs. In order to design a suitable and efficient algorithm, appropriate formulations of the p -median problem were used to implement the algorithms. A discussion of different formulations of the p -median problem can be found in [18].

3.1 Genetic Algorithms

Genetic algorithms (GAs) were initially developed by Holland in the sixties. They are guided random search algorithms based on the model of biological evolution. The encoding is a critical decision since a poor choice may result a poor algorithm regardless its other features. Other important decisions include the population size, the selection of parents to produce new solutions, the crossover operator, the method and frequency of the mutations, and the number of generations.

Hosage and Goodchild [19] are one of the first to apply a GA for solving the p -median problem. In their study, the solutions of the problem were encoded as a binary string of n digits (genes). This encoding does not guarantee the feasibility of solutions, that is the selection of exactly p facilities in each solution, and the authors use a penalty function to impose this constraint. This is a inefficient encoding scheme, not only does it waste memory, but also it requires unnecessary operations for the crossover operator and for the computation of the fitness function.

The constructive genetic algorithm (cGA) proposed by Nogueira and Furtado [20] is other alternative to the traditional GA approach, particularly in that cGA evaluates directly schemata. This algorithm works with a dynamic population size which is enlarged after the use of recombination operators, or made smaller as evolution proceeds, guided by an evolution parameter. Other different is that the problem to be solved with a cGA are formulated as bi-objective optimization problems; thus, two fitness functions are defined.

In other more recent work Bozkaya, Zhang, and Erkut [21] describe a fairly complex GA and report the results comparing with the traditional exchange algorithm proposed by Teitz and Bart [10]. Although the GA produce better

solutions than the exchange algorithm, the converge of GA is very slow and spend a lot of computational time.

In contrast with the GA discussed above, a goal of this study is to construct an algorithm much simpler and very quickly that produces good solutions. In this sense, the genetic algorithm proposed in this work is described below.

Genetic Algorithm for the p -Median Problem. The encoding used is based on one of the models proposed by Dominguez and Muñoz [18] for solving the p -median problem. Each solution is encoded as an integer string of length p , where each gene of the chromosome indicate the index of facility selected as median. For instance, the string $\langle 2, 7, 9 \rangle$ represents a candidate solution for the 3-median problem where facilities 2, 7, and 9 are selected as medians.

The fitness function is directly related to the previously discussed objective function (1). More specifically, the fitness of an individual (c) is given by the expression (5), where n is the number of demand points, and d is the distance function between two points.

$$f(c) = \sum_{i=1}^n \left(\min_{1 \leq j \leq p} \{d(i, c_j)\} \right) \quad (5)$$

Bit-flip mutation and a uniform recombination is used with a given probability (0.2 and 0.6, respectively). Also, a roulette wheel selection is proposed in the evolution process. Further comments on the operators are given at the beginning of the forthcoming experimental section.

3.2 Neural Networks

Almost every type of combinatorial optimization problem has been tackled by neural networks as Hopfield and Tank [3] showed using a Hopfield network to solve the TSP. Researchers have been trying for over decades to make neural networks competitive with meta-heuristics such as simulated annealing, tabu search, or genetic algorithms. While it is well known neural networks are a very powerful for solving problems of prediction, classification and pattern recognition, they does not appear to be as successful when applied to optimization problems [22]. This reputation of neural networks is perhaps the cause that they are usually missed in many optimization works.

Particularly, the p -median problem is one of the combinatorial optimization problems with a relatively scarcity of references on neural networks. One of the most recent works is proposed by Dominguez and Muñoz [16]. In this paper, authors provide a recurrent neural network (RNN) similar to Hopfield network for solving the treated problem. Additionally, a reduced formulation for the p -median problem is proposed. This formulation allows to design a RNN with two types of process units (allocation and location units). Although the proposed neural network is very simple and easy to implement, the solution quality found using this technique raises some doubts for solving small-scale instances.

In this paper, we propose an improved RNN based on the previous work. The main purpose was to increase the solution quality. The comparative analysis show

that the solution quality using the improved neural network is better than the previous RNN.

Stochastic Neural Network for the p -Median Problem. Recurrent neural networks involves the execution of a gradient descent procedure many times, and selecting the best results obtained in these attempts. That is, the only acceptable transitions are made from higher energy to lower energy. But in the p -median problem that has many local minima it may not be going down the right descent. In this paper, we suggest an alternative approach that captures essential features of the metallurgical process of annealing.

The proposed stochastic neural network (SNN) is based on the RNN provided by Dominguez and Muñoz [16]. Also, the proposed model by the authors is used to design the SNN. Notation used in this paper is defined as follows

n is the number of demand points

p is the number of facilities or medians

d_{ij} is the distance (cost) between the point i and the facility j

$x_{iq} = \begin{cases} 1 & \text{if the demand point } i \text{ is assigned to the cluster } q \\ 0 & \text{otherwise} \end{cases}$

$y_{jq} = \begin{cases} 1 & \text{if the point } j \text{ is the facility of the cluster } q \\ 0 & \text{otherwise} \end{cases}$

$h_{x_{iq}}$ and $h_{y_{jq}}$ are the activation potential of the above decision variables

T_k is the temperature at the k th iteration

The dynamics of the proposed SNN is defined according to the following expressions

$$\begin{aligned}
 x_{iq} &= \begin{cases} 1 & \text{with probability } P(x_{iq}) \\ 0 & \text{with probability } 1 - P(x_{iq}) \end{cases} \\
 y_{jq} &= \begin{cases} 1 & \text{with probability } P(y_{jq}) \\ 0 & \text{with probability } 1 - P(y_{jq}) \end{cases}
 \end{aligned} \tag{6}$$

where

$$\begin{aligned}
 P(x_{iq}) &= \frac{e^{h_{x_{iq}}/T}}{\sum_{j=1}^n e^{h_{x_{jq}}/T}} \\
 P(y_{jq}) &= \frac{e^{h_{y_{jq}}/T}}{\sum_{r=1}^n e^{h_{y_{jr}}/T}}
 \end{aligned} \tag{7}$$

and T is the temperature of network. The decrement function of temperature is defined by $T_{k+1} = 0.999T_k$.

Table 1. Comparative results of bio-inspired techniques

pmed	(n, p)	Best	GA	RNN	SNN
1	(100, 5)	5819	5819	5868	5891
2	(100,10)	4093	4093	4300	4300
3	(100,10)	4250	4250	4359	4250
4	(100,20)	3034	3181	3173	3079
5	(100,33)	1355	1663	1743	1474
6	(200, 5)	7824	7824	7867	7867
7	(200,10)	5631	5631	5881	6038
8	(200,20)	4445	4681	4835	4897
9	(200,40)	2734	3260	3202	3029
10	(200,67)	1255	1809	1887	1463
11	(300, 5)	7696	7696	7762	7892
12	(300,10)	6634	6640	6810	6781
13	(300,30)	4374	4802	4852	4843
14	(300,60)	2968	3722	3568	3309
15	(300,100)	1729	2517	2492	2033
16	(400, 5)	8162	8162	8244	8209
17	(400,10)	6999	6999	7254	7392
18	(400,40)	4809	5456	5274	5199
19	(400,80)	2845	3662	3436	3180
20	(400,133)	1789	2813	2633	2128
21	(500, 5)	9138	9138	9294	9229
22	(500,10)	8579	8579	8794	8809
23	(500,50)	4619	5335	5056	5092
24	(500,100)	2961	3903	3573	3345
25	(500,167)	1828	2912	2774	2295
26	(600, 5)	9917	9917	10158	10160
27	(600,10)	8307	8310	8713	8761
28	(600,60)	4498	5322	4996	5018
29	(600,120)	3033	4052	3672	3479
30	(600,200)	1989	3193	2915	2414
31	(700, 5)	10086	10086	10141	10086
32	(700,10)	9297	9301	9771	9941
33	(700,70)	4700	5585	5294	5221
34	(700,140)	3013	4171	3760	3507
35	(800, 5)	10400	10400	10400	10467
36	(800,10)	9934	9934	10189	10336
37	(800,80)	5057	6178	5574	5584
38	(900, 5)	11060	11060	11304	11567
39	(900,10)	9423	9429	9742	9899
40	(900,90)	5128	6197	5772	5823

4 Experimental Results

The goal of the computational analysis is to evaluate the performance of the bio-inspired algorithms for solving the p -median problem. Initially, it is common

sense to expect some advantage of traditional customized solvers. In this case we are interested in analyzing the efficiency of the bio-inspired techniques in relation to the effort of algorithm construction and the computational time. In this sense, all analyzed algorithms are very simple and easily implementable. Moreover, we are interested in the final accuracy of the solutions of the provided methods.

For the experiments, we have performed 50 independent runs of each algorithm in order to provide meaningful average values in the tables. We explicitly show the error of the best solution as a percentage of its distance to the known optimum value for each instance. Benchmarks are provided by OR-Library which is available in <http://www.brunel.ac.uk/depts/ma/research/jeb/info.html>

In table 1 is shown the results obtained by the three bio-inspired techniques analyzed. The results show that the best method depends of the ratio n/p . For example, in small-scale instances ($n/p < 10$) GA is quite accurate. However, in medium-scale instances ($n/p > 10$) neural networks outperform genetic algorithms. Moreover, the accuracy of SNN is better than RNN, although the computational time needed to simulate SNN is greater than RNN. We must point out that these instances are in the range of 100 to 900 demand points, which represents a considerable step up in size.

5 Conclusions

In this work we are analyzed a genetic algorithm, a recurrent neural network, and a stochastic network for solving the p -median problem. The experimental results have shown that the genetic algorithms is simple and accurate for solving small-scale problem. We think that applying GA is a good first step for solving problems with a poor knowledge. A disadvantage of GA is the numerous parameters to tune for obtaining good results. In medium-scale instances we have shown that neural networks outperform genetic algorithms. Moreover, the stochastic version has been more accurate than the recurrent neural network.

We have considered problems with a large number of points and medians in the aim of providing a wide set of results and analyzing the performance of the bio-inspired algorithms. While local search or parallelism are important issues that have led to accurate algorithms, in this paper we intend to analyze simple and fast bio-inspired techniques for solving NP-hard problems.

The fact that a simple algorithm is able of computing that accurate solutions along with its simplicity can be a step towards making the research community of the potential inside bio-inspired techniques. This is an important point of view, since most authors research on new and complex algorithms reporting high accuracy results, but that are never actually used, because they are difficult to implement or understand.

References

1. Holland, J.: *Adaptation in Natural and Artificial Systems*. University of Michigan Press (1975)
2. Goldberg, D.: *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison Wesley (1989)
3. Hopfield, J., Tank, D.: Neural computation of decisions in optimization problems. *Biological Cybernetics* **52** (1985) 141–152
4. Hakimi, S.: Optimum locations of switching centers and absolute centers and medians of to graph. *Operations Research* **12** (1964) 450–459
5. Kariv, O., Hakimi, S.: An algorithmic approach to network location problem. part 2: The p-median. *SIAM J. Appl. Math.* **37** (1979) 539–560
6. ReVelle, C., Swain, R.: Central facilities location. *Geographical Analysis* **2** (1970) 30–42
7. Narula, S., Ogbu, U., Samuelsson, H.: An algorithm for the problem p-mediate. *Operations Research* **25** (1977) 709–713
8. Galvao, R.: A dual-bounded algorithm for the problem p-mediate. *Operations Research* **28** (1980) 1112–1121
9. Khumawala, B.: An efficient branch and bound algorithm for the warehouse location problem. *Management Science* **18** (1972) 718–731
10. Teitz, M., Bart, P.: Heuristic methods for estimating the generalized vertex median of a weighted graph. *Operations Research* **16** (1968) 955–961
11. Rolland, E., Schilling, D., J.R., C.: An efficient tabu search procedure for the p-median problem. *European Journal of Operational Research* **96** (1997) 329–342
12. Hribar, M., Daskin, M.: A dynamic programming heuristic for the problem p-mediate. *European Journal of Operational Research* **101** (1997) 499–508
13. Rosing, K., ReVelle, C.: Heuristic concentration: two stage solution construction. *European Journal of Operational Research* **97** (1997) 75–86
14. Hansen, P., Mladenovic, N.: Variable neighborhood search for the p-median problem. *Location Science* **5** (1997) 141–152
15. Lozano, S., Guerrero, F., Onieva, L., Larrañeta, J.: Kohonen maps for solving to class of location-allocation problems. *European Journal of Operational Research* **108** (1998) 106–117
16. Domínguez, E., Muñoz, J.: An efficient neural network for the p-median problem. *Lecture Notes in Artificial Intelligence* **2527** (2002) 460–469
17. Alp, O., Erkut, E., Drezner, Z.: An efficient genetic algorithm for the p-median problem. *Annals of Operations Research* **122** (2003) 21–42
18. Domínguez, E., Muñoz, J.: Integer programming formulations of discrete p-median problems. In: *EWGLA 2005*. (2004)
19. Hosage, C., Goodchild, M.: Discrete space location-allocation solutions from genetic algorithms. *Annals of Operational Research* **6** (1986) 35–46
20. Nogueira Lorena, L., Furtado, J.: Constructive genetic algorithm for clustering problems. *Evolutionary Computation* **9** (2001) 309–327
21. Bozkaya, B., Zhang, J., Erkut, E.: 6. In: *An Efficient Genetic Algorithm for the p-Median Problem*. Springer (2003) 179–205
22. Osman, I.H., Laporte, G.: Metaheuristics: A bibliography. *Annals of Operations Research* **63** (1996) 513–623

Optimal Strategy for Resource Allocation of Two-Dimensional Potts Model Using Genetic Algorithm

Wing Keung Cheung and Kwok Yip Szeto

Department of Physics,
Hong Kong University of Science and Technology,
Clear Water Bay, Hong Kong SAR, China
phszeto@ust.hk

Abstract. The problem of optimal strategies of resource allocation for companies competing in the shopping malls in a metropolis is investigated in the context of two-dimensional three state Potts model in statistical physics. The aim of each company is to find the best strategy of initial distribution of resource to achieve market dominance in the shortest time. Evolutionary Algorithm is used to encode the ensemble of initial patterns of three states Potts model and the fitness of the configuration is measured by the market share of a chosen company after a fixed number of Monte Carlo steps of evolution. Numerical simulation indicates that initial patterns with certain topological properties do evolve faster to market dominance. The description of these topological properties is measured by the degree distribution of each company. Insight on the initial patterns that entail fast dominance is discussed.

1 Introduction

Resource allocation in a competitive world of supermarket chains in a metropolis is an interesting problem in econophysics [1-3]. The objective for a company is to find a strategy of allocating finite resource, which is initially insufficient to dominate the market, so that over time it will defeat its competitors, with maximum market share. Since initially each company has limited resource, and there can be many companies competing, an approach based on a multi-agent model seems reasonable [2]. However, the necessity of obtaining general insights into this complex problem requires us to model the multi-agent system by simple interaction among the agents: agents of the same company will cooperate, while agents belonging to different companies will compete. This leads naturally to usage of standard models in Statistical physics where localized spins interact with their nearest neighbors [4]. The above scenario of cooperation among agents of same company and competition for agents of different companies implies that the ferromagnetic interaction among agents is the right choice of model. While previous works [1-3] address only two companies, this paper addresses the generalization to more than two companies. For two companies, the simplest model that incorporates the above interaction of agents is the ferromagnetic Ising model. The natural spin model in Physics for the introduction of more than two companies is the Potts model. Here we generalize our framework for optimal strategy selection in the Ising model of two supermarket chains in a metropolis [2] to the Potts

model. The resource of each company is not sufficient to open shops in all the malls and each mall can accommodate only one shop. The problem for each company is to allocate the resource for shops so as to maximize the market share in the future, taking into consideration of shops changing hands and/or relocation by the management of each company [3]. The basic tools used in our analysis are techniques in statistical mechanics and Monte Carlo simulation [5] and our primitive approach of solving this problem consists of two steps: (1) generate all the possible configurations with no violation of the resources constraint, (2) perform Monte Carlo simulation of the model based on statistical mechanics and observe the evolution of the pattern of the market shares of these three companies. Note that during the Monte Carlo evolution, the resource constraint is not imposed, so that some companies will gain at the expense of others. This approach however does not satisfactorily solve the problem of finding a good initial configuration of resource allocation as the number of possible initial configurations increases exponentially with the number of shopping malls. A better way to address the selection of initial configuration of shops allocation is to reformulate our problem as an optimization problem. We first fix the time scale of Monte Carlo simulation by fixing the number of Monte Carlo steps. We then inspect the initial patterns for those that lead to market dominance of a chosen company. Thus, the problem becomes a search of the set of good (or fit) initial configurations of resource allocation in the space of all possible configurations. By good configuration, we define a measure of fitness, corresponding to the market share of a chosen company. A natural technique for this search problem is Genetic Algorithm [6]. In this paper, we first introduce the model based on statistical mechanics, with well defined constraints on resources and the associated switching dynamics. We then describe the implementation of the evolutionary algorithm and Monte Carlo simulation. Finally, we make use of the topological analysis to analyze the optimal configuration.

2 The Model and Resources Constraints

The ferromagnetic Potts model in statistical physics is employed to study agent-based interacting companies. This model has been extensively studied in physics of magnetism [4]. It considers three spin states, which in our present context of econophysics becomes three competing companies, A (Black) and B (Grey) and C (White) on the lattice, and describes the interaction between two nearest neighbor spins on the lattice (with site labeled by (i, j) with energy (E) ,

$$E = \sum_{u=A}^C \sum_{v=A}^C \sum_{\langle i, j \rangle} J_{ij}^{uv} x_i^u x_j^v \quad \text{with} \quad J_{ij}^{uv} = \begin{cases} 0 & \text{if } i = j \\ -1 & \text{if } i \neq j \text{ and } u = v \\ 1 & \text{if } i \neq j \text{ and } u \neq v \end{cases} \quad (1)$$

Here J_{ij}^{uv} is the interaction between two nearest-neighboring agents, x_i^u is the binary vector represents the occupancy of company u on the shopping mall located at site i . We assume in this paper the three companies are setting up their shops in a hexagonal lattice represent shopping malls. Only one shop is available for each mall. The

analogy of market competition is shown in *Figure 1*. The resources constraint of the initial configuration is the same for each company, 1/3 of the total number of shopping malls. The total number of shopping malls available is 225. Each company must use up all the resources available. Each hexagon can be assigned one of the three colors (Black, Grey or White).

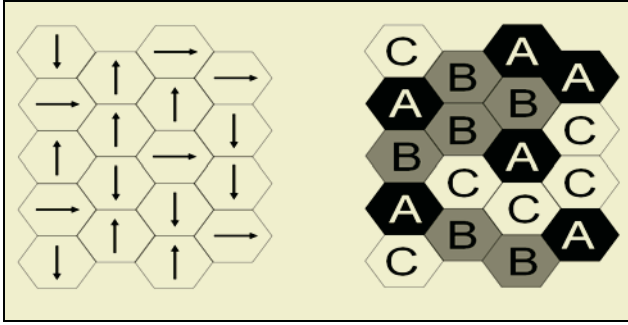


Fig. 1. Spin Model and Company Model

3 Switching Dynamics in Monte Carlo Simulation

We consider the One-shop switching process in our Monte Carlo simulation [5] for the evolution of pattern. Let's assume that the energy change resulted from a switch of a given hexagon to color X be ΔE_i^X . If $\Delta E_i^A, \Delta E_i^B, \Delta E_i^C$ are all non negative at mall i , the transition probability will be calculated by partition of unity and the transition probability of switching for company to X is

$$P_i^X = \frac{e^{-\beta(\Delta E_i^X)}}{e^{-\beta(\Delta E_i^A)} + e^{-\beta(\Delta E_i^B)} + e^{-\beta(\Delta E_i^C)}} \quad (2)$$

Otherwise, the company will switch to X when ΔE_i^X is the most negative value among $\Delta E_i^A, \Delta E_i^B, \Delta E_i^C$. The T in the factor $\beta = 1/(kT)$ is temperature and it represents the thermal noise and the Boltzmann constant k is set to 1. Discussion of the interpretation of this model in economics can be found in [2].

4 Evolutionary Algorithm: Encoding of the Initial Pattern

The strategy of shops allocation can be represented by a triple chromosome. In this paper, we have 225 malls in our city so that the length of the chromosome is 225. We use "1", "2", and "3" to represent shop occupancy of company A, B, and C

respectively. To generate a population of initial configurations, we first prepare a chromosome with 1/3 “1”, 1/3 “2” and 1/3 “3”, and then randomly swap the position of loci. We repeat this procedure until sufficient chromosomes are generated and that form the initial population of our genetic algorithm. Monte Carlo simulation is then performed to evolve the pattern according to the switching dynamics. In each MC step, only one shop is switched. This MC switching mechanism is shown in *Figure 2*.

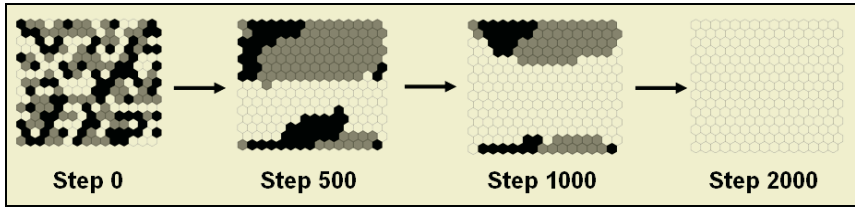


Fig. 2. Monte Carlo Simulation of Market Competition Model

One should note that the switching dynamics depends on temperature. In the context of econophysics, the switching process represents the change of loyalty of a particular agent from one company to another. Temperature, or the effect of thermal noise, corresponds for example to the effects of promotion to an agent in the process of competing for customers to switch their loyalty in a noisy environment. With only 2000 MC steps, we already observe the emergence of a dominant company.

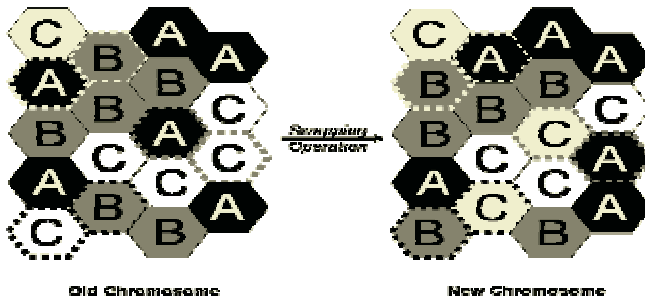


Fig. 3. The Swapping Operation from an old chromosome to a new one

The market share of a chosen company (here C) is computed after 2000 MC steps and it is defined as the fitness function of the chromosome in the population pool of initial configurations. (Note that a given initial configuration corresponds to a given strategy of resource allocation of the company on the hexagonal network.) After the fitness is computed for each chromosome of the population, selection based on fitness is performed. Regeneration of the pool of chromosomes for the non-survivors is

required. Previous study of Cheung and Szeto [3] employs mutation and repairing operators to implement this regeneration with resources constraint. However, this implementation is not efficient in the computational point of view. In most cases, repairing operator has to be activated to correct the target chromosome so that the initial configuration has equal occupancy for each of the three companies. Here we introduce a new swapping operation by assigning a swapping rate on each locus. If the swapping rate at a given locus K is larger than some given value called the global swapping rate, then another locus L is chosen at random and K and L will be swapped. This operation guarantees that each new configuration satisfies the constraint on resources if the old configuration also satisfies the resource constraint. *Figure 3* demonstrates the swapping mechanism. By repeating the whole process, we can determine the optimal initial configuration that will evolve to the highest market share. The optimal configuration is analyzed for its topological properties, which is a common technique in high dimensional network analysis [7]. Here we use the connectivity distribution, which is the distribution of number of edges connected to the neighboring shops from a mall, as the quantitative measurement of topological properties. The pseudo-code below shows the implementation of our algorithm

The *pseudo-code* of our Genetic Algorithm for Resource Allocation

- ```
(1) Prepare a set of initial configuration (first generation)
 Do {
 (2A) Perform Monte Carlo Simulation to evolve the pattern
 (2B) Compute the fitness function (Market Share)
 (2C) Compute Survival of Fitness
 (2D) Apply Swapping Operator
 (2E) New Generation is made
 } until 100 generations are made
(3) Determine the optimal configuration
(4) Analysis of this optimal configuration by topological analysis
```

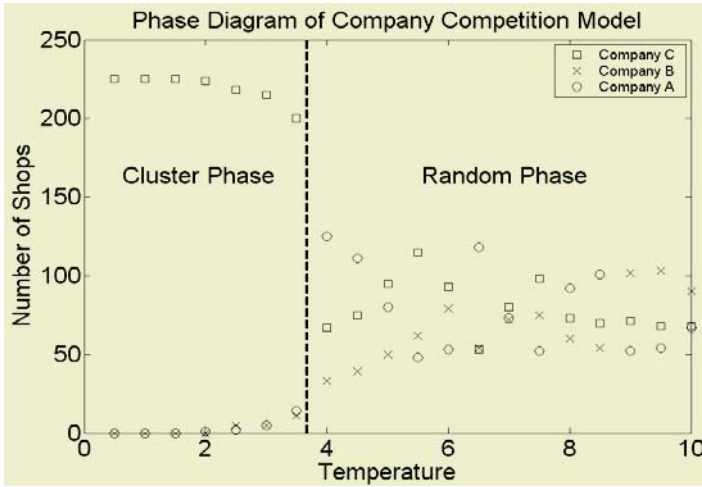
## 5 Result and Discussion

The parameters of our model are shown in *Table 1*.

**Table 1.** The parameters in genetic algorithm and Monte Carlo Simulation

| <i>Genetic Algorithm Parameter</i> | <i>Value</i> | <i>Monte Carlo Parameter</i> | <i>Value</i>   |
|------------------------------------|--------------|------------------------------|----------------|
| Population Size                    | 50           | Temperature                  | 0.5, 1, 1.5, 2 |
| Probability of swapping            | 0.03         | MC Step                      | 2000           |
| Number of generation               | 100          |                              |                |

Since the interesting case concerns market dominance, which only occurs at low noise level (temperature), we only discuss the result for the low  $T$  case. At temperature higher than a critical value, no company will dominate the market. We will leave the discussion of the critical temperature in another paper. *Figure 4* shows the market share as a function of temperature.



**Fig. 4.** Two Distinct Phases: Cluster phase and Random phase

For the low noise level simulation, the final cluster state strongly depends on the initial configuration of shops. Thus our question of finding a good initial configuration is very meaningful and economically important. On the other hand, at high noise level, the final state evolved from any initial state is a randomized pattern, so that the discussion of what is the optimal initial configuration will be meaningless. This is the reason why we only analyse the low noise ( $T$ ) case for the dependence of market share on initial configuration. The optimal strategies for “white” company at noise level = 0.5, 1, 1.5, 2 are extracted. They are shown in *Figure 5*.

We observe that all optimal configurations for “White” company can gain more than 85% average market share after Monte Carlo simulation. Qualitatively, all the “White” shops tend to form links among themselves so as to dominate the market. We further investigate the configuration by computing the connectivity distribution function. It is shown in *Figure 6*. We observe that a common feature among the winners, which are the white hexagons that evolve to dominate the market. We see that the connectivity distributions for the white company are all wider than the others. This observation is confirmed by fitting the connectivity distribution by a Gaussian and extract for the width. One of the cases is shown in *Table 2*. We find that the width of the connectivity distribution for the winner is higher than the loser in all the noise level tested and the peak of the Gaussian fitting is located around two. This latter fact is indicative of link formation.

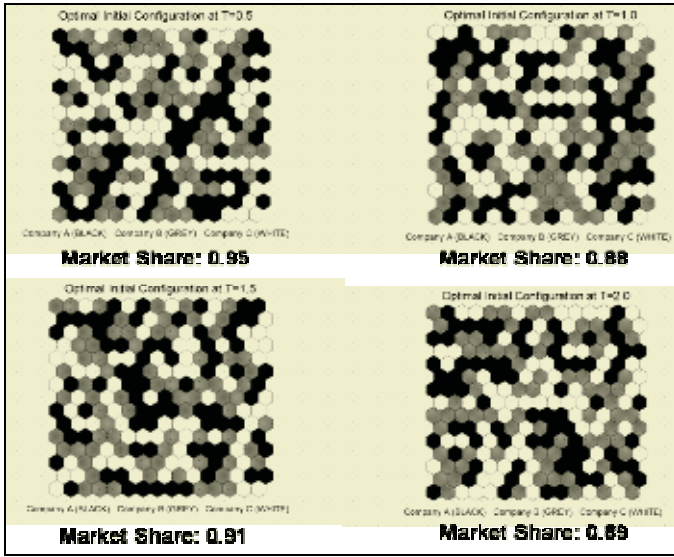


Fig. 5. The Optimal Shops Allocation and Final Market Share at Noise Level = 0.5, 1, 1.5, 2

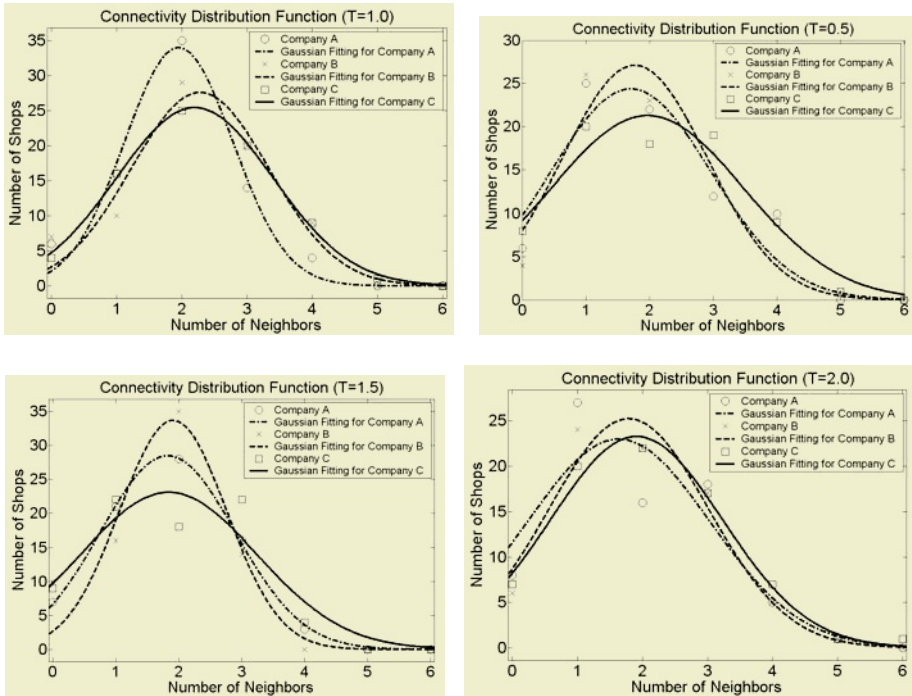


Fig. 6. The Connectivity function with Gaussian Fitting at Noise Level = 0.5, 1, 1.5, 2



**Table 2.** The Gaussian Fitting for the Connectivity Function at Noise Level = 0.5

| <i>Gaussian Fitting Function</i> | <i>Center</i> | <i>Width</i> |
|----------------------------------|---------------|--------------|
| Company A                        | 1.705         | 1.782        |
| Company B                        | 1.768         | 1.603        |
| <b>Company C</b>                 | <b>1.967</b>  | <b>2.14</b>  |

Another interesting observation is that one of the opponents (Grey) has a similar connectivity function to the winner (White). This is indicative of the alliance of White and Grey to defeat the loser, which is Black. In this sense, the Potts model of three companies will evolve towards a model with only two companies, (Ising like), consisting of only the White and Grey color. Further evolution eliminates the weaker company, which is Grey. This gradual change in the competition among three companies to two companies and eventual dominance of one company has been clearly observed from our simulation. The topological reason for the White company becoming the final winner is its possessing more links than the other two companies. It will be interesting to investigate if this conclusion on topological property carries over to more general point patterns of shopping malls [8], and the relevance of damage spreading [9] in a competitive world.

## Acknowledgement

K.Y. Szeto acknowledges the support of grant CERG HKUST6157/01P.

## References

- [1] Chiwah Kong, MPhil. Thesis, Hong Kong University of Science and Technology. 2002
- [2] Kwok Yip Szeto and Chiwah Kong : Different Phases in a Supermarket Chain Network: An Application of an Ising Model on Soap Froth, *Computational Economics*, 22(2): 163-17, 2003
- [3] Wing Keung Cheung and K. Y. Szeto, Strategies for Resource Allocation of Two Competing Companies using Genetic Algorithm, *Proceedings of the Fifth International Conference on Recent Advances in Soft Computing*, Nottingham, United Kingdom, 16-18 December 2004, P.416-421
- [4] K.Huang, *Statistical Mechanics*, Wiley, 1987.
- [5] M. E. J. Newman, G. T. Barkema (1999), *Monte Carlo Methods in Statistical Physics*, Clarendon Press Oxford, 1999, p. 45-59
- [6] Zbigniew Michalewicz (1994), *Genetic Algorithms + Data Structures = Evolution Programs*, Berlin, Hong Kong, Springer-Verlag, 1994, p.119-140
- [7] R. Albert and A.-L. Barabasi, *Rev. Mod. Phys.* 74, 47(2002), and references therein.
- [8] Xiujun Fu, Kwok Yip Szeto, *and* Wing Keung Cheung Phase transition of Ising Model on two-dimensional random point patterns. *Physical Review E70*, 056123(2004)
- [9] Z.Z.Guo, K.Y. Szeto, and Xiujun Fu, Damage spreading on two-dimensional trivalent structures with Glauber dynamics: Hierarchical and random lattices, *Phys. Rev. E70*, 016105(2004)

# Memetic Algorithms to Product-Unit Neural Networks for Regression<sup>1</sup>

Francisco Martínez-Estudillo<sup>1</sup>, César Hervás-Martínez<sup>2</sup>,  
Alfonso Martínez-Estudillo<sup>1</sup>, and Domingo Ortíz-Boyer<sup>2</sup>

<sup>1</sup> Department of Management and Quantitative Methods, ETEA, Spain  
{acme, fjmestud}@etea.com

<sup>2</sup> Department of Computing and Numerical Analysis of the University of Córdoba, Spain  
{chervas, ma1orbod}@uco.es

**Abstract.** In this paper we present a new method for hybrid evolutionary algorithms where only a few best individuals are subject to local optimization. Moreover, the optimization algorithm is only applied at specific stages of the evolutionary process. The key aspect of our work is the use of a clustering algorithm to select the individuals to be optimized. The underlying idea is that we can achieve a very good performance if, instead of optimizing many very similar individuals, we optimize just a few different individuals. This approach is less computationally expensive. Our results show a very interesting performance when this model is compared to other standard algorithms. The proposed model is evaluated in the optimization of the structure and weights of product-unit based neural networks.

## 1 Introduction

Evolutionary algorithms (EAs) are efficient at exploring the entire search space; however, they are relatively poor at finding the precise optimum solution in the region where the algorithm converges to [1]. During the last few years new methods have been developed in order to improve the lack of precision of the EAs using local optimization algorithms [2]. These new methodologies are based on the combination of local optimization procedures, which are good at finding local optima (local exploiters), and evolutionary algorithms (global explorers). These are commonly known as hybrid algorithms.

In this paper, we propose an alternative approach to hybrid algorithms using local optimization procedures. The methodology is based on the combination of an evolutionary algorithm, a clustering process and a local improvement procedure. If we want to efficiently use the hybrid algorithm, we have to reduce the computation time spent by the local search. So, our approach is to select a subset of the best individuals, perform a cluster analysis to group them, and optimize only the best individual of every group.

---

<sup>1</sup> This work has been financed in part by the TIC2002-04036-C05-02 project of the Spanish Inter-Ministerial Commission of Science and Technology (CICYT) and FEDER funds.

The main advantage of this method is that the computational cost of applying the optimization algorithm to just a few individuals hardly affects the total time spent by the algorithm. On the other hand, the use of a clustering algorithm allows the selection of individuals representing different regions in the search space. In this way, the optimized individuals are more likely to converge towards different local optima. The model developed is applied to the evolution of the structure and weights of evolutionary product-unit based neural networks. These kinds of networks are a very interesting alternative to sigmoid-based neural networks, but their major drawback is that the optimization algorithms usually used for training a network are quite inefficient for training product-unit networks. So, an effective algorithm for training these networks is of great practical interest. In order to test the performance of the proposed algorithms, the networks are applied to benchmark regression problems.

This paper is organized as follows: Section 2 describes our model in depth; Section 3 is dedicated to a short description of product-unit based networks; Section 4 states the most relevant aspects of the evolution of product-unit neural networks using the proposed approach; Section 5 explains the experiments carried out; and finally Section 6 summarizes the conclusions of our work.

## 2 Hybrid Evolutionary Programming Algorithms

We propose two methods of hybrid evolutionary algorithms based on the use of a clustering algorithm for deciding which individuals are subject to local optimization. We have two different versions for the hybrid evolutionary algorithm depending on the stage when we carry out the local search and the cluster partitioning. The hybrid evolutionary programming with the clustering (HEPC) algorithm applies the clustering process on a large enough subset of the best individuals of the final population. In this method it is very important to determine the number of best individuals to consider as well as the number of clusters. After that, we apply the L-M algorithm to the best individual of each cluster. The algorithm named dynamic hybrid evolutionary programming with clustering (Dynamic HEPC) carries out both the clustering process and the L-M local search dynamically every  $G_0$  generation. The final solution is the best individual among the local optima found during the evolutionary process. The basic aim of our methodology is the optimization of the number of times a local optimization algorithm is applied without reducing the efficacy of this algorithm. This is especially important when the algorithm is of a high computational cost. On the other hand, removing the local optimization procedure usually yields a worse performance. So, our method is a good compromise, as we applied the optimization algorithm to a reduced number of individuals. Moreover, the clustering process avails us with the possibility of selecting a subset of individuals with different features. In this way, the optimization algorithm is more efficient.

The local optimization algorithm used in our work is the Levenberg- Marquardt (L-M) optimization method. This algorithm is designed specifically for minimizing a sum-of-squares error [3]. In any case, any other local optimization algorithm can be used in a particular problem. Another feature of our approach is that the optimized

individuals are not included in the new population. Once the optimization algorithm is applied, we think that any further modification of the individual would be counter-productive. So, these individuals are stored in a separate population till the end of the evolutionary algorithm.

### 3 Product-Unit Neural Networks

In order to test the validity of our model we have chosen a difficult problem, hard enough to justify the use of complex approaches. The problem is the automatic determination of the structure and weights of product-unit neural networks [4]. Product units enable a neural network to form higher-order combinations of inputs, having the advantages of increased information capacity and smaller network architectures when we have interaction between the input variables. Neurons with multiplicative responses can act as powerful computational elements in real neural networks. Product-unit based neural networks have a major drawback, since their training is more difficult than the training of standard sigmoid based networks. Unfortunately, the error surface for product units can be extremely convoluted, with numerous minima that trap backpropagation. This is because small changes in the exponents can cause large changes in the total error. Several efforts have been made to develop learning methods for product units [5],[6],[7]. Let us consider the family of real functions  $F$  defined by:

$$F = \left\{ f : A \subset \mathbb{R}^k \rightarrow \mathbb{R} : f(\mathbf{x}, \boldsymbol{\theta}) = \sum_{j=1}^m \beta_j \left( \prod_{i=1}^k x_i^{w_{ji}} \right) \right\}$$

where  $\mathbf{x} = (x_1, x_2, \dots, x_k)$ ,  $\boldsymbol{\theta} = (\beta_j, w_{ji})_{i,j}$ ,  $\beta_j \in [-M, M] \subset \mathbb{R}$ ,  $w_{ji} \in [-L, L] \subset \mathbb{R}$ ,  $i = 1, 2, \dots, k$ ,  $j = 1, 2, \dots, m$  y  $m \in \mathbb{N}$ . The domain of the definition of  $f$  is the subset  $A$  of  $\mathbb{R}^k$  given by  $A = \{(x_1, x_2, \dots, x_n) \in \mathbb{R}^k : 0 < x_i \leq k_0\}$ . Using the Stone-Weierstrass Theorem it is straightforward to prove that this family of functions is a dense subset of the continuous functions space defined in a compact. On the other hand, every function of the family can be represented as a neural network. The network must have the following structure: an input layer with a node for every input variable, a hidden layer with several nodes, and an output layer with just one node. There are no connections between the nodes of a layer and none between the input and output layers either. The network has  $k$  inputs that represent the independent variables of the model,  $m$  nodes in the hidden layer, and one node in the output layer.

The activation of  $j$ -th node of the hidden layer is given by  $B_j(\mathbf{x}, \mathbf{w}_j) = \prod_{i=1}^k x_i^{w_{ji}}$  where  $w_{ji}$  is the weight of the connection between input node  $i$  and hidden node  $j$ . The activation of the output node is given by  $\sum_{j=1}^m \beta_j B_j(\mathbf{x}, \mathbf{w}_j)$  where  $\beta_j$  is the weight of the connection between the hidden node  $j$  and the output node. The transfer function of all nodes is the identity function.

## 4 Hybrid Evolutionary Model

The evolution of product-unit networks uses the operations of replication and two types of mutation: structural and parametric. There is no crossover, as this operation is usually regarded as harmful [8] for network evolution. In the following paragraphs we are going to describe each one of the different aspects of the algorithm in detail.

**Initial population.** We generate randomly  $10N_p$  networks and we select the best  $N_p$  among them. So, we construct the initial population  $B$  whose size is  $N_p$ .

**Selection plan.** The  $r\%$  best individuals of the population are selected from the set  $B^* = B - \{\text{best individual of } B\}$  of cardinality  $N^* = N_p - 1$ . With these individuals we make up the population  $B'$  of size  $\lfloor rN_p / 100 \rfloor$ .

**Structural and parametric mutations.** Every individual of the population  $B'$  is subject to structural mutation, obtaining  $B'_{struct}$ . The parametric mutation is applied only to the best  $(100-r)N_p^* / 100$  individuals of  $B'$  obtaining  $B'_{param}$ . We construct the population  $B'' = B'_{struct} \cup B'_{param}$ , where the cardinality of  $B''$  is  $N^* = N_p - 1$ .

The structural mutation implies a modification in the structure of the function performed by the network and allows an exploration of different regions in the search space. There are five different structural mutations: node addition, node deletion, connection addition, connection deletion and node fusion. Parametric mutation is accomplished for each coefficient of the model with Gaussian noise, using a self-adaptive annealing algorithm. For more details see [9].

**Updated plan.** The new population will be  $B = B'' \cup \{\text{Best of } B\}$ .

### 4.1 Clustering Partitioning Techniques

Let  $D = \{(\mathbf{x}_l, y_l) : l = 1, 2, \dots, n_T\}$  be the training data set, where the number of samples is  $n_T$ . We define the following application from the family of functions  $F$  to the Euclidean space  $\mathbb{R}^{n_T}$ :

$$H : F \rightarrow \mathbb{R}^{n_T}$$

$$f(\mathbf{x}, \boldsymbol{\theta}) \rightarrow H(f(\mathbf{x}, \boldsymbol{\theta})) = \hat{y}_f = \left( f(\mathbf{x}_l, \boldsymbol{\theta}) \right)_{l=1,2,\dots,n_T}$$

where  $\boldsymbol{\theta}$  is the set of parameters of  $f$ . The application assigns to each function of the family the vector obtained with the values of the function over the training data set. Thus we can define the distance between two functions of the family as the Euclidean distance between the associated vectors:

$$d(f, g) = \|\hat{y}_f - \hat{y}_g\| = \left[ \sum_{l=1}^{n_T} |f(\mathbf{x}_l, \boldsymbol{\theta}) - g(\mathbf{x}_l, \boldsymbol{\theta})|^2 \right]^{1/2}$$

With this distance measure, the proximity between two functions is related to their performance. So, similar functions using this distance will have a similar performance over the same regression problem. Now, considering a set of functions  $\{f_1, f_2, \dots, f_M\}$  of the family  $F$ , we can build the set of associated vectors  $\{\hat{y}_{f_1}, \hat{y}_{f_2}, \dots, \hat{y}_{f_M}\}$  of  $\mathbb{R}^n$ . The minimum sum-of-squares clustering problem is to find a partition  $P = \{C_1, C_2, \dots, C_K\}$  of  $\{\hat{y}_{f_1}, \hat{y}_{f_2}, \dots, \hat{y}_{f_M}\}$  in  $K$  disjoint subsets (clusters) such that the sum of squared distances from each point to the centroid of its cluster is minimum. We use k-means clustering [10]. In this algorithm, the cluster centroid is defined as the mean data vector averaged over all items in the cluster. The number of clusters must be pre-assigned.

## 4.2 Hybrid Evolutionary Algorithm

As stated, we have two different algorithms depending on the stage when the clustering and local search algorithms are carried out:

1. **Algorithm HEPC.** We apply the clustering process to the best  $\tilde{s}N_p$  individuals of the final population which is divided into  $K$  clusters  $C_1, C_2, \dots, C_K$ . After that, we apply the L-M algorithm to the best individual of each cluster. The individuals obtained with the local-search algorithm are stored in a local optimum set  $C$ .
2. **Algorithm Dynamic HEPC.** We apply the clustering process and the L-M algorithm to the best individual of each cluster every  $G_0$  generation and in the final population. The clustering process is applied on each selected generation to the best  $\tilde{s}N_p$  individuals of the current population. The individuals obtained with the local-search algorithm are stored in  $C$ .

In cases 1 and 2, the final solution is the best individual among the local optimum of set  $C$ .

On the other hand, the parameters used are: the exponents  $w_{ji}$  are initialized in the interval  $[-5, 5]$ , the coefficients  $\beta_j$  are initialized in  $[-10, 10]$ . The size of the population is  $N_p = 1000$ . The maximum number of generations is 4000. The only parameter of the L-M algorithm is the tolerance of the error to stop the algorithm, in our experiment this parameter has the value 0.01. The k-means algorithm is applied to 25% of the best individuals of the population. The number of clusters  $K$  is 4 in the static version of HEPC. In the dynamic version the number of clusters varies from 6, at the beginning of the evolution, to a final value of 4 in the last generation.

## 5 Experiments

In order to test the performance of the proposed algorithms, they are applied to two benchmark regression problems: Friedman #1 and Sugeno functions.

### 5.1 Friedman#1 Function

This is a synthetic benchmark dataset proposed in [11]. The function is given by:

$$f(\mathbf{x}) = 10 \sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5 + \varepsilon$$

where  $\varepsilon$  is a Gaussian random noise  $N(0;1)$  and the input variables are uniformly distributed in the interval  $(0;1]$ . 1000 samples are created randomly, 750 are randomly selected for the training set, and the remaining 250 are used as a test set. Table 1 shows the results for Friedman#1 function. The final solution has 6 nodes in the hidden layer. Table 1 shows a comparison of the results of the different algorithms proposed, where  $MSE_G$  is

$$MSE_G = \frac{1}{n_G} \sum_{i=1}^{n_G} [y_i - \hat{y}_i]^2$$

where the  $\hat{y}_i$  are the predicted values. In order to test the soundness of the clustering approach, we carried out two additional experiments. First, the same methodology of HEPC was applied, removing the cluster analysis. Instead of performing the clustering, we applied the L-M algorithm to  $K = 4$  randomly selected individuals from the best 25% of the population. The generalization results after 10 runs of the algorithm using both methods are shown in Table 2.

**Table 1.** Statistics of  $MSE_G$  for different models in the Friedman#1 problem. Results of NeuralBAG (NBAG), Simple and General Regression Neural Networks (GRNNFA) are adapted from [12] and [13]

| Algorithms   | Mean  | SD    | Runs |
|--------------|-------|-------|------|
| NBAG         | 4.502 | 0.268 | 20   |
| Simple       | 4.948 | 0.589 | 20   |
| GRNNFA       | 4.563 | 0.195 | 20   |
| HEPC         | 1.105 | 0.102 | 30   |
| Dynamic HEPC | 1.081 | 0.040 | 30   |

**Table 2.** Statistics of  $MSE_G$  for Friedman#1 function for 10 runs of HEPC and randomly selecting the individuals to optimize

| Algorithm | Mean  | SD    | Best  | Worst | Mann-Whitney's U test |
|-----------|-------|-------|-------|-------|-----------------------|
| HEPC      | 1.095 | 0.114 | 1.066 | 1.607 |                       |
| Random    | 1.408 | 0.665 | 1.068 | 2.981 | p value=0.003         |

Table 2 shows that, on average, the results using the clustering are better and have less variance. We performed a Mann-Whitney's U test, as the results of HEPC did not follow a normal distribution, with a p-value of 0.003. We can conclude that the application of the clustering algorithm improves the results for the Friedman function.

The second experiment was carried out to confirm the performance of our approach. The HEPC method was compared to an evolutionary process, denoted by L-M all, where, instead of applying the optimization algorithm to the best individual of each cluster, we apply the optimization algorithm to every individual in the best 25% of the population. The results in performance and time are shown in Table 3.

**Table 3.** Results for Friedman#1 function for 10 runs of HEPC and selecting all the individuals to be optimized. The time is measured in average seconds per generation

| Algorithm | MSE <sub>G</sub> |       |       |       |        |
|-----------|------------------|-------|-------|-------|--------|
|           | Mean             | SD    | Best  | Worst | t-test |
| HEPC      | 1.081            | 0.009 | 1.064 | 1.093 |        |
| L-M all   | 1.081            | 0.010 | 1.071 | 1.098 | 0.993  |
| Algorithm | Time             |       |       |       |        |
|           | Mean             | SD    | Best  | Worst | t-test |
| HEPC      | 5.3              | 0.67  | 4     | 6     |        |
| L-M all   | 76.9             | 8.65  | 64    | 91    | 0.000  |

## 5.2 Sugeno Function

The second benchmark problem concerns an approximation of the Sugeno function defined as  $f(x_1, x_2, x_3) = (1 + x_1^{0.5} + x_2^{-1} + x_3^{-1.5})^2$ . For training, 216 points for [1,6] interval and 125 points for testing from [1.5,5.5] interval were randomly created. The Average Percentage Error (APE) was used as a measure of approximation error

$$APE = \frac{1}{N} \sum_{i=1}^n \left| \frac{\hat{y}_i - y_i}{y_i} \right| * 100\%$$

Final networks had at most 11 nodes in the hidden layer. Results using Dynamic HEPC are compared to other results obtained by Kosinski and Hirikawa [14] and Jankowski [15] (see Table 4).

**Table 4.** Results for the Sugeno function for 30 runs of dynamic HEPC. Results of Fuzzy Neural Networks (FNN3), MDelt, FuzzyVINET, Incremental Neural Network (IncNet) and Incremental Neural Network with Rotation (Incnet Rot) are adapted from [15]

|                  | FNN3 | MDelt | FuzzyIne | FuzzyVine | IncN  | IncNetRo | Dyn.HEPC |
|------------------|------|-------|----------|-----------|-------|----------|----------|
| APE <sub>T</sub> | 0.63 | 0.72  | 0.18     | 0.076     | 0.119 | 0.053    | 0.065    |
| APE <sub>G</sub> | 1.25 | 0.74  | 0.24     | 0.18      | 0.122 | 0.061    | 0.060    |

## 6 Conclusions

In this work we have proposed a new method for using local optimization procedures in hybrid evolutionary algorithms. This approach is based on the application of a



clustering algorithm for the selection of a small number of individuals subject to local optimization. The algorithm is applied to the optimization of the structure and weights of product-unit based neural networks. The results obtained in two benchmark problems of regression show that the hybrid algorithm provides a very good compromise between performance and computational cost.

## References

- [1] Houck, C. R., Joines, J. A., and Kay, M. G.: Empirical investigation of the benefits of partial Lamarckianism, *Evolutionary Computation*, Vol. 5, no. 1, (1997), pp. 31-60.
- [2] Moscató, P. and Cotta, C., "A gentle introduction to memetic algorithms": In: Glover, F. and Kochenberger, G., (eds.): *Handbook of Metaheuristics*, pp. 1-56. Kluwer, 1999.
- [3] Bishop, C. M.: *Neural Networks for Pattern Recognition*. Oxford University Press, (1995).
- [4] Durbin, R., Rumelhart D.: Product units: A computationally powerful and biologically plausible extension to backpropagation networks. *Neural Computation*, Vol. 1, (1989), pp. 133-142.
- [5] Ismail, A., Engelbrecht, A. P.: Training product units in feedforward neural networks using particle swarm optimisation. In: Bajic, V. B., Sha, D. (eds.): *Development and Practice of Artificial Intelligence Techniques, Proceeding of the International Conference on Artificial Intelligence*, Durban, South Africa, (1999), pp. 36-40.
- [6] Leerink, L. R., Giles, C. L., Horne, B. G., Jabri, M. A.: Learning with product units. *Advances in Neural Information Processing Systems 7*, MIT Press, (1995), pp. 537-544.
- [7] Saito, K., Nakano, R.: Extracting regression rules from neural networks. *Neural Networks*, Vol. 15, no. 10, (2002), pp. 1279-1288
- [8] Angeline P. J., Saunders, G. M., Pollack, J. B.: An evolutionary algorithm that constructs recurrent neural networks. *IEEE Transactions on Neural Networks*, Vol. 5, no. 1, January (1994), pp. 54-65.
- [9] Martínez, A., Martínez, F., Hervás, C., García, N.: Model Fitting by evolutionary computation using product units. *Neural Networks*, (2004) (Submitted).
- [10] Fukunaga, K.: *Introduction to Statistical Pattern Recognition*. Academic Press, (1990).
- [11] Friedman, J., Multivariate adaptive regression splines (with discussion). *Ann. Stat.*, Vol. 19, (1991), pp. 1-41.
- [12] Carney, J., Cunningham, P.: Tuning diversity in bagged ensembles. *Int. J. Neural Systems*, Vol. 10, no. 4, (2000), pp. 267-279.
- [13] Lee, W. M., Lim, Ch. P., Yuen, K. K., Lo, S. M.: A hybrid neural network model for noisy data regression. *IEEE Transactions on Systems, Man and Cybernetics, Part B*, Vol. 34, no. 2, (2004), pp. 951-960.
- [14] Kosinski, W., Weigl, M.: Mapping neural networks and fuzzy inference systems for approximation of multivariate function. In Kacki E., (eds.): *System Modelling Control, Artificial Neural networks and Their Applications*, Vol.3, Lodz, Poland, May (1995), pp. 60-65.
- [15] Jankowski, N.: Approximation with RBF-type neural networks using flexible local and semi-local transfer functions. In 4th Conference on Neural Networks and Their Applications, Zakopane, Poland, May (1999), pp. 77-82.

# Lamarckian Clonal Selection Algorithm Based Function Optimization

Wuhong He, Haifeng Du, Licheng Jiao, and Jing Li

Institute of Intelligent Information Processing and  
National Key Lab of Radar Signal Processing, Xidian University,  
710071, Xi'an, China  
hewuhong@163.com

**Abstract.** Based on Lamarckism and Immune Clonal Selection Theory, Lamarckian Clonal Selection Algorithm (LCSA) is proposed in this paper. In the novel algorithm, the idea that Lamarckian evolution described how organism can evolve through learning, namely the point of “Gain and Convey” is applied, then this kind of learning mechanism is introduced into Standard Clonal Selection Algorithm (SCSA). Through the experimental results of optimizing complex multimodal functions, compared with SCSA and the relevant evolutionary algorithm, LCSA is more robust and has better convergence.

## 1 Introduction

Prior to Charles Darwin’s evolution theory of natural selection, Jean Baptiste Lamarck proposed that characteristics could be acquired and passed on to their offspring during an organism’s life[1][2][3], which means the experiences adapting to the environment can be inherited, and it has been known as Lamarckian evolutionary theory, namely Lamarckism. Although discredited in biology[4], Lamarckian evolution has been proven to be a effective concept for improving artificial intelligence algorithms [5].

In fact, Lamarckism has shown powerful performance in computing[6]. Bryant A Julstrom applied Lamarckian evolution to the 4-cycle problem, and the results showed that compared with Darwinian strategy, Lamarckian strategy sometimes had a better performance but sometimes earlier local search of Lamarckism would lead to premature convergence[7]. Darrell Whitley proposed a method of one iteration of steepest ascent used to local search, which is a good example of Lamarckism application[8]. Furthermore, some researchers John J. Grefenstette[9]. Evelina Lamma [10]. Shinichiro YOSHII[11]applied Lamarckism to some specific problems.

Learning mechanism is introduced into Standard Clonal Selection Algorithm (SCSA)[12], and Lamarckian Clonal Selection Algorithm (LCSA) is presented in this paper. Based on the idea “gain and convey”, LCSA makes full use of the experiences gained during the learning process to enhance the information communication among individuals, and improve the performance of algorithm. Relative function optimi-

zation experimental results shows that compared with SCSA and relative evolutionary algorithm, such as IGA[13], LCSA have better performance in global search and computing efficiency.

The rest of this paper is organized as follow. Section 2 describes the basic skeleton of SCSA as well as some problems in it. Section 3 illustrates LCSA and its implementation steps. Section 4 discusses the experimental results of LCSA, SCSA and relative evolutionary algorithm on 10 2-dimensional multimodal functions, and the impact of parameters on the performance of LCSA is analyzed. Finally, Section 5 concludes the paper with a short summary and gives a discussion for our future work.

## 2 Standard Clonal Selection Algorithm (SCSA)

Antibody Clonal Selection Theory is a very important concept in the immunology. Clone means reproduction or asexual propagation. A group of genetically identical cells descended from a single common ancestor, such as a bacterial colony whose members arose from a single original cell as a result of binary fission.

Based on the antibody clonal selection in immunology, we have proposed the SCSA [12] in Fig.1.

**Simple Clonal Selection Algorithm (SCSA)**  
**Step 1**  $k = 0$ ; Initialize antibody population  $A(0)$  at random and set the parameters of algorithm, then calculate the affinity of initialized population.  
**Step 2** According to the antibody affinity and the clonal size, get a new antibody population  $A(k)$  by Clonal Selection Operator.  
**Step 3**  $k = k + 1$ ; if the halted condition is satisfied, the iteration is terminated, otherwise, return to Step2.

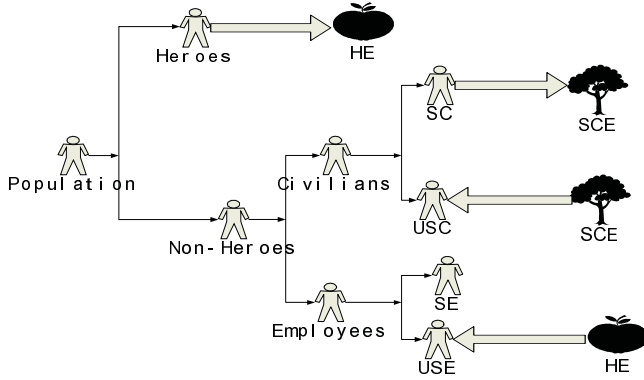
Fig. 1. The skeleton of SCSA

However, there is no learning mechanism in SCSA. As mutation operator is the main evolutionary operator, there is no information communication among individuals in population. Moreover, the probabilities of new good solution and bad can be generated by mutation operator are equal, thus SCSA couldn't ensure the global search speed and efficiency. By introducing learning mechanism into SCSA, evolution and learning are combined well and supplement each other, with the unification between the time and space and between the genotype and phenotype, thus evolving the population under the guide of learning and making algorithm find the global optima fast and efficiently.

## 3 The Skeleton of LCSA

According to the fitness distribution in population, several subpopulations are formed in Fig. 2. In general, the whole candidate population is divided into two parts: high-

fitness subpopulation Heroes and low-fitness subpopulation Non-Heroes. Furthermore, Non-Heroes can be divided into Employees evolving by conveying the Heroes' Experiences (HE) and Civilians evolving by themselves. After iterations, some individuals in Employees can go into Heroes, which are called Successful Employees (SE), while the others are named Unsuccessful Employees (USE). In the same naming way, Civilians can be divided into Successful Civilians (SC) and Unsuccessful Civilians (USC), and USC can get Successful Civilians Experiences (SCE) from SC.



**Fig. 2.** Several subpopulations are formed according to the fitness distribution in population

Based on the division of population, the main steps of LCSA in this paper is:

**Step 1. Initialize:** Set terminated condition of algorithm, mutation probability  $p_m$ , clonal size  $n_c$ , population size  $n$ , the coding length  $l$  and generate an initial binary population randomly, and the generation number  $k=0$ .

**Step 2. Calculate the antibody affinity**

**Step 3. Check the halted condition of algorithm:** If the condition is met, the algorithm terminates, otherwise continue.

**Step 4. Divide the population into several parts  $T_d^C$ :** According to the calculated affinity in **Step2**, the population is divided as  $[H(k), C(k), E(k)]^T = T_d^C(A(k))$ , where, Heroes is  $H(k)$  with the size  $h_n$ ; Civilians is  $C(k)$  with the size  $c_n$ ; Employees  $E(k)$  with the size  $e_n$ , moreover,

$$h_n + c_n + e_n = n \quad (1)$$

In practice, the size of Heroes is got by selecting individuals one tenth high fitness population, and the others are Civilians and Employees. For example, if  $n$  is 50, and we can select 5 individuals with highest fitness in the candidate population to construct Heroes.

**Step 5. Get Heroes' Experiences (HE)**  $T_{HE}^C : \mathbf{H}_E = T_{HE}^C[\mathbf{H}(k)]^T$ .

Where,  $\mathbf{H}_E = s_r \times \mathbf{H}_{h_i}(k) \quad 1 \leq h_i \leq h_n$ , and  $s_r$  is a randomly generated binary string with the length  $l$ :

$$s_r = \{r_1, r_2, \dots, r_L\} \quad \text{and } r_i \in \{0, 1\} \quad 1 \leq i \leq l \quad (2)$$

**Step 6. Clonal operating**  $T_c^C$ : Define

$$\mathbf{Y}(k) = T_c^C(\mathbf{A}(k)) = [T_c^C(\mathbf{a}_1(k)), T_c^C(\mathbf{a}_2(k)), \dots, T_c^C(\mathbf{a}_n(k))]^T \quad (3)$$

Where,  $T_c^C(\mathbf{a}_i(k)) = \mathbf{I}_i \times \mathbf{a}_i(k)$ ,  $i = 1, 2, \dots, n$ ,  $\mathbf{I}_i$  is a  $q_i$  dimension row vector with element 1, which is called the  $q_i$  clone of antibody  $\mathbf{a}_i$ . After clone, the population becomes:

$$\mathbf{Y}(k) = \{\mathbf{Y}_1(k), \mathbf{Y}_2(k), \dots, \mathbf{Y}_n(k)\} \quad (4)$$

**Step 7. Mutation operating**  $T_m^C$ : According to the mutation probability  $p_m$ , the cloned antibody populations are mutated,  $\mathbf{Z}(k) = T_m^C(\mathbf{Y}(k))$ .

$$\mathbf{Z}_i(k) = \{\mathbf{z}_{ij}(k)\} = \{T_m^C(\mathbf{y}_{ij}(k))\} = \{(random \leq p_m) - (-1)^{1-(random \leq p_m)} \mathbf{y}_{ij}(k)\} \quad (5)$$

**Step 8. Clonal selection operating**  $T_s^C$ :  $\forall i = 1, 2, \dots, n$ , if there are mutated antibodies.

$$\mathbf{b}_i(k) = \max\{\mathbf{Z}_i(k)\} = \{\mathbf{Z}_{ij}(k) \mid \max f(\mathbf{Z}_{ij}(k)) \quad j = 1, 2, \dots, q_i\} \quad (6)$$

the probability of  $\mathbf{b}_i(k)$  replacing  $\mathbf{a}'_i(k)$  is:

$$p_s^k(\mathbf{a}'_i(k) = \mathbf{b}_i(k)) = \begin{cases} 1 & \text{when } f(\mathbf{a}'_i(k)) < f(\mathbf{b}_i(k)) \\ 0 & \text{when } f(\mathbf{a}'_i(k)) \geq f(\mathbf{b}_i(k)) \end{cases} \quad (7)$$

**Step 9. Gain the Successful Civilians' Experience (SCE)**  $T_{SCE}^C : \mathbf{C}_{SE} = T_{SCE}^C[\mathbf{C}_S(k)]^T$ .

According to (9) some individuals in  $\mathbf{C}(k)$  and  $\mathbf{E}(k)$  can go into Heroes, which are called Successful Civilians (SC)  $\mathbf{C}_S(k)$  and Successful Employees (SE)  $\mathbf{E}_S(k)$  relatively. Others are named Unsuccessful Civilians (USC)  $\mathbf{C}_{US}(k)$  and Unsuccessful Employees (USE)  $\mathbf{E}_{US}(k)$ , then  $\mathbf{C}_{SE} = \mathbf{s}'_r \times \mathbf{C}_{MS}(k)$ , and  $\mathbf{C}_{MS}(k) = \max(f(\mathbf{C}_S(k)))$ .

**Step 10. Conveying HE and SCE**  $T_T^C : \mathbf{B}'(k) = T_T^C[\mathbf{A}'(k)]^T$ . Convey HE and SCE to  $\mathbf{E}_{US}(k)$  and  $\mathbf{C}_{US}(k)$  respectively,  $\mathbf{I}'_i$  is a row vector with the length  $l$ , then there is:

$$\mathbf{E}'_{US}(k) = (\mathbf{I}'_i - \mathbf{s}'_r) \times \mathbf{E}_{US}(k) + \mathbf{H}_E \quad (8)$$

$$\mathbf{C}'_{US}(k) = (\mathbf{I}'_i - \mathbf{s}'_r) \times \mathbf{C}_{US}(k) + \mathbf{C}_{SE} \quad (9)$$

From the algorithm steps above, note that Heroes with high fitness dominate the whole population and guide the evolution direction of the whole candidate population.

During the process of learning, HE is the most important factor; on the other hand, SCE from a small population is a good helper of HE, which also should be noticed.

## 4 Validation of LCSA

### 4.1 Experimental Results

In order to prove the effect and efficiency of proposed algorithm, the similar type of evolutionary algorithm Immune Genetic Algorithm (IGA)[13] with prior knowledge is introduced. We regard the global hits(the percentage of the global maximum found) as the primary objective achieved by the algorithm and function evaluations as the second objective[14]. For 10 functions[15], the parameters of SCSA, LCSA are set as: population size  $n$  is 50, the clonal size 5, coding length 40, the maximum number of iteration 200, and the required precision 0.01. For comparisons, the parameters of IGA are fixed as: population size  $n$  is 250, the others are set the same as two algorithms above. The statistical results of three algorithms out of 20 runs respectively are summarized in Table1. For  $f_3, f_4, f_5, f_6$ , the evolutionary process of three algorithms are shown in Fig.3.

**Table 1.** The results of IGA, SCSA and LCSA on  $f_1$ -  $f_{10}$ . “G.hit” is the global hits. “Eval” is the number of average evaluations out of 20 runs

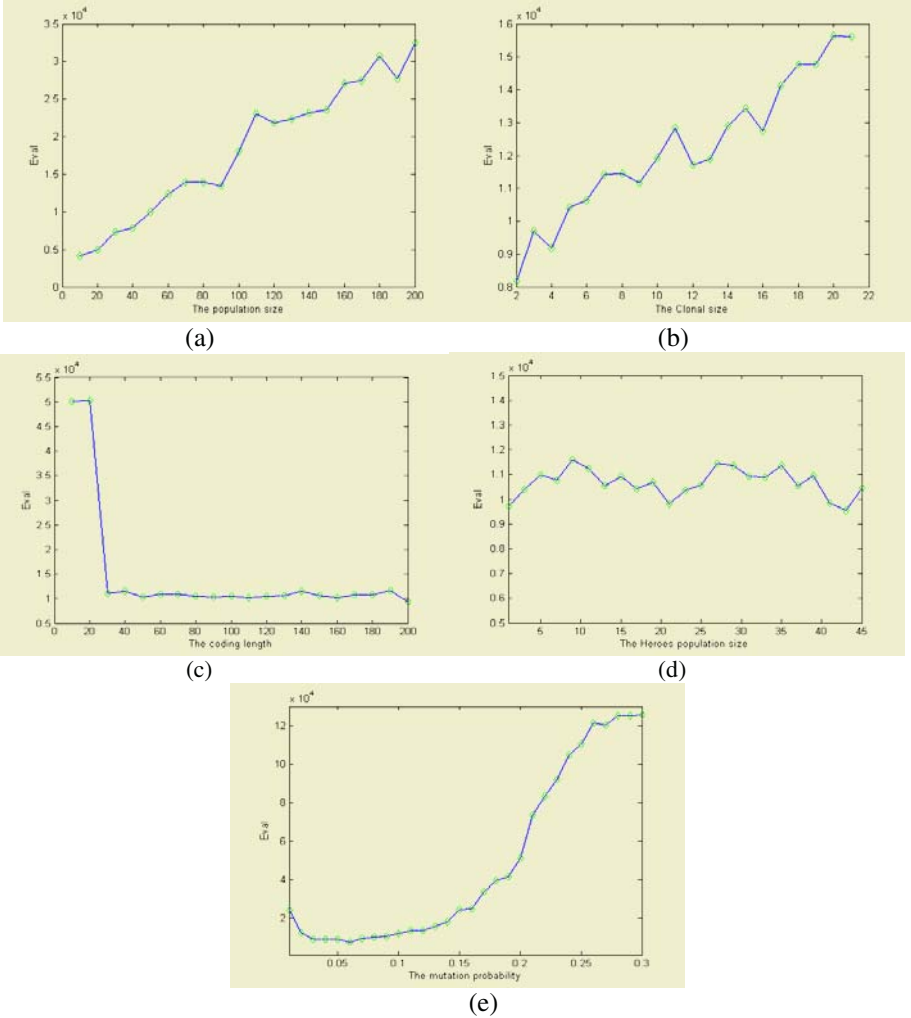
| Prob     | IGA    |               | SCSA   |             | LCSA         |               |
|----------|--------|---------------|--------|-------------|--------------|---------------|
|          | G. hit | Eval          | G. hit | Eval        | G. hit       | Eval          |
| $f_1$    | 16/20  | 12504         | 20/20  | 2315        | <b>20/20</b> | <b>1605.8</b> |
| $f_2$    | 20/20  | <b>1539.2</b> | 20/20  | 1805        | <b>20/20</b> | 1700          |
| $f_3$    | 20/20  | 11479         | 20/20  | 13385       | <b>20/20</b> | <b>10915</b>  |
| $f_4$    | 14/20  | 18756         | 19/20  | 12035       | <b>20/20</b> | <b>8720.9</b> |
| $f_5$    | 18/20  | 7044.1        | 20/20  | 3335        | <b>20/20</b> | <b>2797.8</b> |
| $f_6$    | 20/20  | <b>15794</b>  | 19/20  | 24050       | <b>20/20</b> | 19927         |
| $f_7$    | 18/20  | 11180         | 20/20  | <b>8015</b> | <b>20/20</b> | 10340         |
| $f_8$    | 20/20  | <b>2372.9</b> | 20/20  | 6155        | <b>20/20</b> | 2822.9        |
| $f_9$    | 14/20  | 18022         | 20/20  | 4955        | <b>20/20</b> | <b>3952.3</b> |
| $f_{10}$ | 13/20  | 24624         | 20/20  | 10970       | <b>20/20</b> | <b>5659.1</b> |

And it is indicated from Table 1 that for all test functions, LCSA can find the global optima out 20 runs, namely the on the primary principle “G.hit” is 20/20, which shows its good robustness. For  $f_1, f_4, f_5, f_7, f_9$  and  $f_{10}$ , IGA can’t find the global solution. For  $f_4, f_6$ , SCSA can’t achieve the global hits 20/20. These indicate LCSA outperforms SCSA and IGA with respect of global search. In terms of the second objective “Eval”, for functions  $f_2, f_6, f_7$  and  $f_8$  LCSA need a little more computer cost than SCSA and IGA, but for the others, the computer cost of LCSA outperforms is much less that two of them. Those are caused by the loss of diversity in the later evolutionary stage in LCSA. Finding better techniques to improve this weakness will be our future research.

### 4.2 The Analysis of Parameters

In LCSA, the main parameters are the population size  $n$ , the clonal size  $n_c$ , the coding length  $l$ , the Civilians size  $c_n$ , Employees size  $e_n$  and mutation probability  $p_m$ . To test

the impact of each parameter on the global convergence (Global hits) and the number of function evaluations (Eval) of our algorithm, 7 parameters are generally fixed as  $n=50$ ,  $n_c=5$ ,  $l=40$ ,  $h_n=5$ ,  $p_m=0.025$ ,  $c_n$  and  $e_n$  vary randomly in range  $[2, n-h_n-2]$  and  $[2, n-h_n-2]$  respectively, and the sum of  $c_n$  and  $e_n$  is 45. Since  $c_n$  and  $e_n$  are random numbers, actually there are only 5 parameters here. In experiment, we let one of them changeable, and set 4 of them as above fixed values. Taking  $f_3$  for example, the the number of function evaluations (“Eval”) are acquired out of 20 runs for each value of parameter.



**Fig. 3.** The main parameters population size  $n$ , clonal size  $n_c$ , coding length  $l$ , Heroes size  $h_n$  and mutation probability  $p_m$ 's impacts on the global search and numbers of function evaluations of LCSA, which are shown respectively in (a), (b), (c), (d) and (e). The horizontal axis is the tested parameter values, and the vertical axis is the average numbers of evaluation function out of 20 runs

**A) Influence of Population Size  $n$** 

Sample  $n$  by the interval of 10 from 10 to 200. It is shown in Fig.3(a) that in case that  $n=10$ , "Eval" is the least, and with the increase of  $n$ , it increases and the convergence speed slower, but relevant number of iteration becomes smaller. Those suggest that some candidate solutions that contribute less to generating the optimal solutions due to the increase of  $n$ .

**B) Influence of Clonal Size  $n_c$** 

Sample  $n_c$  by the interval of 1 from 2 to 21 and the result shown in Fig.3(b) indicates "Eval" is the least when  $n_c=4$ . While with  $n_c$ 's increase, the convergence speed is slower, and the relative number of iteration becomes smaller. It indicates that after  $n_c$  increases to some level, some candidate solutions contributing less to generating the optimal solutions, which is similar to the influence of  $n$  in term of this point.

**C) Influence of Coding Length  $l$** 

Sample  $l$  by interval of 10 from 10 to 100, from the result shown in Fig.3(c),  $l$  lying between 30 and 200 has a little influence on "Eval". In addition, it should be note that with bigger coding length, the computer cost also gets more during decoding process.

**D) Influence of Heroes Size  $h_n$** 

Sample  $h_n$  by interval of 2 from 1 to 45. It is implies from Fig.3(d) that  $h_n$  between 1 and 45 influence on "Eval" is not very obvious, and the values are about 10000, which is mainly caused by the random value of  $c_n$  and  $e_n$ , thus making the learning active and showing the advantage of learning to avoid the similarity between gaining and conveying of experiences.

**E) Influence of Mutation Probability  $p_m$** 

For the analysis, sample  $p_m$  by interval 0.01 from 0.01 and 0.3, but not set it as the reciprocal of coding length  $l$  in algorithm. It is suggested from Fig.3(e) that "Eval" doesn't change a lot in the range of 0.01 and 0.1 which is a reasonable interval. But beyond this interval, bigger  $p_m$  destroys the current population solution, thus making the evolution stagnate and "Eval" bigger.

On the analysis of parameters, the parameters in LCSA only can influence the convergence speed, but a little the global convergence, which shows good robustness of algorithm. It should be noted that those parameters sets are not the best options, but we still get good global performances.

## 5 Conclusions

In this paper, based on immune clonal selection theory and Lamarckism, Lamarckian Clonal Selection Algorithm (LCSA) is presented. In detail, according to the fitness distribution in population, several subpopulations and experiences are formed and LCSA is proposed. The LCSA improves some of the drawbacks of the SCSA. The comparisons of the algorithms' performances are made considering the global hits, the function evaluations. The LCSA can provide more efficient performance with better reliability on most 2-D multimodal functions in test. However, there are some experimental results raising the issue that LCSA may not have enough search diversity



and influence the convergence speed during the later evolutionary stage. Hence, using new techniques to make LCSA better will be included in our future work.

## Acknowledgment

The work was sponsored by the National Natural Science Foundation of China (No: 60133010 ) grants and the Natural Science Foundation of Shaanxi (No: 2004F29).

## References

1. Dawkins, R.: *The Blind Watchmaker*. Norton, 1996
2. Cochrane, E.: *Viva Lamarck: A Brief History of the Inheritance of Acquired Characteristics*. 1997
3. Gould, S.J.: *The panda's Thumb*. New York, Norton, 1980
4. Ross, B.J.: A Lamarckian Evolution Strategy for Genetic Algorithms. In: L. Chambers (ed.) *Practical Handbook of Genetic Algorithm*, Vol. 3, chapter 1, CRC Press (1999) 1-16
5. Mitchell, T.M.: *Machine Learning*, McGraw Hill, 1997
6. Hart, W.E., Belew, R.K.: Optimization with genetic algorithms hybrids that use local search. In: Belew, R.K., Mitchell, M. (eds.): *Adaptive Individuals in Evolving Populations: Models and Algorithms*, Vol. 26, chapter 27, SFI Studies in the Sciences of Complexity. (1996) 483--496
7. Julstrom, B.A.: Comparing Darwinian, Baldwinian, and Lamarckian Search in a Genetic Algorithm for the 4-Cycle Problem. *Late Breaking Papers at the 1999 Genetic and Evolutionary Computation Conference*. 1999
8. Whitley, D., Gordon, V.S., Mathias, K.: Lamarckian Evolution, the Baldwin Effect and Function Optimization. *Proceeding of the International Conference on Evolutionary Computation. The Third Conference on Parallel Problem Solving from Nature: Parallel Problem Solving from Nature*, (1994) 6-15
9. Grefenstette, J.J.: Lamarckian Learning in Multi-agent Environments. *Proceedings of the 4th International Conference on Genetic Algorithms*, (1991) 303-310
10. Lamma, E., Riguzzi, F., Pereira, L.M.: Belief Revision via Lamarckian Evolution. *Technical Report DEIS-LIA-00-004, University of Bologna, LIA Series. Italy, No.44, 2000*
11. Yoshii, S., Suzuki, K., Kakazu, Y.: Lamarckian GA with Genetic Supervision. In *ICEC'95*. 1995
12. Du, H.F., Jiao, L.C., Wang, S.A.: Clonal Operator and Antibody Clone Algorithms. In: Shichao, Z., Qiang, Y., Chengqi, Z. (eds.): *Proceedings of the First International Conference on Machine Learning and Cybernetics*. IEEE, Beijing (2002) 506–510
13. Jiao, L.C., Wang L.: A Novel Genetic Algorithm based on Immunity. *IEEE Trans. Systems, Man and Cybernetics, Part A*. Vol.30, No.5 (2000) 552–561
14. Liang, K.H., Yao, X., Newton, C.: Evolutionary Search of Approximated N-Dimensional Landscapes. *International Journal of Knowledge-Based Intelligent Engineering Systems*. Vol.4, No.5 (2000) 172-183
15. Syrjakow, M., Szerbicka, H.: Efficient parameter optimization based on combination of direct global and local search methods. In Davis, L.D., Jong, K.D., Vose, M.D., Whitley, D. (eds.): *Evolutionary Algorithms, IMA program on mathematics in high-performance computing*. Springer-Verlag, New York (1999) 227-249

# Artificial Neural Networks Based on Brain Circuits Behaviour and Genetic Algorithms

Ana Porto<sup>1</sup>, Alejandro Pazos<sup>1</sup>, and Alfonso Araque<sup>2</sup>

<sup>1</sup> Department of Information and Communications Technologies, University of A Coruña,  
15071 Campus de Elviña, A Coruña, Spain  
{anuska, apazos}@udc.es

<sup>2</sup> Laboratorio de Electrofisiología, Instituto de Neurobiología Ramón y Cajal,  
CSIC 28002 Madrid, Spain  
araque@cajal.csic.es

**Abstract.** Once the behaviour of particular brain circuits has been analyzed, we have added up some of these patterns to Artificial Neural Networks; thus a new hybrid learning method has emerged. In order to find the best solution to a given problem, this method combines the use of Genetic Algorithms with particular changes to connection weights based in the behaviour observed in the brain circuits analyzed. The design and implementation of this combination is shown in feed-forward multilayer artificial neural networks, specifically created to solve a simple problem. We also illustrate the benefits obtained with these new nets from a comparison with previous results achieved by the optimal Artificial Neural Networks used so far for solving the same problem.

## 1 Introduction

In order to create Artificial Neural Networks (ANN) which emulate the brain and try to take maximum advantage of its functionality, we need a deep knowledge into its structure and function.

Computational models of cells involved in learning and memory processes have been analyzed [3] [4] by making use of actual computer capabilities and thanks to the efficacy of research into computer modeling, which allows a better understanding of the structure and function of certain brain cells. The findings made with the help of these models have been used for trying to get advances in Computer Science and, more specifically, in Artificial Intelligence.

Considering the behaviour observed in brain cells, we have decided to create ANN which integrate part of this behaviour. The objective was to check if these new ANN, which would mainly emulate brain circuits, were more efficient when solving simple problems actually solved by means of multilayer ANN trained with Genetic Algorithms (GA). A learning hybrid method has been designed. Such method looks for optimal ANN connection weights by combining, in a first learning phase, weight value changes following rules based in brain circuits behaviour and, in a second phase, training by means of GA. Details of such model can be seen further on.

The use of GA in the second learning phase has permitted us to incorporate this type of brain-observed behaviour in the first phase. Such behaviour has not been considered so far because of the difficulty other learning methods have to train ANN with more complex information processing elements than those traditionally used.

In the present work, ANN design has focused on fully connected feed-forward multilayer ANN, without backpropagation and without lateral connections. These ANN are generally used to solve simple problems of classification and pattern recognition [5]. We think it is more adequate to start using the latest research about information processing into ANN for classification or recognition, because synaptic plasticity in brain connections is imitated. We have decided to apply this method into simple ANN; however, in the future, we hope to develop models with increasingly complex ANN. Thus, we look for a starting point that stimulates further research.

## 2 ANN Development

ANN development follows the satisfactory steps usually used in the construction and use of Connectionist Systems: design, training, test and running.

### 2.1 Design

We decided to develop feed-forward multilayer ANN fully connected without backpropagation and without lateral connections. They were simple ANN oriented to classification and pattern recognition.

### 2.2 Training

The Hybrid learning method combines a non-supervised learning method, based on brain cells behaviour (stage one), with a supervised learning method using GA (stage two). The former has already been modeled with NEURON [1] both in D. LeRay [3] and in A. Porto [4]. Given that it is necessary to have a population of GA individuals, a set of individuals was created in the first phase. Every individual is composed of the same values as those representing all the ANN connections weights. Every random set of weights values forms a different GA individual.

In the first phase, we study the ANN functioning with all the individuals. Every individual is modified every time the training pattern is presented to the ANN, following rules based on the neuron activity generated by that pattern. For each individual, every in-pattern is presented to the ANN during a number of established iterations. These iterations allow the application of rules based on cellular behaviour and constitute a pattern cycle. During each iteration, connections are modified according to rules depending of neural activity. We can establish the number of iteration we want. When the pattern cycle has finished, the ANN error for that pattern is calculated. It will be the difference between obtained out and expected out. ANN error for each pattern is stored. Later on, when all training patterns have been presented to the ANN, mean square error (MSE) of that individual is calculated. MSE

gives a measure relative to the amount of patterns supplied to ANN and permits comparing the error between different architectures and trial tests.

This is the process followed for all the individuals. This phase is non supervised learning because when the connection weights are modified the error is not taken into account.

The second learning phase (supervised learning) consists of applying GA to the individuals according to their MSE, which was stored in the first phase. The GA crosses, mutes and selects new individuals until to get it gets the individual with minor error. So as to develop this AG training system we will use the AG specifications following the J. Holland specifications [2].

### 2.3 Test and Running

Once the training is finished and the best individual is available it is necessary to test ANN generalization capability. If this capability is right ANN will be ready for being used.

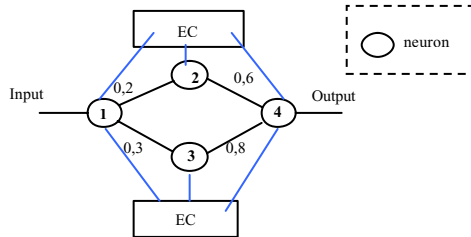


Fig. 1. ANN with CE

We want to emphasize that at this stage, the brain behaviour introduced in the non-supervised learning phase will be always applied. Thus, this behaviour that has been incorporated to the ANN as control elements (fig. 1) is present during all the ANN life stages and participates directly in information processing.

## 3 Tests Performed

### 3.1 Hardware and Software

With the purpose of building a design, training and test system for the ANN we have created a software tool that allows us to compare training only by means of GA with this new ANN, which uses hybrid learning method. This tool is based in an existing tool designed by J. Rabuñal [5] and has been modified in order to incorporate the new functioning already explained above.

The implementation was made with Borlan DELPHI and Visual C++. The tests were run in an AMD Athlon XP 2000 PC, with 256 MB of RAM and Windows XP OS.

### 3.2 Problem to Solve: MULTIPLEXOR

We decided to make the test over a known multilayer ANN that solves a classification problem: the simulation of an electronic device called MultipleXor (MUX) with four inputs and one output (fig. 2).

This device behaviour is explained in [5] The results obtained with our new nets were compared with the results obtained with ANN trained only with GA.

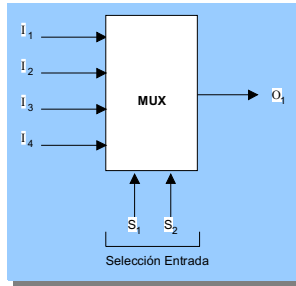


Fig. 2. MUX device

Table 1 shows all possible input combinations to the MUX and the desired output. It has been denoted with an X the fact that the binary value will be indifferent in some cases. The fifty eight first combinations comprise the training set and the last six the test set.

The structure and functioning of MUX device is suitable for implementing a multilayer ANN because there is not any factor that depends on time or on other values which require the use of other architecture types.

Table 1. Function MUX values

| I1 | I2 | I3 | I4 | S1 | S2 | O1 |
|----|----|----|----|----|----|----|
| 0  | X  | X  | X  | 0  | 0  | 0  |
| 1  | X  | X  | X  | 0  | 0  | 1  |
| X  | 0  | X  | X  | 0  | 1  | 0  |
| X  | 1  | X  | X  | 0  | 1  | 1  |
| X  | X  | 0  | X  | 1  | 0  | 0  |
| X  | X  | 1  | X  | 1  | 0  | 1  |
| X  | X  | X  | 0  | 1  | 1  | 0  |
| X  | X  | X  | 1  | 1  | 1  | 1  |

### 3.3 New ANN Architectures

Once input and output patterns for training and test were selected, we established the ANN parameters which allowed us to define its structure and solve the problem. These parameters are: maximum value for the weights (1), neurons activation

function (threshold: 0.5), number of layers and number of neurons per layer. In order to make an adequate comparison we got this parameters from J. Rabuñal [5].

It is a very simple ANN with a three-layer architecture: six neurons in the input layer (four I inputs plus two S inputs in fig. 2), four neurons in the shadow layer and one neuron in the output layer (O in fig. 2).

As regards the modifications over the connection weights, we selected a set from all the different possibilities analyzed in computational models [4]. These modifications consist of a percentage increment or decrement in the weights according to neuronal activity. Moreover, we made distinct trial depending on the number of iterations performed with every pattern (pattern cycle). The possibilities selected will be detailed in the tests.

### 3.4 GA Characteristics

The tests were carried out by keeping the same populations of individuals (three populations) and the same random seed (seed=0) which originates the selection of the individuals to crossover and mute. In this way, we guarantee the validity of the comparison between the different possibilities for applying the modifications following brain phenomena and the comparison between existing ANN and the new ANN proposed in this paper.

The GA parameters selected for all the tests in both ANN were the following: population size of 100 individuals; Montecarlo technique in order to select the individuals; Darwinian substitution method; crossover rate 90% and mutations rate 10%.

In all tests we wanted to record the number of the generations every ANN needs for being trained, i.e. generations until MSE is zero or near zero.

The training simulation was the unique process running in the computer during the test.

## 4 Results

We show results per population and the average of the three populations, both in the case of existing ANN and in case of the new ANN. Four thousand generations were established in all the simulations.

### 4.1 ANN Training Only with GA

Table 2 shows results per population.

**Table 2.** Simulation results

| Population | MSE    | Seconds | Generations |
|------------|--------|---------|-------------|
| 1          | 0,0682 | 210     | 4000        |
| 2          | 0,0517 | 222     | 4000        |
| 3          | 0,0517 | 220     | 4000        |
| Average:   | 0,0571 | 217,3   | 4000        |

## 4.2 ANN Training with Hybrid Method

The modifications carried out by the control elements (CE) were the following:

- 25% increase in the connection weights from activated neurons in two or three consecutive iterations.
- 50% decrease in the connections weights from inactivated neurons during two or three consecutive iterations.

We considered that if a neuron is active during these defined iterations, the CE that controls the activated neuron layer also activates and influences (potentiates) the connections between this neuron and its postsynaptic neurons. Likewise, if the neuron activity is not enough to reach the percentage of activation established, the CE is not excited by that neuron and the connections leaving it are weakened.

These two or three consecutive iterations are aimed at representing great neuronal activity (high frequency stimulation) in the real brain. We found that these percentages provide satisfactory results after trying tests with several combinations of possibilities. The biological observations justified this choice because if the increment caused by reinforcement is minor than the decrement when the activity is not constant, only those neuron connections with continuous activity will remain reinforced. Other percentages will be selected in the future with the purpose of improving these results.

**Table 3.** Results of ANN with hybrid learning

| Iterations | Iterations per pattern | MSE    | Time simulation (s.) | Generations |
|------------|------------------------|--------|----------------------|-------------|
| 2          | 2                      | 0,0517 | 218                  | 4000        |
|            | 6                      | 0,0172 | 578                  | 4000        |
|            | 8                      | 0,0344 | 660                  | 4000        |
|            | 10                     | 0,0862 | 981                  | 4000        |
| 3          | 2                      | 0,0517 | 191                  | 4000        |
|            | 6                      | 0,0172 | 515                  | 4000        |
|            | 8                      | 0,0172 | 655                  | 4000        |
|            | 10                     | 0,0862 | 873                  | 4000        |

The number of iterations per pattern was estimated in four, six and eight. As table 3 shows, before the design of trial sets we verified that if we considered less than four iterations we did not observe differences with regard to training with only one iteration per pattern (rows 1 and 5 in table 3). Moreover, if we considered more than eight iterations, the results were not better than before and simulation runtime grew a lot (rows 4 and 8 in table 3).

Every neuron can activate more or fewer times depending on its job in the ANN, conditioned by its inputs, like it happens in the brain. It is not possible to control how many times a neuron is going to activate, but a neuron is allowed to be activated a certain number of iterations.

Table 4 shows detailed results of tests performed with ANN training by means of the hybrid method keeping each population constant. The average results with the three populations are presented in table 5.

**Table 4.** Results of the three populations in function of neuronal activity

| Population | Active Iteracions | Iterations per pattern | MSE    | Seconds | Generations |
|------------|-------------------|------------------------|--------|---------|-------------|
| 1          | 2                 | 4                      | 0,0172 | 372     | 4000        |
|            |                   | 6                      | 0,0866 | 578     | 4000        |
|            |                   | 8                      | 0,0344 | 660     | 4000        |
|            | 3                 | 4                      | 0,0172 | 369     | 4000        |
|            |                   | 6                      | 0,0172 | 510     | 4000        |
|            |                   | 8                      | 0      | 343     | 2094        |
| 2          | 2                 | 4                      | 0,0517 | 370     | 4000        |
|            |                   | 6                      | 0      | 207     | 1432        |
|            |                   | 8                      | 0,12   | 657     | 4000        |
|            | 3                 | 4                      | 0,0517 | 366     | 4000        |
|            |                   | 6                      | 0,0517 | 515     | 4000        |
|            |                   | 8                      | 0      | 245     | 1432        |
| 3          | 2                 | 4                      | 0      | 61      | 655         |
|            |                   | 6                      | 0,0172 | 573     | 4000        |
|            |                   | 8                      | 0      | 162     | 928         |
|            | 3                 | 4                      | 0      | 60      | 655         |
|            |                   | 6                      | 0      | 82      | 655         |
|            |                   | 8                      | 0,0172 | 655     | 4000        |

**Table 5.** Average results of the three population in function of neuronal activity

| Active Iteration | Iterations per pattern | MSE    | Seconds | Generations |
|------------------|------------------------|--------|---------|-------------|
| 2                | 4                      | 0,0229 | 266,6   | 2885        |
|                  | 6                      | 0,0346 | 452,6   | 3144        |
|                  | 8                      | 0,514  | 493     | 2976        |
| 3                | 4                      | 0,0229 | 265     | 2885        |
|                  | 6                      | 0,0229 | 369     | 2885        |
|                  | 8                      | 0,0057 | 414,3   | 2508        |

If we compare these results with results in table 2, we observe that in the case of the three populations, results are better.

Afterwards, the training test phase was run with the new ANN. The best two cases of table 4 were selected (second row of second population – 1432 generations and first row of third population – 655 generations). ANN training with hybrid method with optimum weights gave the adequate outputs.

## 5 Discussion

Given the great amount of possibilities of influence over connection weights based on brain circuits, the study of what happens by using random increases and decreases is justified by the fact that the way CE act is not known at all.

In all the tests we observed that only those neurons activated several times did potentiate their leaving connections. These conditions are a step forward in the



problem studied here, i.e. ANN training. Table 4 indicates that in the case of ANN trained with hybrid method the best results were reached when three neuronal activity iterations were considered, i.e. when more neuron activity was demanded for exciting CE. In some cases, for different combinations, results are the same because the same modifications in weights could occur.

The reason why these satisfactory results were achieved with the new ANN are attributed to what was expected when the hybrid method was designed. The individuals, once evaluated, are presented to the GA arranged by their MSE. The changes made to the weights during the first learning phase cause this resulting order to be different from an potential order obtained in case the first phase did not exist. This is so because their MSE varies according to this new method. The number of generations necessary for reaching zero error is in many cases smaller than in the case of the ANN which are only trained with GA. However, the simulation time is higher, but not to the extent of becoming critical for training.

## 6 Conclusions

The newly created ANN, trained using a hybrid method that combines brain behaviour and GA, manages to the functioning of the multilayer ANN studied.

We have created ANN that closely reproduce several aspects of the human central nervous system. This directly benefits Artificial Intelligence because it can improve information processing capabilities which would allow us to deal with another range of problems.

We have indirectly benefited Neuroscience because experiments with computational models that simulate brain circuits smooth the way for difficult experiments conducted in laboratory as well as provide new ideas for research.

## References

1. Hines, M.: The NEURON simulation program. In J. Skrzypek. Norwell (ed.) Neural Network Simulation Environments, MA: Kluwer, (1994) 147-163
2. Holland, J.H.: Adaptation in Natural and Artificial Systems. University of Michigan Press. Ann Arbor, MI, USA (1975)
3. LeRay, D., Fernández, D., Porto, A., Fuenzalida, M., Buño, W.: Heterosynaptic Metaplastic Regulation of Synaptic Efficacy in CA1 Pyramidal Neurons of Rat Hippocampus. Hippocampus (2004)
4. Porto, A.: Modelos Computacionales para optimizar el Aprendizaje y el Procesamiento de la Información en Sistemas Adaptativos: Redes Neurogliales Artificiales (RR.NG.AA.). Tesis Doctoral. Universidade da Coruña. A Coruña. (2004)
5. Rabuñal, J.: Entrenamiento de Redes de Neuronas Artificiales con Algoritmos Genéticos. Tesis de Licenciatura. Dep. Computación. Facultad de Informática. Universidade da Coruña (1998)

# Modeling Synaptic Transmission and Quantifying Information Transfer in the Granular Layer of the Cerebellum

Egidio D'Angelo<sup>1,2</sup>, Thierry Nieuws<sup>1</sup>, Michele Bezzi<sup>3</sup>,  
Angelo Arleo<sup>3</sup>, and Olivier J.-M.D. Coenen<sup>3</sup>

<sup>1</sup> Dept. of Cellular-Molecular Physiological and Pharmacological Sciences,  
Univ. of Pavia, and INFM, Via Forlanini

6, I-27100, Pavia, Italy

<sup>2</sup> Dept. of Functional and Evolutionary Biology, Univ. of Parma,  
Parco Area delle Scienze 11a, I-34100, Parma, Italy

<sup>3</sup> Neuroscience Group, Sony CSL, 6 rue Amyot, 75005 Paris, France

**Abstract.** Neurons communicate through spikes; their arrangement in different sequences generates the neural code. Spikes are transmitted between neurons via synapses; the mechanism underlying synaptic transmission involves numerous processes including neurotransmitter release and diffusion, postsynaptic receptor activation, and intrinsic electroresponsiveness. Based on available experimental data and theoretical considerations, we have developed a realistic model predicting the dynamics of neurotransmission at the mossy fiber - granule cell synapse of the cerebellum. The model permits systematic investigation of the multiple mechanisms regulating synaptic transmission and provides predictions on the role of the numerous factors driving synaptic plasticity. The model is also employed to quantify information transfer at the mossy fiber - granule cell synaptic relay. This work was funded in part by the EU SpikeForce project (IST-2001-35271 [www.spikeforce.org](http://www.spikeforce.org)).

## 1 Introduction

Neurons communicate through sequences of stereotyped pulses, called spikes or action potentials, which are transmitted between them at the synapses (Fig. 1). There, the presynaptic spike train is transformed and converted into a postsynaptic signal. The information content of these spike trains can be assessed by considering either the precise timing of action potentials or their average frequency [1, 2]. The former approach tends to be more efficient from the informative viewpoint because it captures the fine temporal structure of the neural signal (e.g., the interspike interval distribution). In addition, the temporal pattern of the spike train can affect the dynamics of the synaptic contacts, and hence the processing. For instance, short-term memory effects (i.e., short-term facilitation and depression) may regulate postsynaptic temporal summation in a time-dependent manner [3, 4]. Accordingly, to fully understand information

processing in neuronal assemblies it is useful to develop detailed models accounting for the main biological features. We also need theoretical tools that allow us to quantify the ability of the specific model to transmit information at different levels of resolutions, and to assess the robustness of this process. In this regard, information theory [5] has proved to be suitable for studying information processing in different brain areas [1]. Within this framework it is possible to quantify the amount of information that a given set of neural responses provides about a specific set of stimuli or a set of upstream neuron activities. Furthermore, Shannon information and similar quantities can be used to investigate the coding strategies or the contribution of spatial and temporal correlations to the information transmission [6, 7].

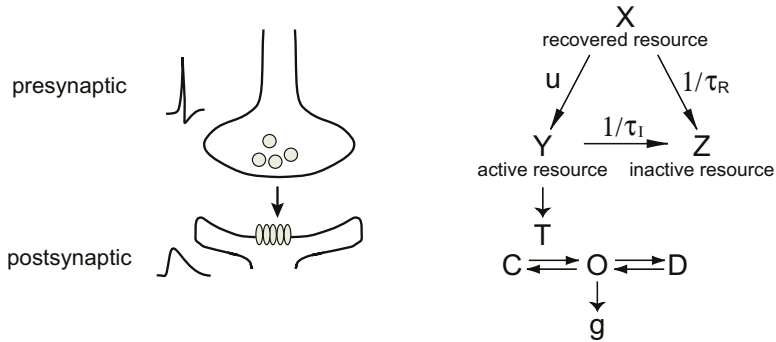
The cerebellum is one part of the brain responsible for the learning and the automatic execution of coordinated movements, particularly those too rapid for conscious feedback control. Besides its direct importance in clinical research (motor coordination diseases), there are several other motivations for focusing on this system. First, it permits a detailed experimental characterization at different levels: molecular, single cell, and neural population, both *in vitro* and *in vivo*. Second, it represents an excellent test bed to investigate how microscopic interactions at the single cell level can initiate complex collective behavior (cerebellar functions) at the population level. Third, from an application point of view, implementing biomimetic cerebellar models to control mobile robots or industrial processes may augment their ability to learn and coordinate their actions in complex contexts.

The cerebellar input layer is of particular interest: it is characterized by a huge number ( $10^{11}$ ) of tiny cells (granule cells) that, according to classical theories of Marr and Albus [8, 9], are able to encode afferent information into a sparse representation that facilitates discrimination of very similar inputs. In this study, we focus on the mossy fiber - granule cell synapse, which is the major site of plasticity in the cerebellum granular layer. The next section presents the main characteristics of this synapse. Section 3 introduces a detailed biophysical model of this system. Section 4 describes an information-theoretic approach for studying information processing at this site quantitatively.

## 2 Synaptic Transmission at the Cerebellar Granular Layer

Mossy fibers (MFs) are the primary afferents to the cerebellar cortex and convey multimodal sensory inputs to the granule cells (GCs). The MF-GC synaptic transmission constitutes the core of the granular layer computation and has complex temporal dynamics [10, 11] capable of regulating the input-output relationship via synaptic gain modulation [9, 12].

Neurophysiological data suggest that long-term potentiation (LTP) can enhance the probability of release of neurotransmitters at the MF-GC synapse [13], and that GCs tend to discharge in bursts *in vivo* [14]. At a finer scale, several factors can influence the relationship between neurotransmitter release and GC



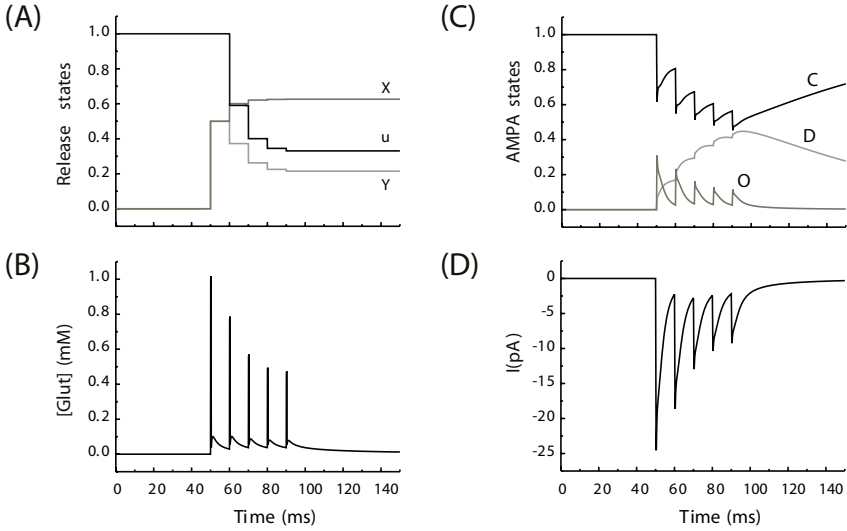
**Fig. 1.** Synaptic transmission and modeling. (Left) Synaptic processing transforming an incoming spike into a postsynaptic response. (Right) Schematic representation of the processes involved in determining the change of postsynaptic conductance ( $g$ ) and the generation of excitatory postsynaptic currents (EPSC).  $X$  denotes the transmitter resources available for release,  $Y$  is the amount of released transmitter, and  $Z$  represents the amount of recovered transmitter. The time constants of recovery of releasable transmitter ( $\tau_R$ ), facilitation ( $\tau_F$ ), and inactivation ( $\tau_I$ ), are indicated together with  $u$ , the probability of release.  $T$  denotes the glutamate concentration, while  $C$ ,  $O$ , and  $D$  indicate the postsynaptic receptor state ‘closed’, ‘open’, and ‘desensitize’, respectively

firing. The intense glutamate spillover observed in the cerebellar glomerulus, by protracting AMPA and NMDA receptor activation [15, 16, 17], generates opposing processes like receptor desensitization and temporal summation of excitatory postsynaptic potentials (EPSP) [18]. Moreover, postsynaptic voltage-dependent currents determine complex regulation of spike discharge [19, 20]. Because these factors interact in a complex non-linear manner, no firm statement can be given a priori on burst processing at the MF-GC synapse and its regulation during LTP. Hence, we developed a model of synaptic transmission at the MF-GC relay accounting for the interaction of these multiple effects.

### 3 A Model of the MF-GC Synaptic Transmission

Our aim was to conjugate fundamental aspects of neurotransmission derived from physiological recordings with a detailed reconstruction of postsynaptic electroresponsiveness. A model of the GC derived from our previous study [20] was updated based on recent experimental data. GCs are electrotonically compact [18, 21, 17], hence there is little need to simulate dendrites and a mono-compartmental structure was employed. The GC model includes four identical and independent synapses. The NEURON simulator [22] was employed to implement and validate the model.

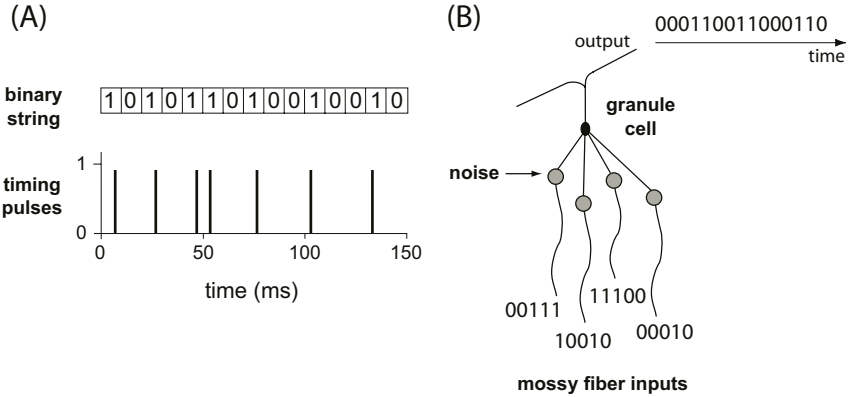
The simulation of a single excitatory postsynaptic current (EPSC) involves modeling the neurotransmitter release at the presynaptic site, the diffusion of the neurotransmitter within the synaptic cleft, and the postsynaptic receptor



**Fig. 2.** AMPA current parameters during a voltage-clamp simulation ( $-70\text{mV}$ ). A burst at 100 Hz stimulates the MF. (A) Evolution of the presynaptic variables  $X$ ,  $Y$ , and  $u$ . The decrease of  $u$  determines synaptic depression. (B) Diffusion protracts the glutamate waveform. (C) Evolution of postsynaptic receptor states. Note the decrease of open states and accumulation of desensitization. (D) Temporal summation of AMPA EPSCs

dynamics [23]. The situation becomes even more complicated when considering a multiple release-site synapse; this is due to the stochastic activation of different receptor clusters and to the diffusion of neurotransmitters between synaptic sites [24]. In the model, the state of the presynaptic terminal is computed according to a three-state scheme adapted from [25]. When a presynaptic spike arrives, a proportion  $u$  of the transmitter resource  $X$  is transferred into an amount of released transmitter  $Y$  (Fig. 1). Depletion of the resource  $X$  causes synaptic depression (another component of synaptic depression depends on postsynaptic receptor desensitization, see below). Synaptic facilitation is governed by the activity-dependence of the transmitter release  $u$ .

GC postsynaptic responses are generated through both direct release from active zones onto corresponding postsynaptic receptors and spillover of glutamate from neighboring releasing sites [16, 17]. In the model, the glutamate concentration  $T$  for AMPA receptors, which are located into the cleft, is obtained by combining a synaptic pulse ( $T_s$ ) with a diffusion wave ( $T_d$ ), while NMDA receptors are only activated by the diffusion wave  $T_d$ . The released glutamate acting on AMPA receptors is generated with a 1 mM - 0.3 ms squared pulse, which has been shown to approximate transmitter action in the cleft properly [23]. Diffusion is simulated with 2D Crank equation [26, 24, 27]. Glutamate binding to postsynaptic receptors activates kinetic schemes governed by microscopic first-order transitions, leading to the open state  $O(T)$ . The AMPA postsynaptic current is reproduced with a  $D = C = O$  scheme ( $D$  and  $C$  are the ‘desensitize’



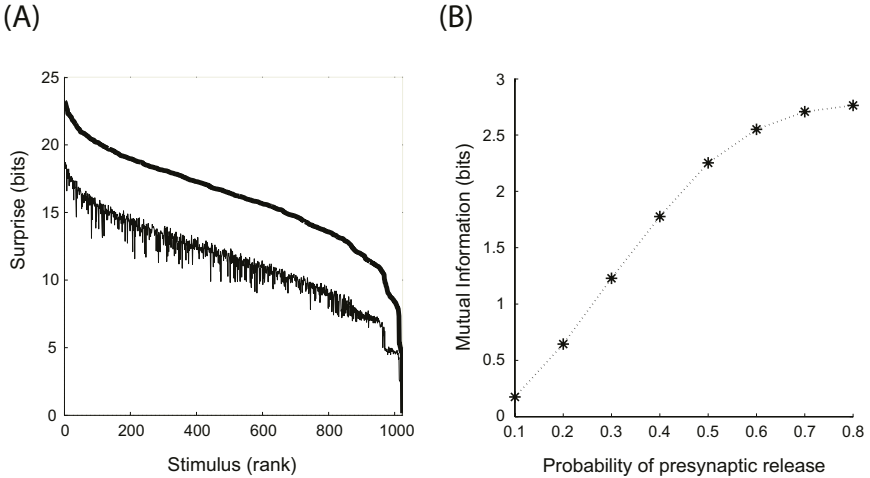
**Fig. 3.** (A) A spike train can be represented by a binary string of 0 and 1. The string depends on the precision chosen for time discretization. (B) A GC receives four MF inputs and generates an output spike train. The GC response is affected by the multiple voltage- and time-dependent mechanisms of its membrane. Noise is introduced mainly by stochastic vesicular release, which is explicitly modeled. Appropriate construction of the input and output strings permits MI calculation

and ‘closed’ state, respectively) [27], the NMDA current with a more complex scheme derived from [28]. It follows that we can compute the EPSC composed of AMPA (Fig. 2) and NMDA currents. Once coupled to the excitable mechanisms endowed in our previous GC model [20], the present system reproduces the main aspects of GC synaptic excitation [18].

## 4 Information Transmission at the MF-GC Synapse

Shannon mutual information (MI) [5] provides a natural mathematical framework to answer the question *how much* information is transmitted by the neural patterns. Our aim was to understand how information is transmitted by the GC through the MF-GC relays, and how it is affected by various factors related to the intrinsic organization of the circuit. The GC constitutes an ideal system for MI calculation since the number of possible inputs is very reduced compared to other brain cells due to the limited number of afferent MFs to each GC (4 on average). To represent the stochasticity of neurotransmission accurately we developed a stochastic version of the model presented above in which the neurotransmitter release was probabilistic (the release at individual sites was an *all-or-none* event determined once a random number between 0 and 1 passed a release probability threshold). This stochastic model was used for the mutual information computation.

In a typical simulation all spike trains were digitalized (Fig. 3A), and a controlled set of stimuli  $\mathcal{S}$  (each stimulus being formed by 4 input spike trains, Fig. 3B) was chosen. Then, we recorded the elicited neural responses  $r \in \mathcal{R}$  when one stimulus  $s \in \mathcal{S}$  was repeatedly presented with a known a priori proba-



**Fig. 4.** (A) Information theoretic *surprise* contained in the spike count (thin line) and in the binary string representation (thick line) - see text. Stimuli are ordered by their values of binary string surprise. (B) MI as a function of neurotransmitter probability of release. Simulation parameters: the stimulus set was composed by 1024 spike trains randomly drawn from a Poisson distribution (average firing rate 10 Hz, time bin 5 ms, length of the binary string 10 bins). The same stimulus was presented to the four MFs

bility  $p(s)$ . Once we collected all the data, we estimated the corresponding joint probabilities,  $p(r, s)$ , and the probability distribution of responses averaged over the stimuli,  $p(r)$ . The mutual information was computed using:

$$I(\mathcal{R}; \mathcal{S}) = \sum_{s \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(r, s) \log_2 \left[ \frac{p(r, s)}{p(r)p(s)} \right] \quad (1)$$

Shannon MI provides a quantitative measure of the averaged information transmitted through the synapse by a set of responses given a set of input spike trains (or vice-versa). We were also interested in identifying those stimuli that were best encoded by the GC. Thus, we computed the stimulus specific contribution to the MI (namely, the *surprise*  $I(s) = \sum_{r \in \mathcal{R}} p(r|s) \log_2 \frac{p(r|s)}{p(r)}$ ). This allowed us to find the most informative set of stimuli, and understand in which conditions the cell coding capability was optimized.

Previous results [29] indicated that the temporal structure of the spike train conveyed a large fraction of the total information transmitted. In the simulation of Fig. 4, the MI measured was only 0.44 bits when the neuron response was represented only by its spike count (total number of spikes), whereas it was 0.73 bits when the full binary string representation was used (Fig. 3). In other words, considering the spike train temporal structure resulted in a 75% increase of information transmitted. Fig. 4A shows the value of the *surprise* calculated using spike counts (thin line) and binary strings (thick line). Stimulus surprise analysis determined that the most informative stimuli were usually characterized by high

correlation between the MFs (not shown), suggesting a role for GCs as correlation detectors across their different afferents. In addition, the results suggest that, when LTP occurs at the MF-GC synapse, the overall information transfer is enhanced in the system (Fig 4B), whereas stimulus specific information for the most informative stimuli reaches a maximum at an intermediate value of release probability (not shown).

## 5 Conclusions

The development of detailed spiking models was another step towards understanding the information transfer and the coding in the cerebellum granular layer [30]. A simplification of the model currently underway will allow us to construct large and realistic networks. Their implementation into hybrid software-FPGA circuits and efficient software [31] will eventually provide the basis for robotic and industrial implementations.

## References

1. Rieke, F., Warland, D., Steveninck, R.R., Bialek, W.: Spikes - Exploring the neural code. The MIT Press (1997)
2. Gerstner, W., Kistler, W.M.: Spiking Neuron Models. Cambridge University Press (2002)
3. O'Donovan, M.J., Rinzel, J.: Synaptic depression: a dynamic regulator of synaptic communication with varied functional roles. *Trends Neurosci* **20(10)** (1997) 431–433
4. Buonomano, D.V.: Decoding temporal information: A model based on short-term synaptic plasticity. *J Neurosci* **20(23)** (2000) 1129–1141
5. Shannon, C.E.: A mathematical theory of communication. *Bell System Technical J* **27** (1948) 379–423
6. Panzeri, S., Schultz, S.R., Treves, A., Rolls, E.T.: Correlations and the encoding of information in the nervous system. *Proc Royal Soc London B: Biol Sci* **266** (1999) 1001–1012
7. Bezzi, M., Diamond, M., Treves, A.: Redundancy and synergy arising from correlations in large ensembles. *J Comput Neurosci* **12** (2002) 165–174
8. Marr, D.: A theory of cerebellar cortex. *J Physiol* **202(2)** (1969) 437–470
9. Albus, J.S.: A theory of cerebellar function. *Math Biosci* **10** (1971) 25–61
10. Braitenberg, V.: Is the cerebellar cortex a biological clock in the millisecond range? *Prog Brain Res* **25** (1967) 334–346
11. Medina, J.F., Garcia, K.S., Nores, W.L., Taylor, N.M., Mauk, M.D.: Timing mechanisms in the cerebellum: testing predictions of a large-scale computer simulation. *J Neurosci* **20(14)** (2000) 5516–5525
12. Mitchell, S.J., Silver, R.A.: Shunting inhibition modulates neuronal gain during synaptic excitation. *Neuron* **38(3)** (2003) 433–445
13. Sola, E., Prestori, F., Rossi, P., Taglietti, V., D'Angelo, E.: Increased neurotransmitter release during long-term potentiation at mossy fibre-granule cell synapses in rat cerebellum. *J Physiol* **557** (2004) 843–861



14. Chadderton, P., Margrie, T.W., Häusser, M.: Integration of quanta in cerebellar granule cells during sensory processing. *Nature* **428(6985)** (2004) 856–860
15. Rossi, P., Sola, E., Taglietti, V., Borchartd, T., Steigerwald, F., Utvik, J.K., Ottersen, O.P., Köhr, G., D'Angelo, E.: NMDA receptor 2 (NR2) C-terminal control of NR open probability regulates synaptic transmission and plasticity at a cerebellar synapse. *J Neurosci* **22(22)** (2002) 9687–9697
16. DiGregorio, D.A., Nusser, Z., Silver, R.A.: Spillover of glutamate onto synaptic AMPA receptors enhances fast transmission at a cerebellar synapse. *Neuron* **35(3)** (2002) 521–533
17. Cathala, L., Brickley, S., Cull-Candy, S., Farrant, M.: Maturation of EPSCs and intrinsic membrane properties enhances precision at a cerebellar synapse. *J Neurosci* **23(14)** (2003) 6074–6085
18. D'Angelo, E., De Filippi, G., Rossi, P., Taglietti, V.: Synaptic excitation of individual rat cerebellar granule cells in situ: evidence for the role of NMDA receptors. *J Physiol* **484** (1995) 397–413
19. D'Angelo, E., Rossi, P., Armano, S., Taglietti, V.: Evidence for NMDA and mGlu receptor-dependent long-term potentiation of mossy fiber-granule cell transmission in rat cerebellum. *J Neurophysiol* **81(1)** (99) 277–87
20. D'Angelo, E., Nieuwenhuis, T., Maffei, A., Armano, S., Rossi, P., Taglietti, V., Fontana, A., Naldi, G.: Theta-frequency bursting and resonance in cerebellar granule cells: experimental evidence and modeling of a slow k<sup>+</sup>-dependent mechanism. *J Neurosci* **21(3)** (2001) 759–770
21. Silver, R.A., Cull-Candy, S.G., Takahashi, T.: Non-NMDA glutamate receptor occupancy and open probability at a rat cerebellar synapse with single and multiple release sites. *J Physiol* **494** (1996) 231–250
22. Hines, M.L., Carnevale, N.T.: NEURON: a tool for neuroscientists. *Neuroscientist* **7(2)** (2001) 123–135
23. Destexhe, A., Mainen, Z.F., Sejnowski, T.J.: Synthesis of models for excitable membranes, synaptic transmission and neuromodulation using a common kinetic formalism. *J Comput Neurosci* **1(3)** (1994) 195–230
24. Barbour, B.: An evaluation of synapse independence. *J Neurosci* **21(20)** (2001) 7969–84
25. Tsodyks, M.V., Markram, H.: The neural code between neocortical pyramidal neurons depends on neurotransmitter release probability. *Proc Nat Acad Sci USA* **94(2)** (1997) 719–23
26. Neher, E., Sakaba, T.: Estimating transmitter release rates from postsynaptic current fluctuations. *J Neurosci* **21(24)** (2001) 9638–54
27. Saftenku, E.: Modeling of slow glutamate diffusion and AMPA receptor activation in the cerebellar glomerulus. *J Theor Biol* (in press) (2005)
28. Rosenmund, C., Feltz, A., Westbrook, G.L.: Synaptic NMDA receptor channels have a low open probability. *J Neurosci* **15(4)** (1995) 2788–2795
29. Bezzi, M., Nieuwenhuis, T., Arleo, A., D'Angelo, E., Coenen, O.J.M.: Information transfer at the mossy fiber-granule cell synapse of the cerebellum. *Soc Neurosci Abs* **827.5** (2004)
30. Philippon, D., Coenen, O.J.M.: Model of granular layer encoding in the cerebellum. *Neurocomputing* **58-60** (2003) 575–580
31. Boucheny, C., Carrillo, R.R., Ros, E., Coenen, O.J.M.D.: Real-time spiking neural network: an adaptive cerebellar model. In: *Proc 8th Int. Work-Conf Artif Neural Net*, Springer-Verlag LNCS (2005)

# The After-Hyperpolarization Amplitude and the Rise Time Constant of IPSC Affect the Synchronization Properties of Networks of Inhibitory Interneurons

Angelo Di Garbo, Alessandro Panarese, Michele Barbi,  
and Santi Chillemi

Istituto di Biofisica CNR, Sezione di Pisa,  
Via G. Moruzzi 1, 56124 Pisa, Italy  
{angelo.digarbo, alessandro.panarese, michele.barbi,  
santi.chillemi}@pi.ibf.cnr.it  
<http://www.pi.ibf.cnr.it>

**Abstract.** The Fast Spiking (FS) interneurons are coupled by both electrical and inhibitory synapses and experimental findings suggest that they operate as a clockwork affecting the processing of neural information. At present, it is not known which is the functional role of electrical synapses in a network of inhibitory interneurons. In our contribution, by using a single compartment biophysical model of an FS cell, we determine the parameter values leading to the emergence of synchronous regimes in a network of FS interneurons coupled by chemical and electrical synapses. We also compare our results with those recently obtained for a pair of coupled Integrate & Fire neural models [1].

## 1 Introduction

Experimental findings reveal that networks of GABAergic inhibitory interneurons contribute to the generation of  $\gamma$ -rhythms (30-80 Hz) in neocortex by synchronizing their own firing and that of the pyramidal cells receiving their inhibitory inputs [2]. Both experimental and theoretical results relate the synchronous discharge of a population of inhibitory interneurons to some features of their coupling [3-8]. Recently, it was found that Fast-Spiking (FS) neocortical inhibitory interneurons are interconnected by electrical synapses too [9-10]. At present an open problem is that to understand which is the functional role for the coexistence of these two types of coupling, and to determine how the network dynamical behavior depends on them.

This topic was recently addressed for a pair of Leaky Integrate & Fire models in [1] and the main results were the followings: 1) synchrony is promoted increasing the amplitude of the stimulation current; 2) increasing electrical coupling fosters in-phase (anti-phase) coordination in pairs connected by fast (slow) inhibitory synapses. In this paper we show that these results do not hold in general for more realistic biophysical models of FS interneurons. In particular we found that the increasing of electrical coupling also with slow synapses can foster synchrony, whereas only a growing amplitude of the stimulation

current cannot. Finally, we highlight how both the after-hyperpolarization amplitude and the rise time constant of the inhibitory postsynaptic current (IPSC) affect the synchronization properties of a network of coupled cells.

## 2 Methods

FS interneurons are not capable of generating repetitive firing of arbitrary low frequency when injected with constant currents [11, 12], thereby they have type II excitability property [13].

Recent experiments carried out on *in vitro* FS cells reveal that they have high firing rates (up to  $\sim 200$  Hz), average resting membrane potential of  $-72$  mV and input resistance  $\sim 89$  M $\Omega$ ; their action potential has a mean half-width  $\sim 0.35$  ms, average amplitude  $\sim 61$  mV and after-hyperpolarization amplitude  $\sim 25$  mV [9,10,12].

### 2.1 Model Description

Here we propose a new single compartment biophysical model of a FS interneuron, well accounting for the features above, defined as follows:

$$CdV/dt = I_E - g_{Na} m^3 h (V - V_{Na}) - g_K n (V - V_K) - g_L (V - V_L) \quad (1a)$$

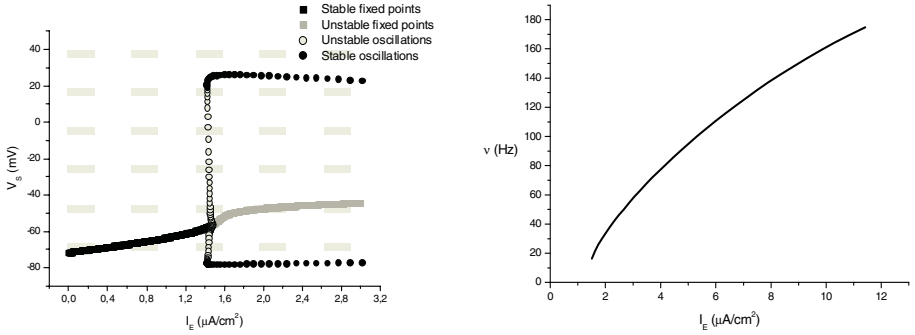
$$dx/dt = (x_\infty - x) / \tau_x, \quad x_\infty = \alpha_x / (\alpha_x + \beta_x), \quad \tau_x = 1 / (\alpha_x + \beta_x), \quad (x = m, h, n) \quad (1b)$$

where  $C = 1 \mu F/cm^2$  and  $I_E$  is the external stimulation current. The maximal specific conductances and the reversal potentials are respectively:  $g_{Na} = 85$  mS/cm $^2$ ,  $g_K = 60$  mS/cm $^2$ ,  $g_L = 0.15$  mS/cm $^2$  and  $V_{Na} = 65$  mV,  $V_K = -95$  mV,  $V_L = -72$  mV.

To model the kinetic of the Na $^+$  current we estimated theoretical curves for the voltage dependence of the activation and deactivation rate variables  $\alpha_m(V)$ ,  $\beta_m(V)$ . This was carried out by using the data obtained on hippocampal FS interneurons, concerning the steady-state activation and inactivation and the corresponding activation/deactivation time constant of the Na $^+$  current, reported in a recent paper [14]. A similar work was done for  $\alpha_h(V)$ ,  $\beta_h(V)$  and the final result is:  $\alpha_m = 3.0 \exp[(V+25.1)/20]$ ,  $\beta_m = 3.0 \exp[-(V+25.1)/27]$ ,  $\alpha_h = 0.026 \exp[-(V+58.3)/15]$ ,  $\beta_h = 0.026 \exp[(V+58.3)/12]$ .

Voltage-gated K $^+$  channels are assembled from subunits of four major subfamilies, designated as Kv1, Kv2, Kv3, Kv4 [15]. FS inhibitory interneurons express Kv3 subunits at a very high level and this has been found necessary for their phenotype [16]. A model for the gating of Kv3 channels in hippocampal FS interneurons has yet been proposed [16]. In this paper we will use this formulation with a different value for the parameter vAHP (that shapes the amplitude of the after-hyperpolarization phase of the action potential) in order that the simulated action potential has an after-hyperpolarization amplitude  $\sim 25$  mV. This leads to the rate variable expressions :  $\alpha_n = [-0.019*(V-4.2)]/\{\exp[-(V-4.2)/6.4]-1\}$ ,  $\beta_n = 0.016*\exp(-V/vAHP)$ , with vAHP = 13 mV.

In this model the onset of periodic firing occurs through a subcritical Hopf bifurcation for  $I_E \approx 1.47 \mu A/cm^2$  with a well defined frequency ( $\sim 16$  Hz), according to the type II excitability property [13] (Fig. 1).



**Fig. 1.** Characterization of the model. Left panel: bifurcation diagram for the onset of periodic firing; stable fixed points (black squares), unstable fixed points (gray squares), stable oscillations (black circles), unstable oscillations (white circles). Right panel: discharge frequency against stimulation current

## 2.2 Synaptic Coupling Modeling

The estimated mean conductance of the inhibitory cell-cell coupling for young (old) animals was  $\langle G_{\text{GABA}} \rangle = 1.7 \text{ nS}$  ( $0.8 \text{ nS}$ ) [9,12]. The IPSC was characterized by a reversal potential  $V_{\text{Rev}} = -80 \text{ mV}$  and a mean decay time constant  $\langle \tau_{\text{Decay}} \rangle = 2.6 \text{ ms}$  [12]. Similarly, for young (old) animals the mean value of the electrical conductance between a pair of coupled cells was  $\langle G_{\text{Ei}} \rangle = 0.6 \text{ nS}$  ( $0.2 \text{ nS}$ ) [9,12]. However, even for young animals, values of  $G_{\text{Ei}}$  ranging from 0.4 up to  $5.5 \text{ nS}$  ( $\langle G_{\text{Ei}} \rangle = 1.6 \text{ nS}$ ) were also found [10].

The electrical and chemical synapses are modeled as follows. The postsynaptic current for the GABAergic synapse at time  $t > t_N$  is given by  $I_{\text{Sy}}(t) = g_{\text{Sy}} s_T(t) (V_{\text{Post}}(t) - V_{\text{Rev}}) = g_{\text{Sy}} \sum_j s(t - t_j) (V_{\text{Post}}(t) - V_{\text{Rev}})$ , where  $g_{\text{Sy}}$  is the specific maximal conductance of the inhibitory synapse (in  $\text{mS}/\text{cm}^2$  unit),  $s(t) = A[\exp(-t/\tau_{\text{Decay}}) - \exp(-t/\tau_{\text{Rise}})]$ ,  $t_j$  ( $j = 1, 2, \dots, N$ ) are the times at which the presynaptic neuron generated spikes,  $\tau_{\text{Decay}}$  and  $\tau_{\text{Rise}}$  are the decay and rise time constants of the IPSC. The electrical synapse is modeled as  $I_{\text{Ei}} = g_{\text{Ei}} (V_{\text{Post}} - V_{\text{Pre}})$ , where  $g_{\text{Ei}}$  is the maximal conductance of the gap junction (in  $\text{mS}/\text{cm}^2$  unit).

## 2.3 Analysis Method

The theory of weakly coupled oscillators is used to analyze the pair of coupled FS interneuron models. The theory shows that, in the limit of weak coupling ( $\epsilon \ll 1$ ), the state of each oscillating cell is completely described by its phase. Moreover the time evolution of the phase of each oscillator can be described by a well defined equation accounting for the coupling with the other units. Thus, this approach will be used here to investigate the phase-locked states of a set of coupled oscillators and the corresponding stability [17].

Let us consider the case of two coupled oscillators and assume that for  $\epsilon = 0$  each oscillator possesses a stable limit cycle  $X_o(t)$  of period  $T$ . Then, for  $\epsilon \neq 0$  ( $\epsilon \ll 1$ ) the state of each oscillator is defined by its phase  $\theta_i$  ( $i = 1, 2$ ) and the corresponding

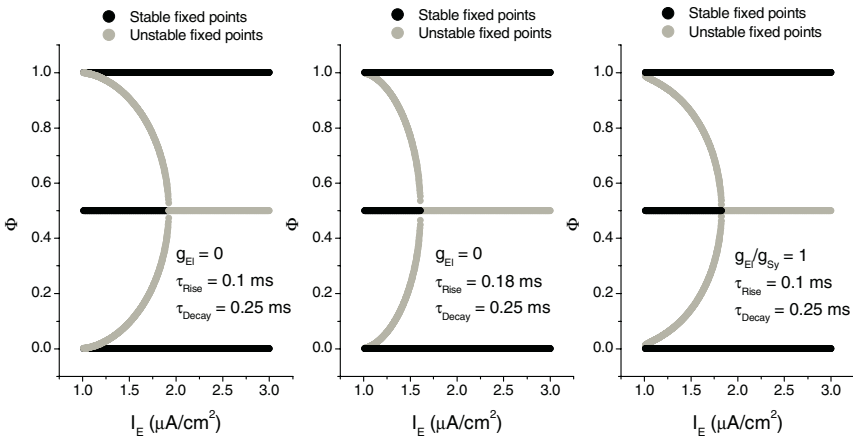
dynamical behavior is determined by the following equations:  $d\theta_1/dt = 1 + \varepsilon H_1(\phi)$  and  $d\theta_2/dt = 1 + \varepsilon H_2(-\phi)$ , where  $\varepsilon$  is the intensity of the coupling,  $\phi = \theta_2 - \theta_1$ ,  $H_1(\phi)$  and  $H_2(-\phi)$  are  $T$ -periodic functions which take into account the effect of coupling.

For a pair of identical FS cells coupled by inhibitory synapses it is  $H_{1,s}(\phi) = 1/T \int_0^T Y_1(t) s_T(t+\phi)(V_{Rev} - V_o(t)) dt$  and  $H_{2,s}(-\phi) = 1/T \int_0^T Y_1(t) s_T(t - \phi)(V_{Rev} - V_o(t)) dt$ , where  $Y_1(t)$  is the first component of the normalized  $T$ -periodic solution of the adjoint equation  $dY(t)/dt = - [J(X_o(t))]^T Y(t)$ , with  $\int_0^T Y(t) \cdot X_o(t) dt = 1$ ,  $J^T$  is the transpose of the Jacobian matrix  $J = \partial F_1(X_o)/\partial X_j$ ,  $V_o(t)$  is the membrane voltage of the unperturbed FS cell model and  $s_T(t)$  is the coupling  $T$ -periodic function when the membrane voltage of the presynaptic cell is given by  $V_o(t)$ . Similarly for electrical synapses it is  $H_{1,e}(\phi) = 1/T \int_0^T Y_1(t) (V_o(t+\phi) - V_o(t)) dt$ ,  $H_{2,e}(-\phi) = 1/T \int_0^T Y_1(t) (V_o(t - \phi) - V_o(t)) dt$  [17].

The time evolution of  $\phi$  is determined by  $d\phi/dt = \varepsilon[H_2(-\phi) - H_1(\phi)] = -\varepsilon D(\phi)$  ( $\varepsilon = g_{Sy}$  for inhibitory coupling,  $g_{El}$  for electrical coupling) and the phase locked states can be determined by searching the solutions of the equation  $D(\phi^*) = 0$ . A phase locked state is characterized by a constant phase difference  $\phi^*$  between the two oscillators and it will be stable (or unstable) according as  $dD/d\phi > 0$  ( $dD/d\phi < 0$ ). For mixed coupling  $\varepsilon D(\phi) = [g_{Sy}D_s(\phi) + g_{El}D_e(\phi)]$ .

### 3 Results

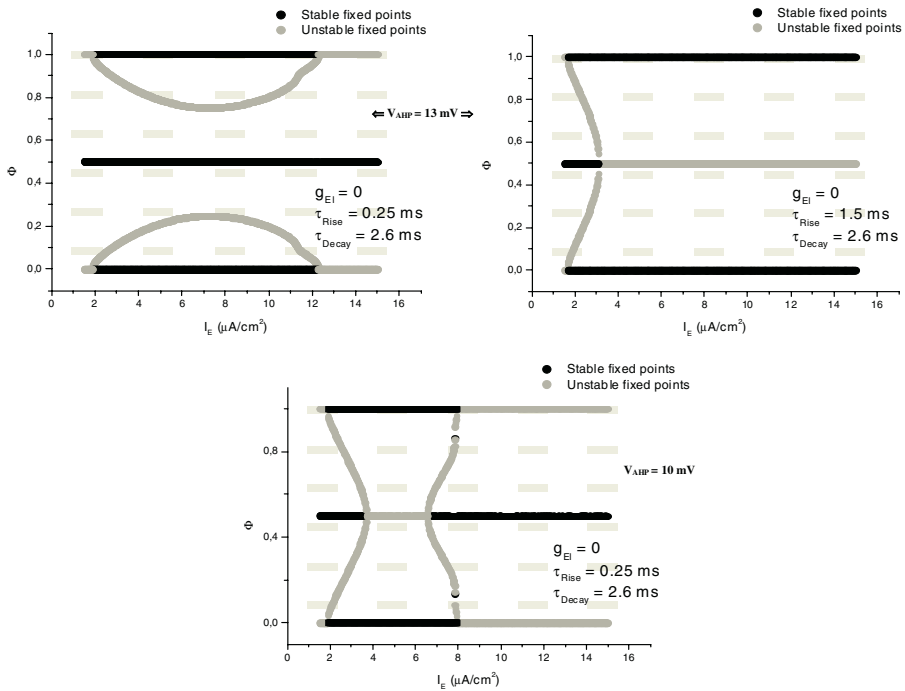
First of all we studied the phase-locking states of a pair of LIF model as done in [1], but using a more realistic time course of the IPSC which fits better the FS experimental data than the  $\alpha$ -function. We shown, according to [1] that, in absence of electrical coupling, increasing the amplitude of the stimulation current promotes in-phase firing;



**Fig. 2.** Phase-locking states for two LIF models coupled with realistic IPSC and electrical coupling. Left panel: inhibitory coupling alone, with  $\tau_{Rise} = 0.1$  ms,  $\tau_{Decay} = 0.25$  ms. Middle panel: inhibitory coupling alone with  $\tau_{Rise} = 0.18$  ms,  $\tau_{Decay} = 0.25$  ms. Right panel: coexistence of inhibitory and electrical coupling with  $\tau_{Rise} = 0.1$  ms,  $\tau_{Decay} = 0.25$  ms,  $g_{El}/g_{Sy} = 1$

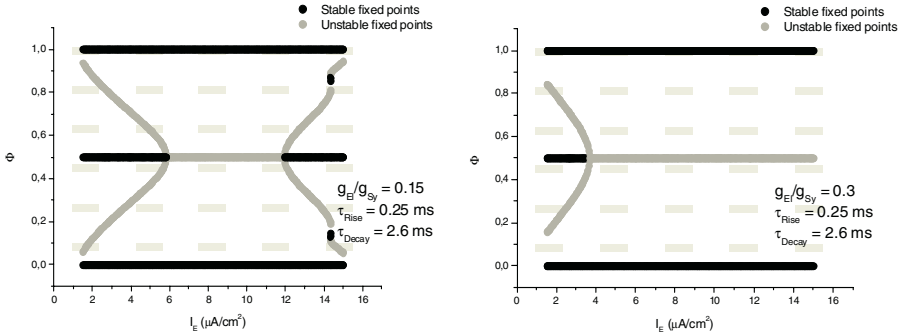
it must be highlighted that this effect is as great as the parameter  $\tau_{\text{Rise}} \rightarrow \tau_{\text{Decay}}$  (Fig. 2, left and middle panels). When the electrical coupling is on synchronous regimes are promoted too (Fig. 2, right panel). Moreover, our results indicate also that increasing the intensity of the electrical coupling promotes synchrony in despite of the speed of the inhibitory synapse (data not shown). This result is quite different from that obtained in [1] and is strictly true if we consider a realistic action potential half-width (this means  $\beta \approx 0.3$  in [1]).

Let us come to the study of the phase-locking states of a pair of biophysical models of FS neural cells. What we found indicates that the results obtained using the LIF model with IPSCs described by the  $\alpha$ -function do not hold in general for conductance-based models of FS interneurons with more realistic IPSCs. In fact we shown that for  $\tau_{\text{Rise}}$ ,  $\tau_{\text{Decay}}$  values within their physiological range, increasing  $I_E$  does not promote synchronization when  $g_{EI} = 0$ ,  $g_{Sy} \neq 0$  (Fig. 3, top left panel). Moreover, two facts must be noted: first that increasing  $\tau_{\text{Rise}}$  ( $\tau_{\text{Rise}} < \tau_{\text{Decay}}$ ) promotes in-phase firing as the stimulation current increases and second that decreasing the after-hyperpolarization amplitude of the action potential (decreasing vAHP) promotes synchronization in a well defined range of  $I_E$  values alone. In the last case for higher  $I_E$  values only anti-phase firing are stable (Fig. 3, top right panel, bottom panel). Finally,



**Fig. 3.** Phase-locking states for two biophysical models coupled by inhibitory synapses ( $g_{EI} = 0$ ). Top left panel:  $\tau_{\text{Rise}} = 0.25$  ms,  $\tau_{\text{Decay}} = 2.6$  ms, vAHP = 13 mV. Top right panel:  $\tau_{\text{Rise}} = 1.5$  ms,  $\tau_{\text{Decay}} = 2.6$  ms, vAHP = 13 mV. Bottom panel:  $\tau_{\text{Rise}} = 0.25$  ms,  $\tau_{\text{Decay}} = 2.6$  ms, vAHP = 10 mV

we show that the electrical coupling plays a crucial role in promoting synchrony. In fact in the case  $g_{EI} \neq 0$ ,  $g_{Sy} \neq 0$  the probability to get synchronous high-frequency phase-locking states increases as  $g_{EI}$  increases too (**Fig. 4**).



**Fig. 4.** Phase-locking states for two biophysical models coupled by inhibitory and electrical synapses. Left panel:  $\tau_{\text{Rise}} = 0.25$  ms,  $\tau_{\text{Decay}} = 2.6$  ms,  $g_{EI}/g_{Sy} = 0.15$ . Right panel:  $\tau_{\text{Rise}} = 0.25$  ms,  $\tau_{\text{Decay}} = 2.6$  ms,  $g_{EI}/g_{Sy} = 0.3$

## 4 Discussion

In this paper we studied the phase-locking states of a pair of identical FS interneuron models to understand how the combination of electrical and inhibitory coupling affects synchronization patterns in networks of FS spiking interneurons.

At first, we studied a LIF pair as in [1] with a more realistic time course of the IPSC ( $\tau_{\text{Rise}} \ll \tau_{\text{Decay}}$ ). In the case of inhibitory coupling alone, our results agree with those obtained in [1], but when we consider the electrical coupling the results are contrasting. In fact we found that if we consider a realistic action potential half-width ( $\beta \approx 0.3$  in [1]), in phase firing is promoted even for slow synapses.

Later we introduced a new biophysical model of a FS interneuron and we studied the phase-locking states of a pair of cells connected by reciprocal inhibitory and electrical coupling. By comparing these results with those obtained with the LIF model in [1], we found that some of them are contrasting. In particular the phase-locking patterns depend critically on some parameters of the biophysical model, as the after-hyperpolarization potential and the time constants of the IPSC time course.

Finally, we found that for our biophysical network model, the electrical coupling plays a crucial role in promoting the synchronization of the FS cells, when the relevant parameter values are in the physiological range.

## References

1. Lewis T., Rinzel J.: Dynamics of spiking neurons connected by both inhibitory and electrical coupling, *J. Comp. Neuroscience* 14 (2003) 283-309
2. Galarreta M., Hestrin S.: Electrical synapses between GABA-releasing interneurons. *Nat. Neurosci.* 2 (2001) 425 - 433

3. Di Garbo A., Barbi M., Chillemi S.: Synchronization in a network of fast-spiking interneurons, *BioSystems* 67 (2002) 45 - 53
4. Van Vreeswijk C. A., Abbott L. F., Ermentrout G. B.: Inhibition, not excitation, synchronizes coupled neurons. *J. Comp. Neurosci.*, 1 (1995) 303 - 313
5. Wang X. J., Rinzel J.: Alternating and synchronous rhythms in reciprocally inhibitory model neurons. *Neural Comput.*, 4 (1992) 84 - 97
6. Wang X. J., Buzsaki G.: Gamma oscillations by synaptic inhibition in an interneuronal network model, *J. Neurosci.* 16 (1996) 6402 - 6413
7. Whittington M. A., Traub R. D., Jefferys J. G. R.: Synchronized oscillations in interneuron networks driven by metabotropic glutamate receptor activation. *Nature*, 373 (1995) 612-615.
8. Whittington M. A., Traub R. D., Kopell N., Ermentrout B., Buhl E. H.: Inhibition-based rhythms: experimental and mathematical observations on network dynamics, *Int. J. Psychophysio.* 38 (2001) 315 - 336
9. Galarreta M., Hestrin S.: A network of fast-spiking cells in the cortex connected by electrical synapses. *Nature*, 402 (1999) 72-75
10. Gibson J.R., Beierlein M., Connors B.W.: Two networks of electrically coupled inhibitory neurons in neocortex. *Nature*, 402 (1999) 75-79
11. Erisir A., Lau D., Rudy B., Leonard C. S.: Function of specific  $K^+$  channels in sustained high-frequency firing of fast-spiking neocortical interneurons, *J. Neurophysiology* 82 (1999) 2476-2489
12. Galarreta M., Hestrin S.: Electrical and chemical Synapses among parvalbumin fast-spiking GABAergic interneurons in adult mouse neocortex, *PNAS USA* 99 (2002) 12438-12443
13. Rinzel J., Ermentrout B.: Analysis of neural excitability and oscillations, Eds. Koch and Segev, *Methods in neural modelling* (1989), The MIT Press, Cambridge
14. Martina M., Jonas P.: Functional differences in  $Na^+$  channel gating between fast spiking interneurons and principal neurons of rat hippocampus, *J. of Physiol.* 505.3 (1997) 593-603
15. Coetzee W. A. et al.: Molecular diversity of  $K^+$  channels, *Annals of the New York Academy of Sciences* 868 (1999), 233 - 285
16. Lien C. C., Jonas P.:  $Kv3$  potassium conductance is necessary and kinetically optimized for high-frequency action potential generation in hippocampal interneurons, *J. of Neurosci.* 23 (2003) 2058-2068
17. Ermentrout B.: Neural networks as spatio-temporal pattern-forming systems. *Rep. Prog. Phys.* 61 (1998) 353-430.



# TiViPE Simulation of a Cortical Crossing Cell Model

Tino Lourens<sup>1</sup> and Emilia Barakova<sup>2</sup>

<sup>1</sup> Honda Research Institute Japan Co., Ltd.,  
8-1 Honcho, Wako-shi, Saitama, 351-0114, Japan  
`tino@jp.honda-ri.com`

<sup>2</sup> Brain Science Institute, Riken,  
2-1 Hirosawa, Wako-shi, Saitama, 351-0198, Japan  
`emilia@brain.riken.jp`

**Abstract.** Many cells in cat and monkey visual cortex (area V1 and area 17) respond to gratings and bar patterns of different orientation between center and surround [18]. It has been shown that these cells respond on average 3.3 times stronger to a crossing pattern than to a single bar [16]. In this paper a computational model for a group of neurons that respond solely to crossing patterns is proposed, and has been implemented in visual programming environment TiViPE [10]. Simulations show that the operator responds very accurately to crossing patterns that have an angular difference between 2 bars of 40 degrees or more, the operator responds appropriately to bar widths that are bound by 50 to 200 percent of the preferred bar width and is insensitive to non-uniform illumination conditions, which appear to be consistent with the experimental results.

## 1 Introduction

Neurons in the primary visual cortex (V1) respond in well defined ways to stimuli within their classical receptive field (CRF), but these responses can be modified by stimuli overlying the surrounding area. This non classical surround provides input from a larger portion of the visual scene than originally thought, permitting integration of information at early levels in the visual processing stream. Recent works indicate that neuronal surround modulation at cross-orientation, an orientation orthogonal to the preferred orientation of the CRF, play a key role in intermediate-level visual tasks, such as perceptual pop-out [14], contrast facilitation [3, 20], and contextual modulation [8, 4, 5], and could endow neurons with a graded specialization for processing angular visual features such as corners and junctions [18, 4].

Neuronal output activity was enhanced in both cat and macaque primary visual cortex (V1) when a surrounding field at a significantly different orientation (30 degrees or more) was added to the preferred orientation of the CRF [18]. The response of these neurons to line crossings at different angles have been described more extensively by [15, 17, 16]. They reported that more than 30 percent of the

neurons studied in the cat striate cortex (area 17) had a significant increase in response (by 3.3 times on average) under stimulation of two crossing bars of distinctive preferred orientation as compared to a single bar.

Many of the cells found so far are cross-orientation selective, and respond vigorously to patterns that contain a different orientation between center (CRF) and non-classical surround. However, most of these cells also respond to a single bar or grating pattern of one orientation. Neuronal cells with a graded specialization for crossing type of junctions, will be modeled in this paper. The constructed operator will be termed *crossing cell operator*.

The paper is organized as follows: In Section 2 the crossing cell operator will be constructed and its properties evaluated. Section 3 the experimental setup will be presented and results of the operator in a real world environment under different non-uniform illumination conditions will be demonstrated. The paper finishes with a discussion.

## 2 Crossing Cells

Recent reports [18, 16, 3, 7] have demonstrated that the responsiveness of neurons in V1 is modulated by stimuli placed outside their CRFs. Cells with suppressive surround influence corresponding to surround orientation of a grating or bar pattern at the neuron's preferred orientation were found. The responses of these cells were roughly sinusoidal to sigmoidal increasing when the surround orientation was modified, reaching the strongest response when it was perpendicular to the preferred orientation of the CRF. We believe that crossing cells, which have a graded specialization for crossing type of junction of neurons, belong to this group of neurons.

### 2.1 Crossing Cell Operator

The complex cell operator responds strongly to a bar or an edge of a preferred frequency, but its response decreases at line ends, junctions and crossings. Since its response is weak at crossings, a subunit is modeled with a center-off response and a surround-on response, as suggested by [1, 18], for a single preferred orientation  $\theta$ :

$$\mathbf{O}_{\sigma,\theta}(x, y) = w_o (\mathcal{C}_{\sigma,\theta}(x_2, y_2) + \mathcal{C}_{\sigma,\theta}(x_3, y_3) - (\mathcal{C}_{\sigma,\theta}(x, y))) \quad , \quad (1)$$

where  $x_2 = x + d\sigma \sin \theta$ ,  $y_2 = y - d\sigma \cos \theta$ ,  $x_3 = x - d\sigma \sin \theta$ ,  $y_3 = y + d\sigma \cos \theta$ ,  $d = 8/\sqrt{5}$ ,  $w_o = 2$ ,  $\sigma$  represents the scale, and  $\theta$  the preferred orientation. Details of complex cell operator  $\mathcal{C}_{\sigma,\theta}$  can be found in [9, 11].

A crossing with preferred orientations  $\theta$  and  $\theta + \varphi$  is modeled to a computational subunit by taking the minimum of the two units and the amplified complex cell responses:

$$\widehat{\mathbf{X}}_{\sigma,\theta,\varphi} = \min (\mathbf{O}_{\sigma,\theta}, \mathbf{O}_{\sigma,\theta+\varphi}, 2w_o\mathcal{C}_{\sigma,\theta}, 2w_o\mathcal{C}_{\sigma,\theta+\varphi}) \quad . \quad (2)$$

It denotes that such a subunit responds only when there are strong flanked responses ( $\mathbf{O}$ ) and strong center responses ( $\mathcal{C}$ ). Since the strongest response was

found at an orientation that is perpendicular to the preferred orientation,  $\varphi$  is set to 90 degrees. This is in analogy with the orientation difference between center and non-classical surround [7].

The crossing cell model, should respond solely to crossing patterns, hence every response other than at crossing position  $(x, y)$  is a false response and must be eliminated. In analogy with the model for endstopped cells [19] this is done by a tangential and a radial inhibition operator, correspondingly:

$$\mathcal{I}_\sigma^t(x, y) = \sum_{i=0}^{2N-1} [\mathcal{C}_{\sigma, \theta_i}(x_4, y_4) - \mathcal{C}_{\sigma, \theta_i}(x, y)]^{\geq 0} \quad (3)$$

and

$$\mathcal{I}_\sigma^r(x, y) = \sum_{i=0}^{2N-1} [\mathcal{C}_{\sigma, \theta_i}(x, y) - w_r \mathcal{C}_{\sigma, \theta_{i+N/2}}(x, y)]^{\geq 0} \quad , \quad (4)$$

where  $x_4 = x + d\sigma \cos \theta_i$ ,  $y_4 = y + d\sigma \sin \theta_i$ ,  $[z]^{\geq 0}$  is equal to 0 for negative  $z$  and equal to  $z$  elsewhere (half-wave rectification), and constant  $w_r = 4$ .

The crossing operator at a single scale and a single orientation is:

$$\mathbf{X}_{\sigma, \theta_i} = \left[ \left[ \hat{\mathbf{X}}_{\sigma, \theta_i} \right]^{\geq 0} - \frac{g}{N} (\mathcal{I}_\sigma^t + \mathcal{I}_\sigma^r) \right]^{\geq 0} \quad , \quad (5)$$

where  $g$  is a gain factor. An appropriate value is  $g = 1$ , which is used in all experiments.

Finally, a weighted summation is made to obtain appropriate spatial properties:

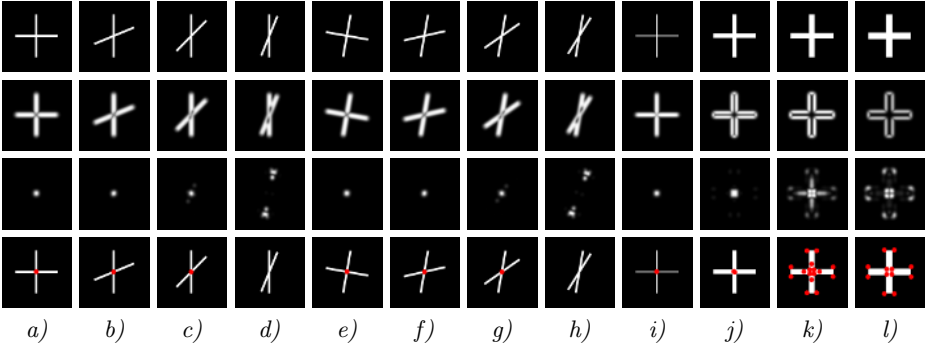
$$\mathcal{X}_{\sigma, \theta, \varphi} = \mathbf{X}_{\sigma, \theta, \varphi} * G_\sigma \quad , \quad (6)$$

where  $G_\sigma(x, y) = 1/(2\pi\sigma^2) \exp(-(x^2 + y^2)/(2\sigma^2))$  is a two-dimensional Gaussian function, to yield the crossing cell operator.

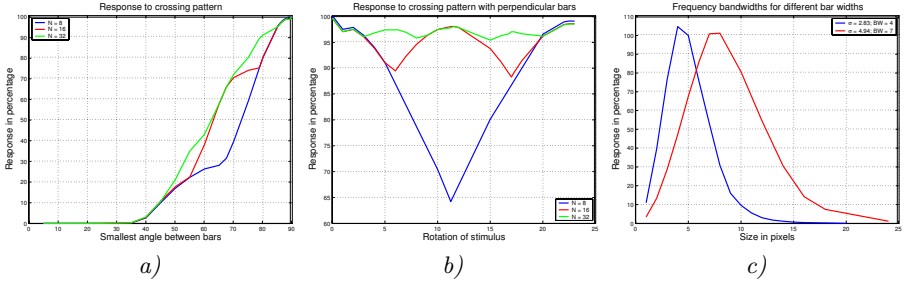
## 2.2 Properties of Crossing Cells

Figure 1 demonstrates desired responses to crossings that have a minimum angle between the two bars of 45 degrees or larger. Due to normalization there appear to be false responses at Figure 1d and f, but these responses are very small, and can therefore be neglected. The response strength to different angles is illustrated in Figure 2a. The response curve shows a relatively strong decrease in when decreasing the angle between the two bars. A half maximum response is found at an angle of around 62 degrees. Responses vanishes when the angle is smaller than 40 degrees. Figure 2a show that at least 32 orientations are needed to obtain a smooth sigmoidal response curve when changing the angle between the two bars, while Figure 2b illustrates that this number of orientations yields little fluctuation in response when rotating the input stimulus.

Figure 2c illustrates that a half-maximum response is at 55 and 180% from the preferred bar width. The crossing operator responds accurately to crossings at 50 and 200 percent of the preferred bar width, as illustrated in Figure 1i and j, but shows artefacts to bars that have a width that is more than twice the preferred bar width (Figure 1k and l).



**Fig. 1.** Responses of the crossing cell operator (6) to crossing patterns of different orientation, and bar width. First to fourth row represent input images, complex cell responses, crossing cell responses, and marked crossings, respectively. For better visualization the results of both complex and crossing cell responses have been normalized. (a-d) are the responses to a crossings with a minimum angle between the two bars of 90, 67.5, 45, and 22.5 degrees. (e-h) are like (a-d), but the overall pattern has been rotated by 10 degrees. (i-l) are like (a), but the bar widths have been set to 4, 16, 20, and 24 instead of 8 pixels. In the simulation the following parameters settings have been used:  $N = 32$  orientations,  $\sigma = 5.65$  giving the strongest stimulus to a bar width of 8 pixels, and  $\lambda = \gamma = 1$



**Fig. 2.** Responses to crossing pattern. (a) Response to different angles (measured in degrees) between two bars, where one preferred orientation is a vertical bar. (b) Response to rotation of the whole pattern with perpendicular bars. (c) Spatial frequency profiles for different preferred bar widths (BW = 4 and 7 pixels)

### 3 Crossing Cell Simulation

The crossing cell operator is used in a simulation environment called TiViPE [10], which is explained in more detail below. In the simulation environment, the crossing cell operator has been applied to natural images with different illumination conditions, where an object containing a sharp (#) mark is placed at different distances under different angles. The aim of this simulation is to confirm the results found for the test stimuli. In addition the robustness against irregular

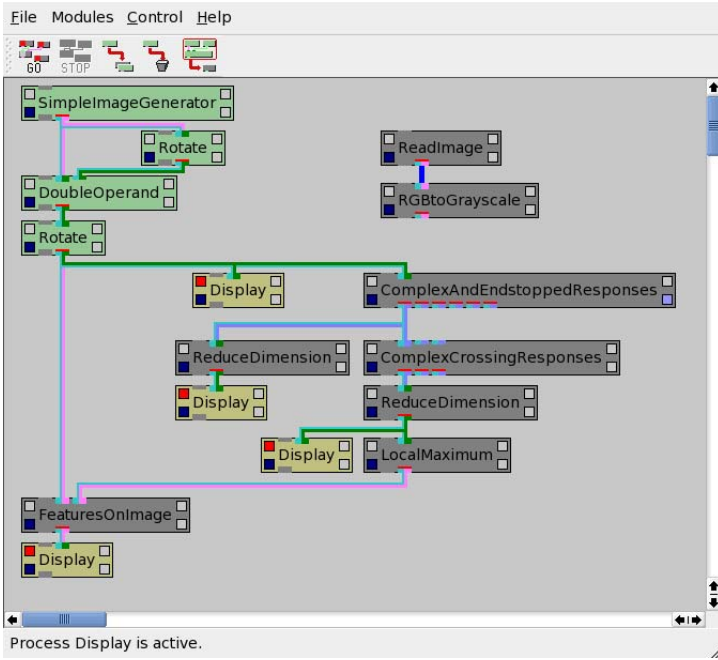


Fig. 3. Experimental setup of crossing cells

illumination conditions is evaluated. The latter is essential in a humanoid robot that operates in a real world environment.

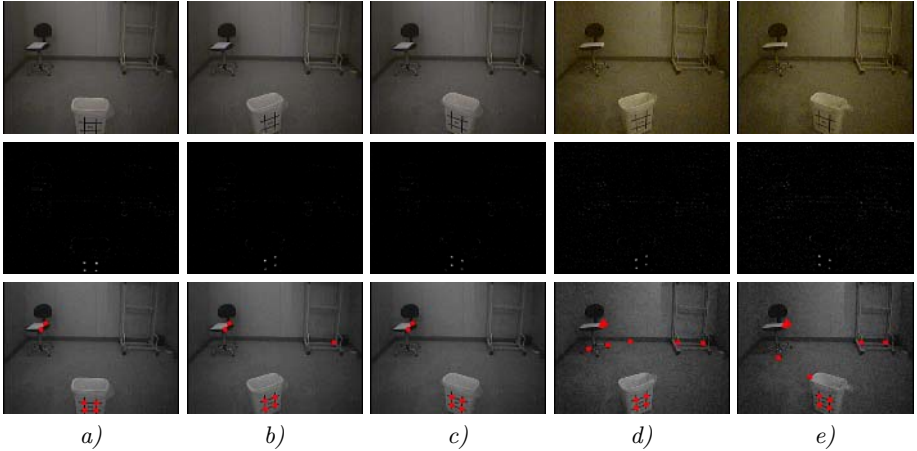
### 3.1 Experimental Setup

Figure 3 illustrates the experimental setup of the crossing cell operator in TiViPE. The four top-left icons (green) generate a synthetic image, as illustrated in the first row of Figure 1. For natural images the “ReadImage” icon followed by the “RGBtoGrayscale” icon is used instead. The four “Display” icons (yellow) from top to bottom result in input image (stimulus), complex response, crossing response, and marked crossing, respectively, and has been used to generate the results of Figure 1 and 4.

### 3.2 Crossing Cell Responses to Natural Images

In the simulation color images in VGA format (640x480 pixels) were used as input stimuli. A bucket was placed at distances of 1, 1.5, 2, and 2.5 meters at angles of -60, -30, 0, 30, and 60 degrees, under 3 different (normal, left side dimmed, right side dimmed) illumination conditions yielding a total of 60 different images, which were applied to the crossing operator (6).

All four crossing points of the sharp (#) on the bucket were determined with high accuracy for all distances and orientations of 0 and  $\pm 30$  degrees, under all



**Fig. 4.** Responses of the crossing cell operator (6) to real world images with different illumination conditions. **First row** Input stimuli. (a-c) Normal light conditions, (d-e) Use of light bulbs where lights have been dimmed on the left side, resulting in a shadow on the left side of the bucket. **Second row** Normalized crossing responses applied obtained from the respective input stimuli. **Third row** Marked crossings overlaid on the monochrome input image. The threshold to mark local maxima at crossings was set to 7, where the response of the operator is between 0 and 255. In the simulation the following parameters settings have been used:  $N = 8$  orientations,  $\sigma = 1.77$  giving the strongest stimulus to a bar width of 2.5 pixels, and  $\lambda = \gamma = 1$

3 illumination conditions, using a single scale only ( $\sigma = 1.77$ ). The operator is not responding when the bucket is placed under an angle of  $\pm 60$  degrees. This result is expected, since the operator failed to respond to the artificial stimuli with similar angle, see Figure 1d and f.

## 4 Discussion

Many neurons in primary visual cortex (V1) respond differently to a simple visual element present in isolation, compared to when it is embedded in a more complex stimulus. The difference is in the modulation by surrounding elements, hence it could tune neurons to a graded specialization for processing junctions [18, 4].

Typically the surround influence was suppressive when the surround grating was at the neuron's preferred orientation [3], but when the orientation in the surround was perpendicular to the preferred orientation facilitation became evident. Neural responses to crossings were on average more than 3 times stronger [16], but they also respond to a grating or a single bar of a preferred orientation [18, 16, 3, 7]. These neurons are therefore not specialized to purely respond to junctions. In the monkey, the majority of cells showed response suppression

with increasing grating patch diameter [2, 18], hence it is likely that a group of these neurons respond solely to junctions, crossings in particular. Another possibility is that neurons in adjacent areas use these responses for further processing that accounts for responses to crossings only. Tangential and a radial inhibition operators were used in our model, yielding the proposed “cortical crossing cell” operator.

The operator is robust to different illumination conditions, and responded accurately to crossing patterns at a wide ( $\pm 40$  degree) angle and at a wide range of scales (50 to 200 percent of the preferred bar width). However, the operator failed at crossings where the smallest angle between a pair of bars was less than four degrees and showed false responses to crossing patterns where the bar width is larger than twice the preferred bar width.

Crossing cells are expected to play a complementary role to endstopped cells which respond to line ends, corners, and junctions, but not to crossings [6, 9]. However, from functional brain modeling perspective, it is desirable to model all junction types [12]. Junctions strongly reduce the amount of visual information and likely play an important role in object recognition. These junctions can be represented as vertices in graphs and be used for symbolic reasoning [13].

The proposed model is developed as part of a larger parallel early vision system, that includes several early vision operators, grouping, attention, and learning mechanisms. The TiViPE environment serves as an integration tool for these operators and mechanisms.

## References

1. C. Blakemore and E. A. Tobin. Lateral inhibition between orientation detectors in the cat’s visual cortex. *Experimental Brain Research*, 15:439–440, 1972.
2. R. T. Born and R. B. H. Tootell. Single unit and 2-deoxyglucose studies of side inhibition in macaque striate cortex. *Proc. Natl. Acad. Sci. USA*, 88:7071–7075, 1991.
3. J. R. Cavanaugh, W. Bair, and J. A. Movshon. Selectivity and spatial distribution of signals from the receptive field surround in macaque v1 neurons. *Journal of Neurophysiology*, 88:2547–2556, 2002.
4. A. Das and C. D. Gilbert. Topography of contextual modulations mediated by short-range interactions in primary visual cortex. *Nature*, 399:655–661, June 1999.
5. V. Dragoi and M. Sur. Dynamic properties of recurrent inhibition in primary visual cortex: Contrast and orientation dependence of contextual effects. *Journal of Neurophysiology*, 83:1019–1030, 2000.
6. F. Heitger, L. Rosenthaler, R. von der Heydt, E. Peterhans, and O. Kübler. Simulation of neural contour mechanisms: from simple to end-stopped cells. *Vision Research*, 32(5):963–981, 1992.
7. H. E. Jones, W. Wang, and A. M. Sillito. Spatial organization and magnitude of orientation contrast interactions in primate v1. *Journal of Neurophysiology*, 88:2797–2808, 2002.
8. J. B. Levitt and J. S. Lund. Contrast dependence of contextual effects in primate visual cortex. *Nature*, 1997.

9. T. Lourens. *A Biologically Plausible Model for Corner-based Object Recognition from Color Images*. Shaker Publishing B.V., Maastricht, The Netherlands, March 1998.
10. T. Lourens. Tivipe –tino’s visual programming environment. In *The 28<sup>th</sup> Annual International Computer Software & Applications Conference, IEEE COMPSAC 2004*, pages 10–15, 2004.
11. T. Lourens, E. I. Barakova, H. G. Okuno, and H. Tsujino. A computational model of monkey cortical grating cells. *Biological Cybernetics*, 92(1):61–70, January 2005. DOI: 10.1007/s00422-004-0522-2.
12. T. Lourens, E. I. Barakova, and H. Tsujino. Interacting modalities through functional brain modeling. In J. Mira and J. R. Álvarez, editors, *Proceedings of the International Work-Conference on Artificial and Natural Neural Networks, IWANN 2003*, volume 2686 of *Lecture Notes in Computer Science*, pages 102–109, Menorca, Spain, June 2003. Springer-Verlag.
13. T. Lourens and R. P. Würtz. Extraction and matching of symbolic contour graphs. *International Journal of Pattern Recognition and Artificial Intelligence (IJPRAI)*, 17(7):1279–1302, November 2003.
14. H. C. Nothdurft, J. L. Gallant, and D. C. Van Essen. Response modulation by texture surround in primate area v1: Correlates of ”popout” under anesthesia. *Visual Neuroscience*, 16:15–34, 1999.
15. I. A. Shevelev, N. A. Lazareva, B. V. Novikova, A. S. Tikhomirov, and G. A. Sharaev. Double orientation tuning in the cat visual cortex units. *Neuroscience*, 61(4):965–973, 1994.
16. I. A. Shevelev, N. A. Lazareva, G. A. Sharaev, R. V. Novikova, and A. S. Tikhomirov. Selective and invariant sensitivity to crosses and corners in cat striate neurons. *Neuroscience*, 84(3):713–721, 1998.
17. I. A. Shevelev, R. V. Novikova, N. A. Lazareva, A. S. Tikhomirov, and G. A. Sharaev. Sensitivity to cross-like figures in the cat striate neurons. *Neuroscience*, 69(1):51–57, 1995.
18. A. M. Sillito, K. L. Grieve, H. E. Jones, J. Cudiero, and J. Davis. Visual cortical mechanisms detecting focal orientation discontinuities. *Nature*, 378:492–496, November 1995.
19. R. P. Würtz and T. Lourens. Corner detection in color images through a multiscale combination of end-stopped cortical cells. *Image and Vision Computing*, 18(6-7):531–541, April 2000.
20. C. Yu, S. A. Klein, and D. M. Levi. Facilitation of contrast detection by cross-oriented surround stimuli and its psychophysical mechanisms. *Journal of Vision*, 2:243–255, 2002.



# A Model of Spiking-Bursting Neuronal Behavior Using a Piecewise Linear Two-Dimensional Map

Carlos Aguirre, Doris Campos, Pedro Pascual, and Eduardo Serrano

GNB, Escuela Politécnica Superior, Universidad Autonoma de Madrid,  
28049 Madrid, Spain

{Carlos.Aguirre, Doris.Campos, Pedro.Pascual, Eduardo.Serrano}@ii.uam.es

**Abstract.** Models of neurons based on iterative maps allows the simulation of big networks of coupled neurons without loss of biophysical properties such as spiking, bursting or tonic bursting and with an affordable computational effort. A piecewise linear two dimensional map with one fast and one slow variable is used to model spiking-bursting neural behavior. This map shows oscillations similar to other phenomenological models based on maps that require a much higher computational effort. The dynamics of coupled neurons is studied for different coupling strengths.

## 1 Introduction

Numerical studies of the collective behavior of ensembles of neurons rely on models of neurons that describe the neuron behavior on the basis of differential equations. These models require a high computational effort to reproduce neuronal behavior such as spiking or bursting. Well known differential models such as the Hodgkin-Huxley (HH) model [2], or the the Hindmarsh-Rose (HR) model [3] require a number of floating point operations that range from 1200 operations in the case of the HH model to 70 operations in the HR model to simulate a single neuron in the network for 1 ms. This means that the simulation of the behavior of a neural network composed for thousand of neurons for even not very long periods of time is computationally inviable.

Recently, some models have solved this drawback of the differential models [4][5]. These new models are built over a phenomenological basis and are implemented by the use of iterative two-dimensional maps that can present similar neuro-computational properties that the differential models. These models present a low computational effort that makes possible the simulation of big ensembles of coupled neurons during relatively long periods of time. Besides, some of these models like [4] are biophysically meaningful in such a way that the parameters of the model can be selected in order to obtain a characteristic neuron behavior, such as, spiking, tonic spiking, bursting, etc.

In this work a two-dimensional continuous piecewise linear map that models spiking-bursting neural behavior is presented. The map is constructed under phenomenological assumptions and mimics the dynamics of oscillations observed

in real neurons. The model has a computational cost as low as two floating point operations at worst for each iteration. This computational effort is lower than the models presented in [4] and [5], that require 8 and 12 floating point operations respectively for each iteration. The behavior of coupled neurons is also investigated, in particular the degree of synchronization of a pair of coupled maps is studied for different coupling strengths.

## 2 The Neuron Model

The modelling of neuronal behavior by means of two-dimensional maps has been object of great interest in the last few years [4][1]. These models consider a two-dimensional system that has both a slow and a fast dynamics. In these models, fast series of spikes burst over oscillations of the slow variable. However, the behavior of fast and slow dynamics are clearly correlated and it can be observed that high values of the slow variable correspond with spikes of the fast variable and low values of the slow variable correspond with resting periods of the fast variable.

The previous observation yields us to consider a simplified model implemented as a coupled two-dimensional map  $(y, s)$  where the value of function  $y$  represents the membrane potential and is implemented as a continuous piecewise linear function where each segment of the function  $y$  represent a different state of the membrane potential (resting or spiking/bursting). The function  $s$  is a bi-valued function taking the discrete values  $(0, 1)$  and indicates if the value of the membrane potential  $y$  is de-polarizing ( $s = 1$ ) or re-polarizing ( $s = 0$ ) by changing the shape of the function  $y$ . The value of  $s$  is modified as the value of  $y$  reaches some predetermined thresholds when spiking, resting or returning from spiking.

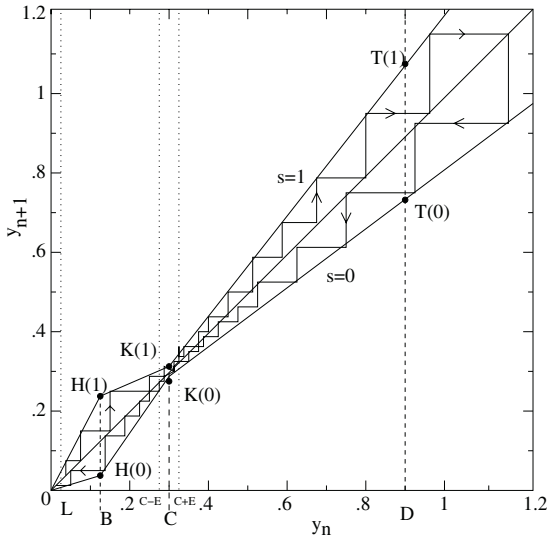
The map  $(y, s)$  is implemented in the following way:

$$y_{n+1} = \begin{cases} \frac{H(s_n)}{B} * y_n & \text{if } 0 \leq y_n < B \\ (y_n - B) * \frac{K(s_n) - H(s_n)}{C - B} + H(s_n) & \text{if } B \leq y_n < C \\ (y_n - C) * \frac{T(s_n) - K(s_n)}{D - C} + K(s_n) & \text{Otherwise} \end{cases} \quad (1)$$

$$s_{n+1} = \begin{cases} 0 & \text{if } s_n = 1 \text{ and } y_n > D \\ 1 & \text{if } s_n = 0 \text{ and } y_n < L \\ 1 & \text{if } s_n = 0 \text{ and } y_n > C - E \text{ and } y_n < C + E \\ s_n & \text{Otherwise} \end{cases} \quad (2)$$

where  $H(s) = H_0 + s*(H_1 + \sigma)$ ,  $K(s) = K_0 + s*(K_1 + \sigma)$  and  $T(s) = T_0 + s*(T_1 + \sigma)$  and  $B, C, D, E, L, H_1, H_0, K_1, K_0, T_1, T_0$  are non-negative parameters verifying the following conditions:  $L < B < C < D$ ,  $H_0 \leq B$ ,  $H_1 + H_0 \geq B$ ,  $K_0 \leq C$ ,  $K_1 + K_0 \geq C$ ,  $T_0 \leq D$  and  $T_1 + T_0 \geq D$ ,  $\sigma$  represents an external total input to the neuron. The parameter  $C$  is the spike threshold. The parameters  $L, E$  and  $D$  are thresholds that force the change of value of the slow variable  $s$  as the fast variable  $y$  crosses one of these thresholds, where  $L$  is the hyper-polarization

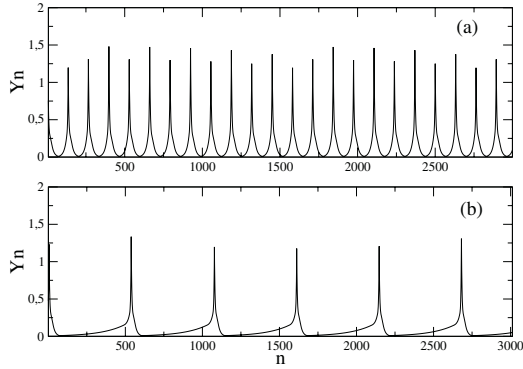
value and  $D$  is the depolarization value. The other parameters determine the length and the step of the segments that compound the function  $y$ . A plot of the map is depicted in figure 1. Note that the values of  $H(s)$ ,  $K(s)$  and  $T(s)$  can be computed in advance for both values of the variable  $s$ , therefore they need not to be computed for each step. This means that for a constant value of the external input  $\sigma$  each step requires at most two floating point operations plus four comparisons. Note that the descending part of the map ( $y, s$ ) ( $s = 0$ ) does not depend on the external input  $\sigma$  presenting so refractory properties of the neuron after spiking.



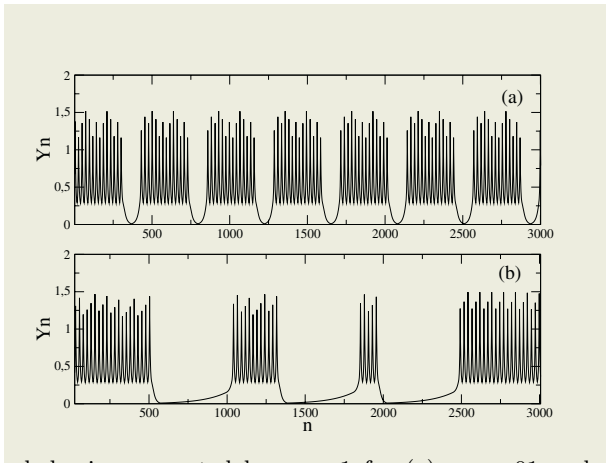
**Fig. 1.** Graph of the piecewise linear function  $y$

The temporal behavior of the map 1 for different values of the external input  $\sigma$  is presented in figures 2 and 3. The map is capable of generating both spiking or bursting series behavior. The parameters for both figures are:  $L = .01$ ,  $B = .15$ ,  $C = .3$ ,  $D = .9$ ,  $H_0 = .14$ ,  $K_0 = .29$ ,  $T_0 = .75$ ,  $H_1 = .01$ ,  $K_1 = .02$ ,  $T_1 = .4$ . In order to obtain bursting behavior we establish  $E = 0.0055$ , the wave of the bursting behavior can be observed in figure 3. Spiking behavior is obtained by simply establishing  $E = 0$ . The wave for spiking behavior can be observed in figure 2. Note that faster spiking/bursting is obtained with higher values of the external input. This behavior is consistent with the results obtained with other phenomenological or differential models (e.g. integrators or class 1 models).

In figure 4 the behavior of a spiking wave under a non-constant injection of external input is depicted. Observe that in the regime of no external input, the model remains in a quiescent state, with no bursting or spiking activity. Once the external input is reestablished, the model continues its spiking activity.



**Fig. 2.** Spiking behavior generated by map 1 for (a)  $\sigma = .01$  and (b)  $\sigma = .001$ . Parameters are described in the text



**Fig. 3.** Bursting behavior generated by map 1 for (a)  $\sigma = .01$  and (b)  $\sigma = .001$ . Parameters are described in the text

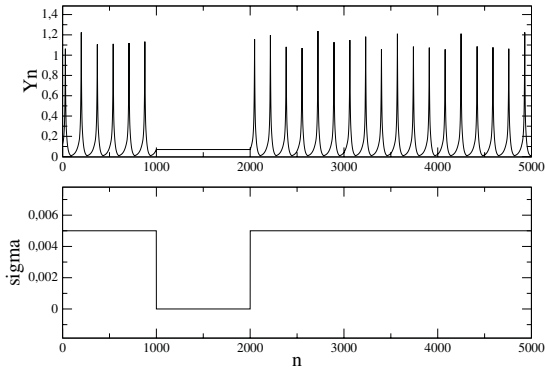
### 3 Synchronization in Two Coupled Maps

The previous map can be generalized in order to receive inputs from other neurons in the network. In the generalized model we substitute the input  $\sigma$  in neuron  $i$  by the following total input:

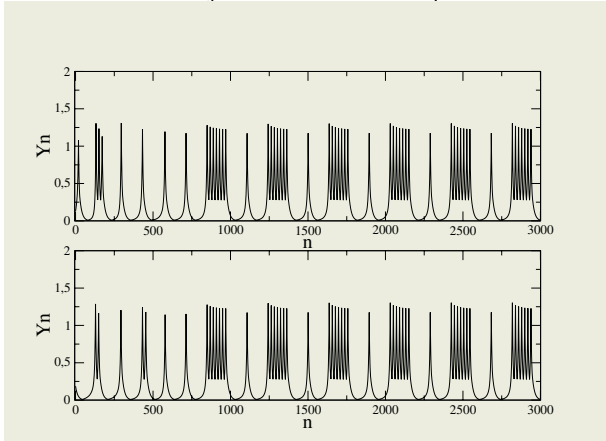
$$\sigma_i = \sigma_i^e + \frac{1}{\Gamma_i} \sum_{j=1}^N g_{ij} y_j \tag{3}$$

where  $\sigma_i^e$  is the external input to neuron  $i$ ,  $\Gamma_i$  is the number of neighbors of neuron  $i$ ,  $N$  is the number of neurons in the network and  $g_{ij}$  is a coupling coefficient between neuron  $i$  and neuron  $j$ .

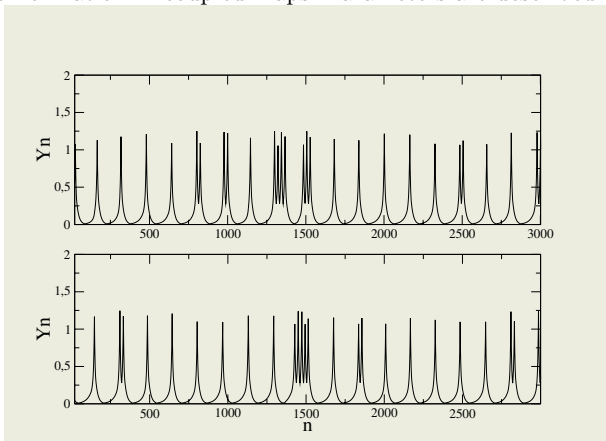
Real neurobiological experiments [6] show that ensembles of coupled neurons can present different regimes of synchronization. These regimes are reproduced



**Fig. 4.** Behavior under different injection of external input. Parameters are described in the text



**Fig. 5.** Synchronization in coupled maps. Parameters are described in the text



**Fig. 6.** Synchronization in coupled maps. Parameters are described in the text

both by differential or iterative models [1]. As a general result, systems with a higher coupling values between neurons present a greater degree of synchronization. When the coupling between neurons is small, synchronization is much slower or can not happen at all. This phenomena can be observed in figures 5 and 6. In 5 the coupling strengths take the values  $g_{12} = g_{21} = .01$ , in this case we observe a clear synchronization of the neuron activity. In figure 6 the coupling strengths take the values  $g_{12} = g_{21} = .001$ , in this case synchronization is not achieved. For both experiments the parameters are the same than the ones used in figure 3.

## 4 Results and Conclusions

The following results and conclusions can be established.

- A piecewise linear two-dimensional map that shows biophysical properties such as spiking or bursting with a low computational effort is presented.
- The map presents a response to external input similar to real neurons or differential models.
- The map can be generalized in order to accept input from other neurons in the network.
- Coupled maps present a regime of synchronization when enough coupling strength is considered. Low coupling strength yields to no (or slow) synchronization.

We thank the Ministerio de Ciencia y Tecnología (BFI-2003-07276). (PP) and (CA) are partially supported by BFM2002-02359. (PP) and (CA) also receive a partial support by POCTI/MAT/40706/2001. (ES) is partially supported by TIN2004-04363-C03-03.

## References

1. Izhikevich, E. M. Neural Excitability, Spiking and Bursting, *International Journal of Bifurcation and Chaos* **10** (2000) 1171–1266.
2. Hodgkin, A. L. and Huxley, A. F., A quantitative description of membrane current and application to conduction and excitation in nerve, *Journal of Physiology* **117** (1954) 165–181.
3. Rose, R. M. and Hindmarsh, J. L., The assembly of ionic currents in a thalamic neuron, I The three dimensional model. *Proceedings of The Royal Society of London B*, **237** (1989) 267–288.
4. Rulkov, N. F., Modeling of spiking-bursting neural behavior using two-dimensional map, *Physical Review E* **65** 041922 (2002).
5. Izhikevich, E. M. Simple Model of Spiking Neurons *IEEE Trans. on Neural Networks* **68** 052901 (2003).
6. Elson, R. C. et al, Synchronous Behavior of Two Coupled biological Neurons. *Physical Review Letters* , 81 (25), 5692 (1998)

# Real-Time Spiking Neural Network: An Adaptive Cerebellar Model

Christian Boucheny<sup>1</sup>, Richard Carrillo<sup>2</sup>, Eduardo Ros<sup>2</sup>,  
and Olivier J.-M.D. Coenen<sup>1</sup>

<sup>1</sup> Sony Computer Science Laboratory Paris,  
75005 Paris, France

{boucheny, coenen}@csl.sony.fr

<sup>2</sup> Department of Computer Architecture and Technology,  
E.T.S.I. Informática, University of Granada,  
18071 Granada, Spain

{rcarrillo, eduardo}@atc.ugr.es

**Abstract.** A spiking neural network modeling the cerebellum is presented. The model, consisting of more than 2000 conductance-based neurons and more than 50 000 synapses, runs in real-time on a dual-processor computer. The model is implemented on an event-driven spiking neural network simulator with table-based conductance and voltage computations. The cerebellar model interacts every millisecond with a time-driven simulation of a simple environment in which adaptation experiments are setup. Learning is achieved in real-time using spike time dependent plasticity rules, which drive synaptic weight changes depending on the neurons activity and the timing in the spiking representation of an error signal. The cerebellar model is tested on learning to continuously predict a target position moving along periodical trajectories. This setup reproduces experiments with primates learning the smooth pursuit of visual targets on a screen. The model learns effectively and concurrently different target trajectories. This is true even though the spiking rate of the error representation is very low, reproducing physiological conditions. Hence, we present a complete physiologically relevant spiking cerebellar model that runs and learns in real-time in realistic conditions reproducing psychophysical experiments. This work was funded in part by the EC SpikeFORCE project (IST-2001-35271, [www.spikeforce.org](http://www.spikeforce.org)).

## 1 Introduction

Recently, an event-driven neural simulator has been devised that permits the simulation of thousands of conductance-based spiking neurons in real-time. The speed gain comes from precomputing into large lookup tables the differential equations governing neuron and synapse dynamics [1]. We demonstrate here how we used this simulator to run in real-time a realistic model of the cerebellum in a complete action-perception experimental setup. The model contains thousands of neurons and features a biologically plausible spike time dependent learning rule.

The cerebellar model is tested on learning in real-time to continuously predict the position of a target moving along periodical trajectories. This setup reproduces experiments with primates learning the smooth pursuit of visual targets on a screen [2]. When the eye is not on target, a retinal error signal is generated that drives neurons in the inferior olive of the cerebellum. These neurons activity directs plasticity in the cerebellum, hence modifying the eye movements to keep it on the visual target. Their low firing rate [3] is puzzling since it does not provide in one trial an accurate rendering of the retinal error. Nevertheless, a non-deterministic generation of spikes in these neurons is capable to sample, over multiple trials, the complete error range [4]. Hence, for the first time, a reliable performance in smooth pursuit is demonstrated with a complete model of the cerebellum with such a low firing rate encoding of the error.

The model and the architecture of the real-time simulating platform are presented in Sec. 2. The supervised learning mechanisms implemented in the network are explicit in Sec. 3. The performance of the model is shown in Sec. 4. The computational efficiency of the simulation framework allowed us to explore and demonstrate the stability of the learning mechanisms employed in the model over long periods of time.

## 2 Architecture of the Model

### 2.1 Cerebellar Network and Simulation Setup

The model described hereafter and shown in Fig. 1 reproduces the known anatomy of the cerebellum (see Ito [3] for a review).

The mossy fibers (MF) are the inputs to the model, carrying sensory information about the target: its lateral and longitudinal positions and velocities are encoded by 4 groups of 10 MF each. This information is transformed into a sparse representation in the large granule cell (GC) layer, in which each cell receives an excitatory connection from one randomly chosen neuron in each of the 4 MF groups. The granule cells then form with the Golgi cells (GO) an excitatory-inhibitory loop that limits the global activity in the granular layer.

The Purkinje cells, divided into 4 arrays of 4 neurons, receive excitatory inputs from all the GC axons, called parallel fibers (PF), and a connection from a single inferior olive (IO) neuron. The conductances of the PF-PC synapses are chosen randomly at the beginning of the simulation, and modified online during the learning of the task according to the rules defined in Sec. 3.

The output of the model is made of 4 cerebellar nucleus (CN) neurons, whose activity provides a prediction of future target position (the prediction is ahead by  $\tau_{pred}$ , set to 200 ms). Two neurons code for the predicted lateral position of the target  $X_{pred}$ , one for the ‘right’ position ( $Cn_{X_+}$ ), and the other for the ‘left’ position ( $Cn_{X_-}$ ). Similarly, the other two neurons code for the predicted longitudinal position  $Y_{pred}$ : ‘up’ ( $Cn_{Y_+}$ ) and ‘down’ ( $Cn_{Y_-}$ ). If  $Cn_i(t)$  is the activity of cell  $i$  at time  $t$ , computed from its spike train, the predicted position of the target at time  $t + \tau_{pred}$  is given by:



$$X_{pred}(t + \tau_{pred}) = Cn_{X_+}(t) - Cn_{X_-}(t) \quad (1)$$

$$Y_{pred}(t + \tau_{pred}) = Cn_{Y_+}(t) - Cn_{Y_-}(t) \quad (2)$$

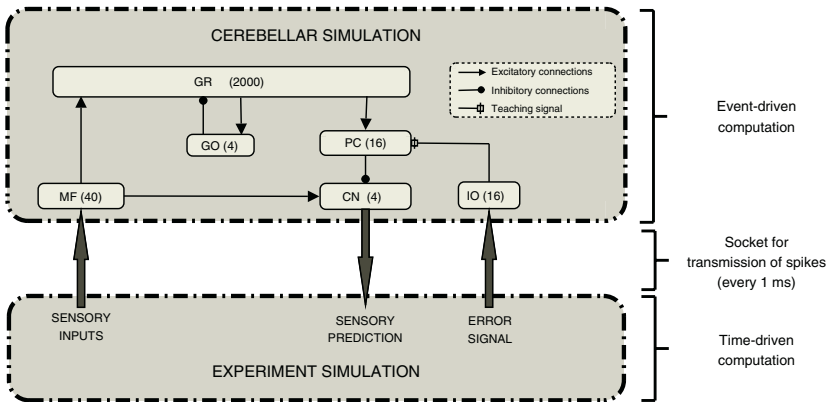
Each CN neuron receives excitatory connections from all the mossy fibers, and inhibitory afferents from the 4 PCs of the same cluster (the PC-CN-IO subcircuit is organised in 4 microzones corresponding to the 4 variables coded by CN cells).

The inferior olive neurons carry the error signal that is used for the supervised learning of PF-PC synaptic conductances (see Sec. 2.2). Different sites of plasticity exist in the cerebellum [5], but the present model focuses on PF-PC plasticity, other synaptic conductances being fixed.

The spiking neurons of the network are simulated with a computationally efficient table-based event-driven simulator [1]. Such a paradigm is particularly suited for a cerebellar model in which sparse activity is expected in the numerous neurons of the granular layer [6]. Plasticity mechanisms have also been implemented in an event-driven scheme to permit online learning.

To close the sensorimotor loop, a time-driven simulation of the experimental setup was run in parallel (time step: 1 ms). It is used both to emulate the experimental world and to transform analog signals into spikes and vice-versa: conversion of sensory (target-related) and error signals into MF and IO spikes, and extraction of the target position prediction from CN spike trains.

As shown in Fig. 1, the two simulators communicate every millisecond through TCP/IP sockets that transmit buffered spikes.



**Fig. 1.** Simulation set-up. The cerebellar model (top) described in Sec. 2.1 is run using an event-driven approach. It communicates through sockets with a time-driven simulator that emulates the experimental setup (movement of the target), generates the input spikes of the neural network and decodes cerebellar output spike trains. Numbers in parentheses in the cerebellar model diagram (top) represent the number of neurons per layer. MF: mossy fibers, GR: granule cells, GO: Golgi cells, PC: Purkinje cells, CN: cerebellar nucleus neurons, IO: inferior olive neurons

## 2.2 Models of Neurons

Mossy fibers are implemented as leaky integrate-and-fire neurons with the membrane potential  $U_m$  defined by the following equation [7]:

$$\tau_m \frac{dU_m}{dt} = -U_m(t) + RI(t) \quad (3)$$

When  $U_m$  reaches the threshold value  $U_{thresh}$ , the neuron fires and  $U_m$  is reset to the value  $U_{reset} < U_{thresh}$ . The input currents  $I(t)$  are computed by applying a radial basis function (RBF) to the coded sensory variable. The RBF centers are evenly distributed across the sensory dimensions, and their widths are chosen to ensure small response overlaps from consecutive mossy fibers.

The inferior olive neurons synapse onto the Purkinje cells and contribute to direct the plasticity of PF-PC synapses, which is believed to play a critical role in the learning of smooth and accurate movements [3]. These neurons, however, fire at very low rates (less than 10 Hz), which appears problematic to capture the high-frequency information of the error signal related to the task being learned. This apparent difficulty may be solved by their irregular firing, which can be exploited to statistically sample the entire range of the error signal over multiple trials [4]. This irregular firing is reproduced by means of a Poisson model of spike generation: if an IO neuron receives a normalised current  $I_{IO}(t)$  at time  $t$ , it will fire in the following time step if  $I_{IO}(t) > rand(t)$ , where  $rand(t)$  is a random number between 0 and 0.01 generated for the neuron at each time step (0.01 was chosen to limit the firing rate to 10 Hz).

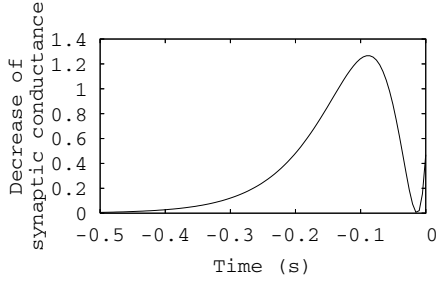
Other cells are implemented as one-compartment conductance-based neurons, whose states are precomputed [1]. Parameters for the granule cells, Purkinje cells and Golgi cells were taken from electrophysiological recordings [1].

## 3 Learning

### 3.1 Plasticity Mechanisms

The main plasticity mechanism observed at the PF-PC synapse is long-term depression (LTD, i.e. a decrease in synaptic conductance) [3]. It is mainly triggered by the spikes of the IO cell contacting the corresponding PC, and depends upon past activity of the afferent PF. More precisely, a PF spike will induce more LTD if it occurs between 200 and 50 ms before the IO spike. To reproduce this, we implemented an ‘eligibility trace’ [2] for each plastic synapse by means of a kernel function  $K(t)$  applied to the spike train of the PF. The amplitude of LTD triggered by a IO spike at time  $t_{IO}$  is given by:  $\int_{-\infty}^{t_{IO}} K(t - t_{IO})\delta_i(t)dt$ , where  $\delta_i(t) = 1$  if the  $i^{th}$  PF fires at time  $t$ ,  $\delta_i(t) = 0$  otherwise. To fit the event-driven simulation framework, the kernel function was implemented as the weighted sum of three exponential functions with different time decays (Fig. 2).

Recent studies [8] have shown that cerebellar LTD can be reversed when PF spikes are followed neither by IO spikes nor by a strong depression of the PC



**Fig. 2.** Kernel used for PF-PC synaptic long-term depression. The kernel is convolved with the spike train of the afferent PF. This provides a measure of past PF activity setting the eligibility of the synapse to depression when the IO neuron afferent to the PC emits a spike ( $t = 0$ )

membrane potential. These findings were implemented by a non-associative plasticity rule: each time the presynaptic PF fires, the PF-PC synapse conductance is increased by a fixed amount  $\delta_{LTP}$ .

### 3.2 Generating the Error Signal

The input current driving the IO neuron is computed by comparing the predicted and the actual position of the target (prediction error). The error encoding by the inferior olive neurons follows the visual target position encoding by the CN neurons. To describe this error signal, let us consider the IO neuron  $i$  connected to a PC inhibiting the  $CN_{X_+}$  neuron (the one coding for prediction of positive lateral position of the target). Its input current at time  $t$ ,  $I(t)$ , is derived from the activity of  $CN_{X_+}$  at time  $t - \tau_{pred}$ , which represents the prediction for the positive lateral target position at time  $t$  ( $X_{pred+}(t) = CN_{X_+}(t - \tau_{pred})$ ), and from the actual lateral position of the target  $X_{target}(t)$ , according to the following equations:

$$\text{if } X_{target}(t) < 0, \quad I(t) = \alpha (0.15 - a X_{pred+}(t)) \quad (4)$$

$$\text{if } X_{target}(t) \geq 0, \quad I(t) = \alpha (0.15 + b (X_{target}(t) - X_{pred+}(t))) \quad (5)$$

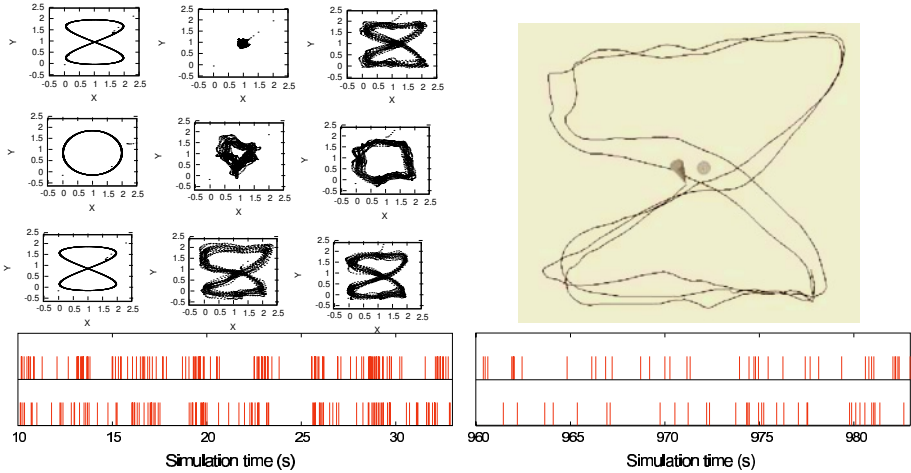
where  $a$ ,  $b$  and  $\alpha$  are normalisation factors limiting the IO firing rate below 10 Hz (see Sec. 2.2). The value 0.15 was chosen to set the mean IO firing rate at 1.5 Hz for zero error, as observed experimentally.

The subtractive part of Eq. 4 represents formally the inhibition of the IO by CN neurons, which has been observed experimentally and suggested to be necessary for the reversal of non-relevant learning [9]. If the IO input current  $I(t)$  is close to 0, then LTP is likely to dominate LTD at the PF-PC synapse; if it is close to 1, then the contrary holds. The converse equations hold for the computation of the contralateral error (i.e. regarding  $CN_{X_-}$ ), and the same system of equations is used to generate the ‘up’ and ‘down’ error signals.

## 4 Experimental Task and Results

Psychophysical experiments have shown that monkeys can achieve accurate smooth pursuit eye tracking of periodically moving targets with hardly no phase lag, whereas delays in the visual pathway can reach over 100 ms. The flocculus and paraflocculus cerebellar areas participate to the control of smooth pursuit eye movements: they learn to produce movement predictions that compensate for the feedback delays in the eye control loop [2]. The present model is based on these findings and focuses on the ability of a cerebellar network to build short term predictions with physiologically constrained spiking learning mechanisms. A target moves along periodical trajectories and the model learns to build a continuous short-term prediction of the target position (Fig. 3, top right). At each time step, the cerebellar model is given the position and velocity of the target, and the CN neuron activities at time  $t$  are interpreted to compute the prediction of the target absolute position at time  $t + \tau_{pred}$  ( $\tau_{pred} = 200$  ms). The learning rules described in Sec. 3 are applied continuously.

The simulation consists of three consecutive stages of 1000 s. The target first describes an ‘8-shape’ trajectory with a period of  $2\pi$  s, then a circle with the



**Fig. 3.** Cerebellar model results. Top left: Performance of the model during three stages of 1000 s: ‘8-shape’ target trajectory (top), followed by a circle (middle) and finally an ‘8-shape’ again (bottom). The left column shows the target trajectories, the center and right columns represent, respectively, the predictions of the model at the beginning and the end of each stage. Top right: Screenshot taken at the end of the simulation. The target to follow is the ball, and the cone represents the delayed cerebellar sensory prediction. The lines are traces of past cerebellar output. Bottom: Spike trains of two IO cells coding for the same error signal, at the beginning of the simulation (left) and at the end of the simulation (right). When learning has stabilised, these cells fire at 1 to 2 Hz, whereas when the error is important frequencies up to 10 Hz can be observed

same period and finally the ‘8-shape’ trajectory again. This last stage seeks to test the possible destructive interferences brought by the second learning stage.

The performance of the model in the predictive task of smooth pursuit is illustrated in Fig. 3. The mean prediction error decreases from  $\eta = 0.6$  at the beginning of the simulation to  $\eta = 0.1$  after 3000 s. The spike trains of two IO cells related to  $CN_{X+}$  output neurons are also shown for the beginning and the end of the first stage of the simulation. The model learned to follow the given trajectory and little interference occurred when changing from one trajectory to another. Learning occurred mainly during the first 500 s of each stage and plateaued after this initial period. Indeed, the matrix of PF-PC weights remained stable afterwards, with only small local variations (not shown).

The simulations were run on a dual-processor computer (Intel(R) Xeon(TM) CPU 2.80 GHz). There were 2080 neurons in the network and more than 52000 synaptic connections. During one second of simulation, the cerebellar network received an average of 395 spikes, sent 405 spikes in output, and processed 935 801 events. One stage of the simulation (1000 s) lasted approximately 280 s, which revealed to be better than real-time. Hence, on this machine, there is room to scale up the size of the network, the main limiting factor being the global level of activity in the network.

## 5 Discussion

To our knowledge, the model presented here is the first to implement a complete and physiologically relevant spiking cerebellar network running in real-time and learning online to build sensory predictions in a closed sensorimotor loop. Good performances were reached despite the physiologically realistic low firing of the inferior olive (less than 10 Hz). This indeed suggests that one of the processing of the inferior olive is to sample non-deterministically the input signals it receives in order to provide over time a complete representation of these signals to the plasticity mechanisms at the Purkinje cells [4, 10]. The model extends many previous cerebellar models. Kettner et al. modeled the learning of predictive commands for smooth pursuit eye movements [2]. Their model featured an eligibility trace to compensate for visual feedback delays, but relied exclusively on analog units. Schweighofer et al. [11] and Spoelstra et al. [12] used the cerebellum to produce predictive corrections in an arm reaching task. Analog units were also used there, except for the IO cells modeled as leaky integrate and fire spiking neurons with low firing rates. However, the IO spikes were deterministically drawn from the error signal and the LTP at PF-PC synapse was modeled as a weights normalisation process. In a similar arm task, Kuroda et al. [4] showed that the stochastic firing of IO neurons increased performances; yet they implemented a very simplified cerebellar model and did not take time delays into account. In all the arm reaching simulations above, the learning took place per trial and not continuously and was implemented off-line, so that the output of the cerebellum could not interfere with the motor command during the learning stage. Medina et al. [13, 9] also implemented a spiking model of

the cerebellum, but they focused on eyeblink conditioning experiments and used coincidence-based learning rules.

Whereas with previous simulators hours or days were required to simulate a spiking cerebellar model learning smooth pursuit, the current event-driven simulator with online plasticity mechanisms permitted to achieve real-time learning of the task. This allowed us to show that the biologically plausible learning mechanisms running continuously over long periods of time were stable and that few destructive interferences occurred in learning the same task in different sensorimotor contexts (different target trajectories). Future simulation results will permit a thorough analysis of this evidence with more diverse contexts.

The current simulation framework demonstrates that a realistic spiking cerebellar model running in real-time in a complete action-perception sensorimotor loop is becoming reality. Future robotic experiments will sustain that claim.

## References

1. Carrillo, R.R., Ros, E., Ortigosa, E.M., Barbour, B., Agís, R.: Lookup Table Powered Neural Event-Driven Simulator. In: Proc. of the Eighth Int. Work-Conf. on Artif. Neural Networks, Springer-Verlag, LNCS series (2005)
2. Kettner, R.E., Mahamud, S., Leung, H., Sittkoff, N., Houk, J.C., Peterson, B.W., Barto, A.G.: Prediction of complex two-dimensional trajectories by a cerebellar model of smooth pursuit eye movement. *Journal of Neurophysiology* **77**(4) (1997) 2115–2130
3. Ito, M.: Cerebellar long-term depression: characterization, signal transduction, and functional roles. *Physiological Reviews* **81**(3) (2001) 1143–1195
4. Kuroda, S., Yamamoto, K., Miyamoto, H., Doya, K., Kawato, M.: Statistical characteristics of climbing fiber spikes necessary for efficient cerebellar learning. *Biological Cybernetics* **84** (2001) 183–192
5. Hansel, C., Linden, D.J., D’Angelo, E.: Beyond parallel fiber LTD: the diversity of synaptic and non-synaptic plasticity in the cerebellum. *Nature Neuroscience* **4**(5) (2001) 467–475
6. Coenen, O.J.M.D., Arnold, M.P., Sejnowski, T.J., Jabri, M.A.: Parallel fiber coding in the cerebellum for life-long learning. *Autonomous Robots* **11**(3) (2001) 291–297
7. Gerstner, W., Kistler, W.M.: Spiking neuron models. Cambridge University Press (2002)
8. Lev-Ram, V., Mehta, S.B., Kleinfeld, D., Tsien, R.Y.: Reversing cerebellar long-term depression. *Proceedings of the National Academy of Sciences* **100**(26) (2003) 15989–15993
9. Medina, J.F., Nores, W.L., Mauk, M.D.: Inhibition of climbing fibres is a signal for the extinction of conditioned eyelid responses. *Nature* **416** (2003) 330–333
10. Schweighofer, N., Doya, K., Fukai, H., Chiron, J.V., Furukawa, T., Kawato, M.: Chaos may enhance information transmission in the inferior olive. *Proceedings of the National Academy of Sciences* **101** (2004) 4655–4660
11. Schweighofer, N., Arbib, A.A., Kawato, M.: Role of the cerebellum in reaching movements in humans. II. A neural model of the intermediate cerebellum. *European Journal Of Neuroscience* **10** (1998) 95–105

12. Spoelstra, J., Schweighofer, N., Arbib, M.A.: Cerebellar learning of accurate predictive control for fast-reaching movements. *Biological Cybernetics* **82** (2000) 321–333
13. Medina, J.F., Mauk, M.D.: Simulations of cerebellar motor learning: computational analysis of plasticity at the mossy fiber to deep nucleus synapse. *The Journal of Neuroscience* **19(16)** (1999) 7140–7151

# Modeling Neural Processes in Lindenmayer Systems

Carlos Martín-Vide and Tseren-Onolt Ishdorj

Research Group on Mathematical Linguistics,  
Rovira i Virgili University,  
Pl. Imperial Tàrraco 1, 43005 Tarragona, Spain  
`carlos.martin@urv.net`  
`tserenonolt.ishdorj@estudiants.urv.es`

**Abstract.** Computing in nature as is the case with the human brain is an emerging research area in theoretical computer science. The present paper's aim is to explore biological neural cell processes of interest and to model them with foundational concepts of computer science. We have started by discovering and studying certain primitive symbolic neural operations of neuron functions, and we have formalized them with Lindenmayer (L) systems.

## 1 Introduction

So far, the investigation of computing and learning in neural systems, or neural computation, with a good number of theoretical contributions, has been dominated by approaches from theoretical physics, information theory, and statistics [5].

Our research interest in biological neural systems is mainly focused on exploring symbolic models of neural systems through the foundations of computer science. More precisely, we attempt to explore the architecture of biological neural systems by modeling and simulating neural processes (as realistic ones as possible) with existing language-theoretic devices and bio-inspired computing techniques. We have investigated the modeling of neural processes studying, for the moment, the behavior of a single neuron concerning both its structure and some functions, as in [1], [2], [7], [8], etc.

In the present paper, we explore the potentialities of Lindenmayer (L) systems for the study of neural processes, particularly impulse transmission and propagation through myelinated axon and axonal terminal tree of the neuron. We show that PDOL language sequences and length sequences of PDOL languages are thus generated.

The common features shared by L systems and neural systems, both of them having distributed architectures and working in a massively parallel manner, are a good starting point for the designing of symbolic models.



## 2 Preliminaries

We briefly mention next some notions, notations, and results from L systems as well as some biological neural phenomena that will be used throughout the paper.

### 2.1 Biological Neural Systems

The basic transmitting unit in the nervous system is brain cells, so called *neurons* [3], [6]. The neuron is not one homogeneous integrative unit but is (potentially) divided in many sub-integrative units, each one with the ability of mediating a local synaptic output to another cell or a local electro-tonic output to another part of the same cell.

Neurons are considered to consist of three main parts: a *soma*, the main part of the cell where the genetic material is present and life functions take place; a *dendrite tree*, the branches of the cell from where the impulses come in; and an *axon*, the branch of the neuron over which the impulse (or signal) is propagated. The branches present at the end of the axons are called *terminal trees*. An axon can be provided by a structure composed by special sheaths. These sheaths are involved in molecular and structural modifications of axons needed to propagate impulse signals rapidly over long distances. The impulse in effect jumps from node to node, and this form of propagation is therefore called *saltatory conduction*. There is a gap between neighboring myelinated regions that is known as the *node of Ranvier*, which contains a high density of voltage-gated  $Na^+$  channels for impulse generation. When the transmitting impulses reach the node of Ranvier or junction nodes of dendrite and terminal trees, or the end bulbs of the trees, it causes the change in polarization of the membrane. The change in potential can be excitatory (moving the potential toward the threshold) or inhibitory (moving the potential away from the threshold).

The complexity revealed by modern research has widened with several new concepts. One of them is that the postsynaptic terminal may send retrograde signals to the presynaptic terminal. Because of this, the synapse can be viewed as having a *bidirectional* nature.

More details about neural biology can be found in [11].

### 2.2 Lindenmayer Systems

In 1968, A. Lindenmayer introduced a formalism for modeling and simulating the development of multicellular organisms [4], subsequently named *L systems*. This formalism was closely related to the theory of automata and formal languages, and immediately attracted the attention of computer scientists [10]. The development of the theory of L systems was followed by its application to the modeling of plants. After a vigorous initial research period, some of the resulting language families, notably the families of DOL, OL, DTOL, EOL and ETOL languages, emerged as fundamental ones. Indeed, nowadays the fundamental L families constitute a

testing ground similar to the Chomsky hierarchy when new devices (grammars, automata, etc.) and new phenomena are investigated in language theory.

We recall here some formal definitions and classes of L systems.

**Definition 1.** A finite substitution  $\sigma$  over an alphabet  $\Sigma$  is a mapping of  $\Sigma^*$  into the set of all finite nonempty languages (over an alphabet  $\Delta$ ) defined as follows. For each letter  $a \in \Sigma$ ,  $\sigma(a)$  is a finite nonempty language,  $\sigma(\lambda) = \lambda$  and, for all words  $w_1, w_2 \in \Sigma^*$ ,  $\sigma(w_1 w_2) = \sigma(w_1)\sigma(w_2)$ . If none of the languages  $\sigma(a)$ ,  $a \in \Sigma$ , contains the empty word, the substitution  $\sigma$  is referred to as  $\lambda$ -free or nonerasing. If each  $\sigma(a)$  consists of a single word,  $\sigma$  is called a morphism. We speak also of nonerasing and letter-to-letter morphisms.

**Definition 2.** A 0L system is a triple  $G = (\Sigma, \sigma, w_0)$ , where  $\Sigma$  is an alphabet,  $\sigma$  is a finite substitution on  $\Sigma$ , and  $w_0$  (referred to as the axiom) is a word over  $\Sigma$ . A 0L system is propagating, or a P0L system, if  $\sigma$  is nonerasing. The 0L system  $G$  generates the languages  $L(G) = \{w_0\} \cup \sigma(w_0) \cup \sigma(\sigma(w_0)) \cup \dots = \bigcup_{i \geq 0} \sigma^i(w_0)$ .

**Definition 3.** A 0L system  $G = (\Sigma, \sigma, w_0)$  is deterministic, or a D0L system, if  $\sigma$  is a morphism.

**Definition 4.** Let  $G = (\Sigma, h, w_0)$  be a D0L system (we use the notation  $h$  to indicate that we are dealing with a morphism). The system  $G$  generates its language  $L(G)$  in a specific order, as a sequence:  $w_0, w_1 = h(w_0), w_2 = h(w_1) = h^2(w_0), w_3, \dots$ . We denote the sequence by  $S(G)$ . Thus, in connection with a D0L system  $G$ , we speak of its language  $L(G)$  and its sequence  $S(G)$ . D0L systems are propagating, that is PD0L systems, if  $\sigma$  is nonerasing.

**Definition 5.** Given a D0L system  $G = (\Sigma, h, w_0)$ , the function  $f_G : \mathbb{N} \rightarrow \mathbb{N}$  defined by

$$f_G(k) = |h^k(w_0)|, n \geq 0 \quad (1)$$

is called the growth function of  $G$ , and the sequence

$$F_{G,k} = |h^k(w_0)|, k = 0, 1, 2, \dots \quad (2)$$

is called its growth sequence. Functions of the form (1) are called D0L growth functions (resp. PD0L growth functions if  $G$  is a PD0L system). Number sequences of the form (2) are called D0L (resp. PD0L) length sequences.

### 3 L Systems in Modeling Neural Cell Processes

Traditionally, Lindenmayer systems are grammatical models of the development of multi-cellular organisms, which consider cells as basic atomic objects. Therefore, in Lindenmayer systems, the cells are identified by symbols. In the current section, we try to focus on zooming a neural cell. We are interested in the cell structure, and in the chemical operations that are present in the cell. We are

also focused on modeling and simulating processes of the neuron in L systems. The chemicals inside cells, not “cells” themselves, will be atomic entities in our model. We consider neural cell processes - impulse transmission and propagation through myelinated axon and axonal terminal tree of the neuron - as our domain of interest and show that essential aspects of impulse transmission can be viewed as parallel rewriting processes.

### 3.1 Impulse Transmission of Axon and DOL Systems

The neural cell’s axon structure can be mathematically represented by a linear undirected graph with  $n$  nodes labeled injectively by  $1, 2, \dots, n$  from the ancestor node until the end node in ascending order, as illustrated in Figure 1.

Let us denote an impulse by  $a_i$ , the subscript  $i$  indicating which impulse sends/reaches from/into  $i$ -th node of the axon. The excite impulse  $a_1$  is transmitted from the ancestor node with label 1 into the target node with label 2 while the impulse evolves to  $a_2$ , then it replicates into two distinguished impulses  $a_1$  and  $a_3$ . Those impulses are transmitted to the adjacent nodes 1 and 3, and so on, the impulses being spreaded through the axon by jumping from node to node.

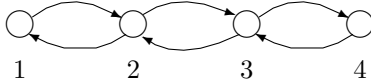


Fig. 1. Myelinated Axon Illustration

Sending the impulse  $a_1$  from node 1 to node 2, and sending back the impulse  $a_n$  from node  $n$  to node  $n - 1$ , are described by the morphisms  $h(a_1) = a_2$  and  $h(a_n) = a_{n-1}$ , respectively. The transmission of impulses from internal nodes is described as  $h(a_i) = a_{i-1}a_{i+1}$ , meaning that the impulse  $a_i$  replicates into  $a_{i-1}$  and  $a_{i+1}$  at the node  $i$ , and they are transmitted to the adjacent nodes  $i - 1$  and  $i + 1$ , respectively.

We send a unique exciting impulse  $a_1$  from the ancestor node into the axon with  $i$  number of nodes. We are interested in the properties held by the firing axon as impulse propagation, spread, transmission through long distance (nodes), paying special attention to sequences of PDOL.

Let us consider a sequence of PDOL systems  $G_i = (\Sigma_i, h_i, a_1)$ , with  $\Sigma_i = \{a_1, \dots, a_i\}$ , each of  $G_i$  having morphisms  $h_i(a_1) = a_2$ ,  $h_i(a_k) = a_{k-1}a_{k+1}$ ,  $h_i(a_i) = a_{i-1}$ ,  $2 \leq k \leq i - 1$ ,  $1 \leq i \leq n$ , and the same axiom  $a_1$ . The subscript  $i$  indicates the number of nodes of the axon. Thus,  $G_3 = (\{a_1 a_2 a_3\}, h_3, a_1)$ . The first few words in the sequence  $S(G_3)$  are  $a_1, a_2, a_1 a_3, a_2 a_2, a_1 a_3 a_1 a_3, a_2 a_2 a_2 a_2, a_1 a_3 a_1 a_3 a_1 a_3 a_1 a_3, a_2 a_2 a_2 a_2, \dots$ . Hence, the length sequence is  $1, 1, 2, 2, 4, 4, 8, 8, \dots$ . On the other hand,  $G_4 = (\{a_1 a_2 a_3 a_4\}, h_4, a_1)$ . The sequence  $S(G_4)$  begins with words  $a_1, a_2, a_1 a_3, a_2 a_2 a_4, a_1 a_3 a_1 a_3 a_3, a_2 a_2 a_4 a_2 a_2 a_4 a_2 a_4, a_1 a_3 a_1 a_3 a_3 a_1 a_3 a_3 a_1 a_3 a_3 a_1 a_3 a_3$ . The length sequence is the well-known *Fibonacci sequence*  $1, 1, 2, 3, 5, 8, 13, \dots$ .

**Proposition 1.** *Alphabets of each adjacent words in a sequence  $S(G_i)$  are completely different:  $\text{alph}(w_j) \cap \text{alph}(w_{j+1}) = \emptyset$ ,  $j \geq 1$ .*

**Proposition 2.** *Alphabets of words in even positions and in odd positions in a sequence  $S(G_i)$  coincide, respectively:  $\text{alph}(w_{i+j}) = \text{alph}(w_{i+j+2})$ ,  $j \geq 1$ .*

Between the sequences of PD0L systems  $G_i$ ,  $3 \leq i \leq n$ , the next properties hold.

**Proposition 3.**  *$G_i$  is obtained from previous grammars  $G_j$ ,  $j \leq i$ , by using the following iterated morphism relations:*

$$\begin{aligned} h_i(a_k) &= h_{i-1}(a_k), \quad 1 \leq k \leq i-2, \\ h_i(a_{i-1}) &= h_{i-1}(a_{i-1})a_i, \\ h_i(a_i) &= a_{i-1}, \quad 3 \leq i \leq n. \end{aligned} \quad (3)$$

Word length sequences  $|w_0|, |w_1|, \dots$  obtained from a D0L sequence  $S(G) = w_0, w_1, w_2, \dots$  determine in a natural way the function  $f$  such that the value  $f(n)$ ,  $n \geq 0$ , is defined to be the number  $|w_n|$  in the sequence. This function is referred to as the growth function of the D0L system  $G$ . Thus, when studying growth functions, we are not interested in the words themselves but only in their lengths.

**Proposition 4.** *The numbers of occurrences of symbols in each word in the sequence are presented in the next recurrent formula. If  $f_{G_n,k}(a_i)$  is the number of occurrences of symbol  $a_i$  in the  $k$ -th word of the sequence  $S(G_n)$ , then*

$$\begin{aligned} f_{G_n,1}(a_1) &= 1, f_{G_n,1}(a_i) = 0, \quad 2 \leq i \leq n, \\ f_{G_n,k}(a_1) &= f_{G_n,k-1}(a_2), \\ f_{G_n,k}(a_i) &= f_{G_n,k-1}(a_{i-1}) + f_{G_n,k-1}(a_{i+1}), \quad 2 \leq i \leq n-1, \\ f_{G_n,k}(a_n) &= f_{G_n,k}(a_{n-1}), \quad k \geq 2, n \geq 3. \end{aligned} \quad (4)$$

**Proposition 5.** *The sum of occurrences of each symbol in the  $k$ -th word of  $S(G_n)$ ,  $n \geq 3$ , (i.e. the length of that word) is:*

$$F_{G_n,k} = f_{G_n,k}(a_1) + f_{G_n,k}(a_2) + \dots + f_{G_n,k}(a_n). \quad (5)$$

**Proposition 6.** *The  $k$ -th member  $F_{G_n,k}$  of the length sequence  $|S(G_n)|$  is described by the previous two members in the general formula*

$$\begin{aligned} F_{G_n,1} &= F_{G_n,2} = 1, \\ F_{G_n,k} &= F_{G_n,k-1} + F_{G_n,k-2} + \sum_{i=3}^{n-2} f_{G_n,k-2}(a_i), \\ |w_n,k| &= |w_n,k-1| + |w_n,k-2| + \sum_{i=3}^{n-2} f_{G_n,k-2}(a_i), \quad n \geq 3, k \geq 3. \end{aligned} \quad (6)$$

It is easy to claim, by using (4), that

$$\begin{aligned} F_{G_n,k} &= f_{G_n,k}(a_1) + \dots + f_{G_n,k}(a_n) = \\ &= f_{G_n,k-1}(a_1) + 2f_{G_n,k-1}(a_2) + \dots + 2f_{G_n,k-1}(a_{n-1}) + f_{G_n,k-1}(a_n) = \\ &= F_{G_n,k-1} + F_{G_n,k-2} + \sum_{i=3}^{n-2} f_{G_n,k-2}(a_i). \end{aligned}$$

The next simple formulas follow from (6). If the graph (axon) has 4 nodes, the tail of (6) is always 0, and  $\sum_{i=3}^{n-2} f_{G_n, k-2}(a_i) = 0$ , then  $|S(G_4)|$  is the Fibonacci numbers.

$$F_{G_4, 1} = F_{G_4, 2} = 1, F_{G_4, k} = F_{G_4, k-1} + F_{G_4, k-2}, k \geq 3 \text{ or}$$

$$|w_{4, k}| = |w_{4, k-1}| + |w_{4, k-2}|, k \geq 3.$$

### 3.2 Impulse Transmission of the Axonal Terminal Tree and PD0L Systems

The axonal terminal tree can be mathematically represented by a tree with  $n$  levels. It is possible to specify impulse transmission operations through axonal terminal trees in L systems by parallel rewriting rules. The internal nodes of the tree are “hot spots”. A reached impulse at a “hot spot” is replicated and distributed into adjacent nodes. For instance, by rule  $2 \rightarrow 341$ , the impulse 2 is replicated into impulses 3, 4 and 1, and distributed to the children nodes 3 and 4, and parent node 1 of the node 2, respectively. In our formalization, from the root node of the tree, impulses are only replicated and distributed to the children nodes, however from the leaf nodes impulses are just transmitted back to the parent nodes without any replication.

Let us now denote an impulse as a number  $i$ . Impulses are replicated at a node and distributed only to the connected nodes as mentioned above. Here, we consider only a binary tree structure of the axonal trees.

As usual, we send a unique excite impulse 1 from the root node into the tree, thus we see the process of impulse propagation and spreading through the tree.

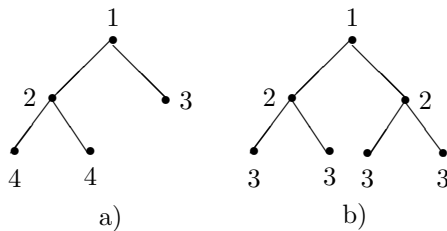


Fig. 2. Axonal Terminal Tree Structures

Consider a PD0L system  $G = (\Sigma, \mu, H, 1)$ , with alphabet  $\Sigma$ , axonal tree structure  $\mu$ , set of rewriting rules  $H$ , and axiom 1.

We consider the following examples of impulse transmission in certain types of a terminal tree.

Axonal terminal binary trees, illustrated in Fig. 2, are described as bracketed strings:

- Three-level trees:

$$\mu_a) 1[ 2[ 44] 3]$$

$\mu_b) 1[ 2[ 33] 2[ 33] ]$  is a complete binary tree with 3 levels.

- Four-level trees:

$$\mu_c) 1[ 2[ 4[ 55] 4[ 55] ] 3]$$

$\mu_d) 1[ 2[ 4[ 66] 4[ 66] 3[ 55] ] ]$  is a complete binary tree with 4 levels.

Corresponding rewriting rules of impulse transmission and propagation through the terminal tree structures as above are:

$$h_a) 1 \rightarrow 23, 2 \rightarrow 144, 3 \rightarrow 1, 4 \rightarrow 2$$

$$h_b) 1 \rightarrow 22, 2 \rightarrow 133, 3 \rightarrow 2$$

$$h_c) 1 \rightarrow 23, 2 \rightarrow 144, 3 \rightarrow 1, 4 \rightarrow 255, 5 \rightarrow 4$$

$$h_d) 1 \rightarrow 23, 2 \rightarrow 144, 3 \rightarrow 155, 4 \rightarrow 266, 5 \rightarrow 3, 6 \rightarrow 4$$

$G_a = (\{1234\}, \mu_a, h_a, 1)$ . The first few words in the sequence  $S(G_a)$  are

$$1, 23, 1441, 232223, 14411441441441, 23222323222322232223, \dots$$

Hence, the length sequence is 1, 2, 4, 6, 14, 20, 48, 68, 164,  $\dots$

The length sequence  $|S(G_a)|$  can be generated by the following formula:

$$w_1 = 1, w_2 = 2, \\ |w_n| = \begin{cases} |w_{n-1}| + |w_{n-2}|, & \text{if } n \text{ is even,} \\ |w_{n-1}| + 2|w_{n-2}|, & \text{if } n \text{ is odd, } n \geq 3. \end{cases}$$

A case of generating formula for the above integer sequence is found as:

Sequence: 0,1,1,2,4,6,14,20,48,68,164,232,560,792,1912,2704,6528,9232, 22288,31520,76096,107616,259808,367424,887040,1254464,

Name:  $a(0) = 0; a(1) = 1; a(n) = a(n - 1) + (3 + (-1)^n) * a(n - 2)/2$

Formula: G.f.:  $x(1 + x - 2x^2)/(1 - 4x^2 + 2x^4)$ .

Example:  $a(4) = a(3) + 2 * a(2) = 2 + 2 = 4$

Author: Olivier Gerard (ogerald(AT)ext.jussieu.fr), Jun 05 2001.

Let us consider a PD0L system  $G_b = (\{123\}, \mu_b, h_b, 1)$ . The first few words in the sequence  $S(G_b)$  are 1, 22, 133133, 22222222, 133133133133133133133,  $\dots$

Hence, the length sequence is 1, 2, 6, 8, 24, 32, 96,  $\dots$

The language generated by the grammar  $G_b$  is

$$L(G_b) = \{1\} \cup \{(22)^{2^{2n}}, (133)^{2^{2n+1}} \mid n \geq 0\}.$$

The length sequence  $|S(G_b)|$  can be generated by the following formula:

$$w_1 = 1, w_2 = 2, \\ |w_n| = \begin{cases} |w_{n-1}| + |w_{n-2}| = 2^{n-1}, & \text{if } n \text{ is even,} \\ 3|w_{n-1}| = 3 * 2^{n-2}, & \text{if } n \text{ is odd, } n \geq 3. \end{cases}$$

$G_c = (\{12345\}, \mu_c, h_c, 1)$ . The first few words in the sequence  $S(G_c)$  are

1,23,1441,2325525523,144114444144441441,

232552552323255255255255232552552552325525523,  $\dots$

Hence, the length sequence is 1, 2, 4, 10, 18, 48,  $\dots$

## 4 Final Remarks

We have tried to show that the application of L systems to neural process as impulse transmission and propagation through an axon with  $n$  nodes and terminal trees produce PD0L languages and length sequences of languages. The growth function for impulse propagation in the axon (5) tells us that the number of impulses propagated in the axon is growing in an exponential manner, and this number does not depend on the number of nodes in the axon. We have seen that, depending on the number of nodes in the axon, impulse transmission generates different types of sequences of numbers, but all of them share the common formula (6). Modeling the whole information transmission process in neurons, receiving impulses in the dendrite tree, transmitting them into the soma, processing impulses in the soma, transmitting the processed impulses into the terminal tree, finally releasing the impulses into the connected junctions of neurons in order to make neural connections, all of this could be simulated with L systems.

**Acknowledgments.** The work partially supported by the Asian Research Center, National University of Mongolia.

## References

1. Cavaliere, M., Ionescu, M., Ishdorj, T.-O., Inhibiting/De-inhibiting Rules in P Systems, Proceedings 5<sup>th</sup> Workshop on Membrane Computing, Milano, Italy, 2004, 60–73, and *Lecture Notes in Computer Science LNCS 3365*, 224–238, Springer, Berlin Heidelberg 2005.
2. Ishdorj, T.-O., Ionescu, M., Replicative-Distribution Rules in P Systems with Active Membranes, Proceedings First International Colloquium on Theoretical Aspects of Computing, Guiyang, China, September 20-24 2004- UNU/IIST Report No. 310, 263–278, Zhiming Liu (Ed.) Macau, and *Lecture Notes in Computer Science LNCS 3407*, 69–84, Springer, Berlin Heidelberg 2005.
3. Kleene, S.C., “Representation of Events in Nerve Nets and Finite Automata”. In *Automata Studies*, Princeton University Press, Princeton, NJ, 1956, 3–42.
4. Lindenmayer, A., Mathematical models for cellular interaction in development I and II. *Journal of Theoretical Biology* 18(1968), 280–315.
5. Maass, W., Neural Computation: A Research Topic for Theoretical Computer Science? Some Thoughts and Pointers, in [9].
6. McCulloch, W.S., Pitts, W.H., “A Logical Calculus of the Ideas Immanent in Nervous Activity”. *Bulletin of Mathematical Biophysics*, 5(1943), 115–133.
7. Pan, L., Alhazov, A., Ishdorj, T.-O., Further Remarks on P Systems with Active Membranes, Separation, Merging and Release Rules. *Soft Computing*, 8(2004), 1–5.
8. Pan, L., Ishdorj, T.-O., P Systems with Active Membranes and Separation Rules. *Journal of Universal Computer Science*, 10, 5(2004), 630–649.
9. Păun, Gh., Rozenberg, G., Salomaa, A.,(eds), *Current Trends in Theoretical Computer Science: Entering the 21<sup>st</sup> Century*, World Scientific, Singapore, 2001.
10. Rozenberg, G., Salomaa, A., *The Mathematical Theory of L Systems*, Academic Press, New York, 1980.
11. Shepherd, G.M., *Neurobiology*, Oxford University Press, Oxford, 1994.

# Modeling Stimulus Equivalence with Multi Layered Neural Networks

Hiroyuki Okada<sup>1</sup>, Masamichi Sakagami<sup>2</sup>, and Hiroshi Yamakawa<sup>3</sup>

<sup>1</sup> Tokai University, Kitakaname 1117,  
Hiratsuka City, Kanagawa 259-1292, Japan  
`okada@ss.u-tokai.ac.jp`

<sup>2</sup> Tamagawa University, Tamagawa Gauen 6-1-1,  
Machida City, Tokyo 194-8610, Japan

<sup>3</sup> Higashi Sumiyoshi 4-2, Tokorozawa City,  
Saitama 359-1124, Japan

**Abstract.** Prior studies showed that stimulus equivalence did not emerge in nonhuman and it may be what distinguish humans from non-humans. We think that stimulus equivalence is the origin of humanfs illogical reasoning.

For applying neural networks to stimulus equivalence, a problem of missing input features and self-supervised learning must be solved. In this paper, we propose a neural network model based on the iterative inversion method which has a potential possibility to explain the stimulus equivalence and demonstrated the validity of the proposed model by computer simulations. Furthermore, it was discussed that the proposed model was an appropriate model of symmetry for human reasoning.

## 1 Introduction

When solving the problem in the real world, a flexible human inferential ability corresponding to the situation demonstrates surprising power. Especially, the great feature that other animals do not have is to have a logical reasoning and the intuitive, non-logical reasoning by the human. It was requested to pay attention to the side of a human logical mind in the early artificial intelligence research, and to be programmed with the computer. However, the example that a man does not follow the right reasoning result logically is also observed various scenes in daily life by one side. Then, although the research which focused on the side of illogical thinking, such as fuzzy reasoning which formulized ambiguous reasoning of a man, prospers, the elucidation is not yet made.

In this paper, the stimulus equivalence known as a man's illogical cognitive bias thinks that it is the factor of illogical reasoning of a man. While stimulus equivalence considers the role played in a man's reasoning process and language acquisition process, the computational model is proposed.



## 2 Stimulation Equivalence and Non-logical Reasoning

The stimulation equivalence is a phenomenon of approving the derivative relations between stimulation it not is in the training case in the conditional discrimination task of the sample matching problem etc. It is acquired without the four remaining relations shown by the dotted line learning by learning only two relations expressed with a solid line among six relations shown by the arrow in Fig.1. The character string "banana" is presented showing the picture of the banana. A character string "banana" is shown and it speaks with "banana".

The stimulation equivalence is formulated with three character the symmetry, equivalence and equivalence (Sidman1982) (Sidman1994). In Fig.1, The picture of the banana is inferred showing character string "banana". When it is made to hear the "banana", character string "banana" is inferred(symmetry). The picture of a banana is shown and it speaks with "banana"(transitivity). The picture of a banana will be reasoned if "banana" is told (equivalence).

Stimulus equivalence is materialized in human being. It is known that it is difficult to materialize symmetry in animals other than a man especially in the composition element of stimulus equivalence.

In this paper, I think that it is the base of the flexible and illogical reasoning with acquisition of the symmetry containing such a logical mistake peculiar to a man. The author thinks that the role that the language plays is large though there are various opinions why a human inferential ability is as flexibly as this excellent.

There are various opinions of whether a human inferential ability is as flexibly as this excellent (Harlow1949) (Vaughan1988) (Delis1998). The author thinks that the role that the language plays is large. According to the restrictions theory of meaning-of-a-word study, a man does not search for and examine all

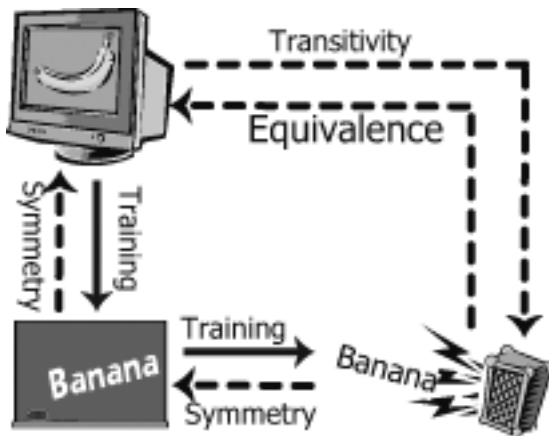


Fig. 1. Stimulus equivalence

the possibilities as a meaning of the word which met for the first time. It is known, although it had a belief about a concept suitable as a meaning of a word, the hypothetical space which should be searched by the belief was restrained and the meaning of a word is gained efficiently. The whole object principle, "a word points not a portion but things whole out" and the mutual exclusivity principle, "things are allowed only one category name" are known as such restrictions. Although a logical error is included, a man dared use such meta-knowledge and, as for each of these principles, has gained language efficiently. On the other hand, an animal's has gained stimulus equivalence is being unable to accept the relation which is logically contradictory. An animal worries in huge hypothetical space in the scene of meaning-of-a-word acquisition, and is considered to continue wandering about.

### 3 Connectionist Model of Stimulus Equivalence

#### 3.1 Symmetry as Ill-Posed Problem

Recently, attention is paid to the research of the connectionist approach by high compatibility with the structure of the brain and a nonlinear modeling ability. The connectionist approach has achieved the result as models of the acknowledgment functions of the language acquisition and the pattern recognition ability, etc.

However, stimulus equivalence can be caught with the ill-posed inverse problem (Fig.2). It is difficult for the the former type multilayer connectionist model to achieve the function to presume an unknown part from such truncated data. When the modeling of a nonlinear system tends to be performed paying attention to an input and an output and it is generally going to aim at a certain problem solution, two kinds of approaches can be considered in the modeling.

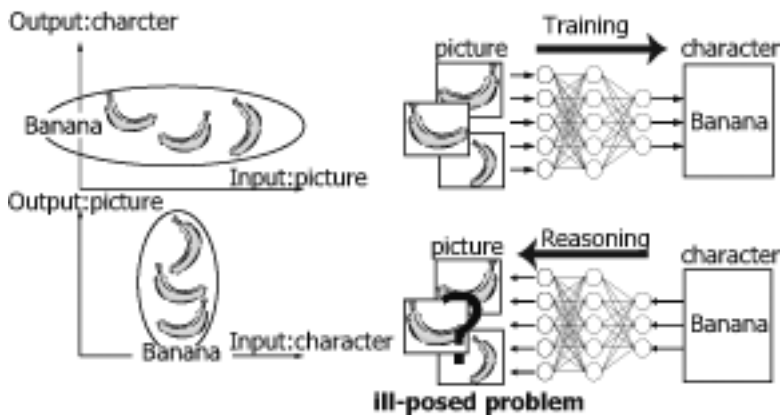


Fig. 2. Connectionist model approach

One is approach which pays its attention to the forward model for modeling, and another is approach which pays its attention to the inverse model for modeling. Especially, when the multilayer connectionist model is used as a modeling technique, this difference becomes an important point.

In general, the input/output relation of the modeling system becomes a map of multi vs. 1 as for the problem of acquiring forward model of the system by using the connectionist model. Therefore, forward model can be acquired comparatively easily by using the learning ability of neural network.

On the other hand, the connectionist model might be able to be used according to the application( no 1 vs. multi ) even by the approach that pays attention to a inverse model of the system. To be demanded from the model of the stimulus equivalence discussed by this paper is to solve a reverse problem of a nonlinear function. These problems become ill-posed problem to which the solution is not uniquely decided. Therefore, it is difficult for the multilayer connectionist model to acquire a inverse model of such a systema map which of multi vs. 1 .

In this paper, we propose a computational model of stimulus equivalence which solves the ill-posed reverse problem .

### 3.2 Connectionist Model of Symmetry Reasoning

A stimulus equivalence model which uses a layered neural network is shown in Fig.3. In Fig.3, the each level of the input layer, the hidden layer, and the output layer sequentially has two or more units from the left. The picture data is treated in the input layer and character string data is treated in the output layer.

Operation of a model consists of a learning phase ((1) of Fig.3), and a symmetry reasoning phase ((2) of Fig. 3). In the learning phase, forward model of the system is acquired by comparing the output from the network with standard signal by error back-propagation method. This process is equivalent to the portion shown by the solid line arrow in Fig.1. The picture of a banana is shown, and a character sequence called a "banana" is learned or it learns showing a character sequence called a "banana" and speaking with "banana".

In the symmetry reasoning phase, Associated reasoning is done by feeding back the error in the output layer on the network, which trained in the learning

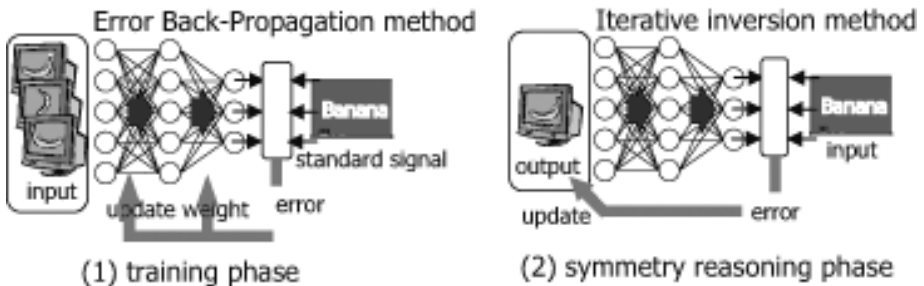


Fig. 3. Connectionist model of acquisition of symmetry reasoning

phase, directly to the input layer without additional training. Here, the symmetry reasoning phase has been the ill-posed inverse problem that the state of two or more input layers corresponds to one certain state in an output layer. Then, in the model to propose, the ill-posed inverse problem is solved by the Iterative Inversion method stated in the following section.

### 3.3 Iterative Inversion Method

We proposed the Iterative inversion method as a connectionist approach that solved a ill-posed inverse problem (Masumoto1992) (Nagata1988). Iterative inversion method acquires not a backward neural network model but a forward neural network model. A ill-posed inverse problem is undone by applying the iterative algorithm based on the gradient descent method such as the error back-propagation etc. to this multilayer neural network.

The Iterative Inversion method works in Figure 4 as follows.

- Step1: Forward model of the system that is from the input layer to the output layer is learned by using the error back-propagation method. It is learned to output character string "banana" showing the picture of the banana in Figure 1.
- Step2: An initial value of the input data  $s$  ( a picture of the banana in Figure 1.) based on a suitable hypothesis is first decided in backward reasoning,

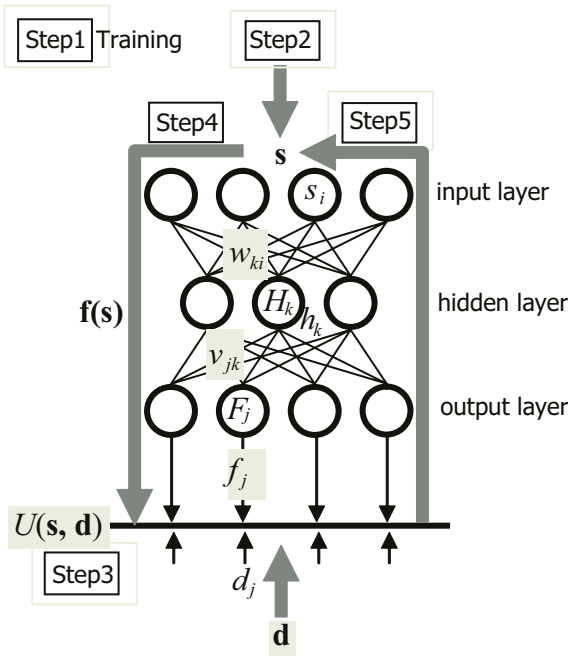


Fig. 4. Iterative inversion method

and it gives it to the input layer of the neural network. It corresponds in Figure 1 to the reasoning of the picture of the banana showing character string "banana".

Step3: Feed-forward calculation is executed and error signal  $U(\mathbf{s}, \mathbf{d})$  of the output value  $f(\mathbf{s})$  and the teaching signal  $\mathbf{d}$  (It is character string "banana." in Figure 1) from the output layer of the neural network is calculated.

$$U(\mathbf{s}, \mathbf{d}) = \sum_j (f_j(\mathbf{s}) - d_j)^2 \tag{1}$$

Step4: To reduce it when it is larger than the constant  $\epsilon$  with  $U(\mathbf{s}, \mathbf{d})$ , input data is corrected by the gradient descent method according to the next equation.

$$s_i(t + \Delta t) = s_i(t) - K \frac{\partial U(\mathbf{s}, \mathbf{d})}{\partial s_i} \tag{2}$$

where  $K$  is a constant.

The second term of the right side of an equation can be developed like the next expression.

$$\frac{\partial U(\mathbf{s}, \mathbf{d})}{\partial s_i} = 2 \sum (f_i(\mathbf{s}) - d) \frac{\partial f_j(\mathbf{s})}{\partial s_i} \tag{3}$$

Here, sensitivity the change of the value of one unit in the input layer of how output data  $f_i(\mathbf{s})$  changes, that is, the neural network exerts the influence of which extent on the output value of one unit in the output layer as for a right partial derivative when  $s_j$  is changed is shown. This partial derivative calculates as follows.

Input or output value to/from input layer's unit  $i$  is  $s_i$ .

Connection weights from input layer's unit  $i$  to hidden layer's unit  $k$  is  $w_{ki}$ .

Input value to hidden layer's unit  $k$  is  $H_k = \sum_i w_{ki} s_i$ .

Output value from hidden layer's unit  $k$  is  $h_k = \sigma(H_k)$  (*sigmoid transform function of unit*).

Connection weights from hidden layer's unit  $k$  to output layer's unit ( $j$ ) is  $v_{jk}$ .

Input value to output layer's unit  $j$  is  $F_j = \sum_k v_{jk} h_k$ .

Output value from output layer's unit  $j$  is  $f_j = \sigma(F_j)$ .

The next expression is obtained when as mentioned above defining it.

$$\begin{aligned} \frac{\partial f_j(\mathbf{s})}{\partial s_i} &= \frac{df_j}{dF_j} \frac{\partial F_j}{\partial s_i} \\ &= \sigma'(F_j) \sum_k v_{jk} \frac{\partial h_k}{\partial s_i} \\ &= \sigma'(F_j) \sum_k v_{jk} \sigma'(H_k) \frac{\partial H_k}{\partial s_i} \\ &= \sigma'(F_j) \sum_k v_{jk} \sigma'(H_k) w_{ki} \end{aligned} \tag{4}$$

Therefore,

$$\frac{\partial U(\mathbf{s}, \mathbf{d})}{\partial s_i} = 2 \sum (f_j - d_j) \sigma'(F_j) \sum_k v_{jk} \sigma'(H_k) w_{ki} \tag{5}$$

Step5: It returns to Step3 in updated  $S$  as an initial value.

The picture of the banana can be associated by showing character "banana" after character "banana" is studied showing the picture of the banana by repeating the processing of the above-mentioned.

### 4 Computer Simulation

The computer simulation was performed in order to check the validity of the proposed model. It is shown that it is realizable by the model which reasoning that the picture of a banana gives a character sequence "banana" proposes it carries out the picture of a banana is shown and a character sequence "banana" is learned.

The picture of various bananas (24 pictures that rotated the learning stimulation by 15 degrees were assumed to be study data in the simulation) was presented in the input layer, and it was studied that the character string "banana" was output to the output layer by the error back-propagation method (Figure3(1) Learning phase). When the error in the study phase decreased enough, the study phase was ended, and symmetry reasoning was executed. About 50,000 epoch's iteration was learned in the learnig phase in this experiment.

Whether the image of the banana was inferable was confirmed from the character string "banana" by the Iterative Inversion method by using an initial

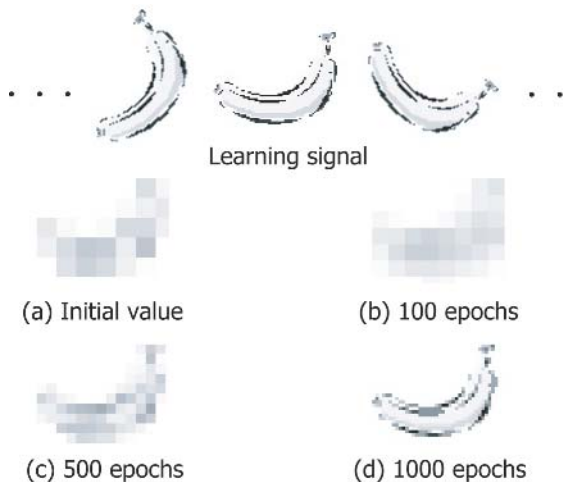


Fig. 5. Simulation result

value ( $s$  in Step2 of Figure4) as an initial value of the image given to the input layer (Figure3(2) Symmetry reasoning phase).

The result of a computer simulation is shown in Fig. 5. In the figure, a learning stimulus is a standard signals shown by the learnig phase in Figure3. When Fig. 5 (a) was given as an initial value of a network by the symmetry reasoning phase, the picture of Fig. 5 (d) was able to be reasoned by 1000 iteration trial.

## 5 Conclusion

In this paper, the stimulus equivalence known as a man's illogical cognitive bias thought that it was the factor of illogical reasoning of a man. While stimulus equivalence considered the role played in a man's reasoning process and language acquisition process, the computational model was proposed.

In order to examine the validity of the proposed model, when the computer simulation was performed, it was checked that a model can aquire symmetry.

## Acknowledgement

This study was partially supported by grants from the Integrative Human Science Program(21st Century COE), Tamgawa University.

## References

- Sidman, M., Tailby, W: Conditional discrimination vs. matching to sample: An expansion of the testing paradigm. *Journal of the Experimental Analysis of Behavior*. **37**(1982) 5–22
- Sidman, M.: *Equivalence relations and behavior: A research story*. Boston, M.A.: Authors Cooperative. (1994)
- Harlow, H.F.: The formation of learning sets. *Psychological Review*. **56**(1949) 51–65
- Vaughan, W.: Formation of Equivalence Sets in Pigeons. *Journal of Experimental Psychology, Animal Behavior Processes*. **14-1**(1988) 36–42
- Delius, J.D., Sieman, M.: Transitive responding in animals and humans: Explanation rather than adaptation?. *Behavioural Processes*. **42** (1988) 107–137
- Masumoto, D., Kimoto, T., Nagata, S.: A Neural Network Approach to Sensorimotor Fusion. *IMACS/SICE RMS92 Kobe*. (1992) 1105–1110
- Nagata, S.: Control of Mobile Robots with Neural Networks. *INNS*. (1988) 349

# Instability of Attractors in Auto-associative Networks with Bio-inspired Fast Synaptic Noise

Joaquín J. Torres<sup>1,2</sup>, Jesús M. Cortés<sup>1,2</sup>, and Joaquín Marro<sup>1,2</sup>

<sup>1</sup> Universidad de Granada, Departamento de Electromagnetismo y Física de la Materia, Facultad de Ciencias, E-18071 Granada, Spain

<sup>2</sup> Universidad de Granada, Instituto Carlos I de Física Teórica y Computacional, Facultad de Ciencias, E-18071 Granada, Spain

**Abstract.** We studied auto-associative networks in which synapses are *noisy* on a time scale much shorter than the one for the neuron dynamics. In our model a presynaptic noise causes postsynaptic depression as recently observed in neurobiological systems. This results in a nonequilibrium condition in which the network sensitivity to an external stimulus is enhanced. In particular, the fixed points are qualitatively modified, and the system may easily escape from the attractors. As a result, in addition to pattern recognition, the model is useful for class identification and categorization.

## 1 Introduction and Model

It is likely that the reported short-time *synaptic noise* determines the transmission of information in the brain [1, 2, 3, 4]. By means of a modified attractor neural network, we shall illustrate here that fast synaptic noise may result in a nonequilibrium condition [6] consistent with short-time depression [5]. We then show how this in turn induces escaping of the system from the attractor. The fact that the stability of fixed points is dramatically modified, in practice allows for complex computational tasks such as class identification and categorization, in close similarity to the situation reported in neurobiological systems [7, 8, 9]. A more detailed account of this work will be published elsewhere [10].

Consider a set of  $N$  binary neurons with configurations  $\mathbf{S} \equiv \{s_i = \pm 1; i = 1, \dots, N\}$  connected by synapses of intensity

$$w_{ij} = \bar{w}_{ij} x_j \quad \forall i, j. \quad (1)$$

Here,  $\bar{w}_{ij}$  is fixed and determined in a previous *learning* process, and  $x_j$  is a stochastic variable. For fixed  $\mathbf{W} \equiv \{\bar{w}_{ij}\}$ , the network state at time  $t$  is determined by  $\mathbf{A} = (\mathbf{S}, \mathbf{X} \equiv \{x_i\})$ . These evolve in time according to

$$\frac{\partial P_t(\mathbf{A})}{\partial t} = \sum_{\mathbf{A}'} [P_t(\mathbf{A}')c(\mathbf{A}' \rightarrow \mathbf{A}) - P_t(\mathbf{A})c(\mathbf{A} \rightarrow \mathbf{A}')] \quad (2)$$

where  $c(\mathbf{A} \rightarrow \mathbf{A}') = pc^{\mathbf{X}}(\mathbf{S} \rightarrow \mathbf{S}')\delta_{\mathbf{X},\mathbf{X}'} + (1-p)c^{\mathbf{S}}(\mathbf{X} \rightarrow \mathbf{X}')\delta_{\mathbf{S},\mathbf{S}'}$  [11]. This amounts to assume that neurons ( $\mathbf{S}$ ) change stochastically in time competing



with a noisy dynamics of synapses ( $\mathbf{X}$ ), the latter with an *a priori* relative weight of  $(1-p)/p$ .

For  $p = 1$ , the model reduces to the Hopfield case, in which synapses are quenched, i.e.,  $x_i$  is constant and independent of  $i$ . We are interested here in the limit  $p \rightarrow 0$  for which neurons evolve as in the presence of a steady distribution for the noise  $\mathbf{X}$ . If we write  $P(\mathbf{S}, \mathbf{X}) = P(\mathbf{X}|\mathbf{S})P(\mathbf{S})$ , where  $P(\mathbf{X}|\mathbf{S})$  stands for the conditional probability of  $\mathbf{X}$  given  $\mathbf{S}$ , one obtains from (2), after rescaling time  $tp \rightarrow t$  and summing over  $\mathbf{X}$  that

$$\frac{\partial P_t(\mathbf{S})}{\partial t} = \sum_{\mathbf{S}'} \{P_t(\mathbf{S}')\bar{c}[\mathbf{S}' \rightarrow \mathbf{S}] - P_t(\mathbf{S})\bar{c}[\mathbf{S} \rightarrow \mathbf{S}']\}. \quad (3)$$

Here,  $\bar{c}[\mathbf{S} \rightarrow \mathbf{S}'] \equiv \sum_{\mathbf{X}} P^{\text{st}}(\mathbf{X}|\mathbf{S}) c^{\mathbf{X}}[\mathbf{S} \rightarrow \mathbf{S}']$ , and the stationary solution is

$$P^{\text{st}}(\mathbf{X}|\mathbf{S}) = \frac{\sum_{\mathbf{X}'} c^{\mathbf{S}}[\mathbf{X}' \rightarrow \mathbf{X}] P^{\text{st}}(\mathbf{X}'|\mathbf{S})}{\sum_{\mathbf{X}} c^{\mathbf{S}}[\mathbf{X} \rightarrow \mathbf{X}']}. \quad (4)$$

This involves an adiabatic elimination of fast variables; see technical details in Ref.[6], for instance.

Notice that  $\bar{c}[\mathbf{S} \rightarrow \mathbf{S}']$  is a superposition. One may interpret that different underlying dynamics, each associated to a different realization of the stochasticity  $\mathbf{X}$ , compete. In the limit  $p \rightarrow 0$ , an *effective* rate results from combining  $c^{\mathbf{X}}[\mathbf{S} \rightarrow \mathbf{S}']$  with probability  $P^{\text{st}}(\mathbf{X}|\mathbf{S})$  for varying  $\mathbf{X}$ . Given that each elementary dynamics tends to drive the system to a different equilibrium state, the results is, in general, a nonequilibrium steady state [6]. The question is if such a competition between synaptic noise and neural activity is at the origin of some of the computational strategies in neurobiological systems.

For simplicity, we shall consider here *spin-flip* dynamics for the neurons, namely, stochastic local inversions  $s_i \rightarrow -s_i$  as induced by a bath at temperature  $T$ . The elementary rate then reduces  $c^{\mathbf{X}}[\mathbf{S} \rightarrow \mathbf{S}'] = \Psi[u^{\mathbf{X}}(\mathbf{S}, i)]$ , where we assume  $\Psi(u) = \exp(-u)\Psi(-u)$ ,  $\Psi(0) = 1$  and  $\Psi(\infty) = 0$  [6]. Here,  $u^{\mathbf{X}}(\mathbf{S}, i) \equiv 2T^{-1}s_i h_i^{\mathbf{X}}(\mathbf{S})$ , where  $h_i^{\mathbf{X}}(\mathbf{S}) = \sum_{j \neq i} \bar{w}_{ij} x_j s_j$  is the net presynaptic current or local field on the (postsynaptic) neuron  $i$ .

Our interest here is in modeling *noise* consistent with short-term synaptic depression [5, 12]. We therefore assume the noise distribution  $P^{\text{st}}(\mathbf{X}|\mathbf{S}) = \prod_j P(x_j|\mathbf{S})$  with

$$P(x_j|\mathbf{S}) = \zeta(\mathbf{m}) \delta(x_j + \Phi) + [1 - \zeta(\mathbf{m})] \delta(x_j - 1). \quad (5)$$

Here,  $\mathbf{m} = \mathbf{m}(\mathbf{S}) \equiv (m^1(\mathbf{S}), \dots, m^M(\mathbf{S}))$  is the  $M$ -dimensional overlap vector, and  $\zeta(\mathbf{m})$  stands for a function of  $\mathbf{m}$  to be determined. The depression effect here, namely,  $x_j = -\Phi$ , depends on the overlap vector which measures the net current arriving to postsynaptic neurons. Consequently, the non-local choice (5) introduces non-trivial correlations between synaptic noise and neural activity.

This new case also reduces to the Hopfield model but only in the limit  $\Phi \rightarrow -1$  for any  $\zeta(\mathbf{m})$ . In general, however, the competition results in a rather complex

nonequilibrium behavior. As far as  $\Psi(u+v) = \Psi(u)\Psi(v)$  and  $\mathbf{P}^{\text{st}}(\mathbf{X}|\mathbf{S})$  factorizes as indicated, time evolution proceeds by the effective transition rate

$$\bar{c}[\mathbf{S} \rightarrow \mathbf{S}^i] = \exp(-s_i h_i^{\text{eff}}/T), \quad (6)$$

where

$$h_i^{\text{eff}} = \sum_{j \neq i} w_{ij}^{\text{eff}} s_j. \quad (7)$$

Here,  $w_{ij}^{\text{eff}} = \{1 - \frac{1+\Phi}{2} [\zeta(\mathbf{m}) + \zeta(\mathbf{m}^i)]\} \bar{w}_{ij}$ , are the effective synaptic intensities as modified by the noise;  $\mathbf{m} = \mathbf{m}(\mathbf{S})$ ,  $\mathbf{m}^i \equiv \mathbf{m}(\mathbf{S}^i) = \mathbf{m} - 2s_i \boldsymbol{\xi}_i/N$ , and  $\boldsymbol{\xi}_i = (\xi_i^1, \xi_i^2, \dots, \xi_i^M)$ ,  $\xi_i^\nu = \{\xi_i^\nu = \pm 1, i = 1, \dots, N\}$ , stands for the binary  $M$ -dimensional stored pattern.

In order to obtain the effective fields (7), we linearized the rate  $\bar{c}[\mathbf{S} \rightarrow \mathbf{S}^i]$  around  $\bar{w}_{ij} = 0$ . This is a good approximation for the Hebbian prescription  $\bar{w}_{ij} = N^{-1} \sum_\nu \xi_i^\nu \xi_j^\nu$  as far as this only stores completely uncorrelated, random patterns and for a sufficiently large system, e.g., in the thermodynamic limit  $N \rightarrow \infty$ . To proceed further, we need to determine a convenient function  $\zeta$  in (5). In order to model activity-dependent mechanisms acting on the synapses,  $\zeta$  should be an increasing function of the field. In fact, this simply needs to depend on the overlaps. Furthermore,  $\zeta(\mathbf{m})$  is a probability, and it needs to preserve the  $\pm 1$  symmetry. A simple choice is

$$\zeta(\mathbf{m}) = \frac{1}{1+\alpha} \sum_\nu [m^\nu(\mathbf{S})]^2, \quad (8)$$

where  $\alpha = M/N$ . We describe next the behavior that ensues from (7)–(8) as implied by the noise distribution (5).

The effective rate (6) may be used in computer simulations, and it may also be substituted in the relevant equations. Consider, for instance, the *overlaps*, defined as the product of the current state with one of the stored patterns,  $m^\nu(\mathbf{S}) \equiv \frac{1}{N} \sum_i s_i \xi_i^\nu$ . After using standard techniques, it follows from (3) that

$$\partial_t m^\nu = 2N^{-1} \sum_i \xi_i^\nu \sinh(h_i^{\text{eff}}/T) - s_i \cosh(h_i^{\text{eff}}/T), \quad (9)$$

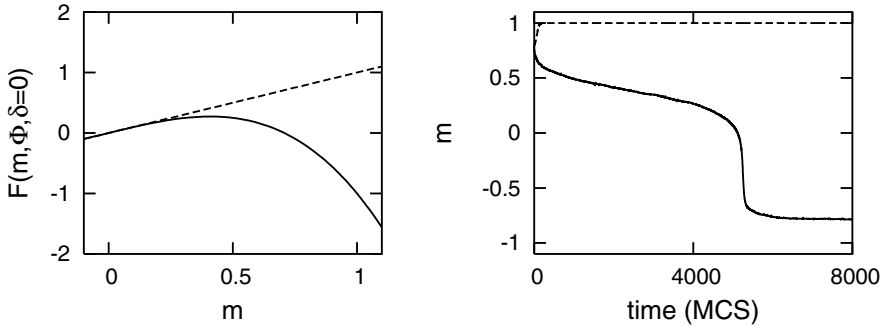
which is to be averaged over both thermal noise and pattern realizations.

## 2 Discussion of Some Main Results

We here illustrate the case of a single stored pattern,  $M = 1$ . After using the simplifying (mean-field) assumption  $\langle s_i \rangle \approx s_i$ , one obtains from (6)–(9) the steady overlap  $m^{\nu=1} \equiv m = \tanh\{T^{-1}m[1 - (m)^2(1 + \Phi)]\}$ . This depicts a transition from a *ferromagnetic-like* phase, i.e., solutions  $m \neq 0$ , to a *paramagnetic-like* phase,  $m = 0$ . The transition is continuous or second order only for  $\Phi > \Phi_c = -4/3$ , and it then follows a critical temperature  $T_c = 1$  [10].

It is to be remarked that a discontinuous phase transition allows for a much better performance of the retrieval process than a continuous one. This is because the behavior is sharp just below the transition temperature in the former case. Consequently, the above indicates that our model performs better for large negative  $\Phi$ ,  $\Phi < -4/3$ . These results are in full agreement with Monte Carlo simulations of neural networks with fast presynaptic noise and using asynchronous sequential updating.

We also investigated the sensitivity of the system under an external stimulus. A high sensitivity will allow for a rapid adaptation of the response to varying stimuli from the environment, which is an important feature of neurobiological systems. A simple external input may be simulated by adding to each local field a driving term  $-\delta\xi_i$ ,  $\forall i$ , with  $0 < \delta \ll 1$  [4]. A negative drive for a single pattern assures that the network activity may go from the attractor,  $\xi$ , to its “antipattern”,  $-\xi$ .



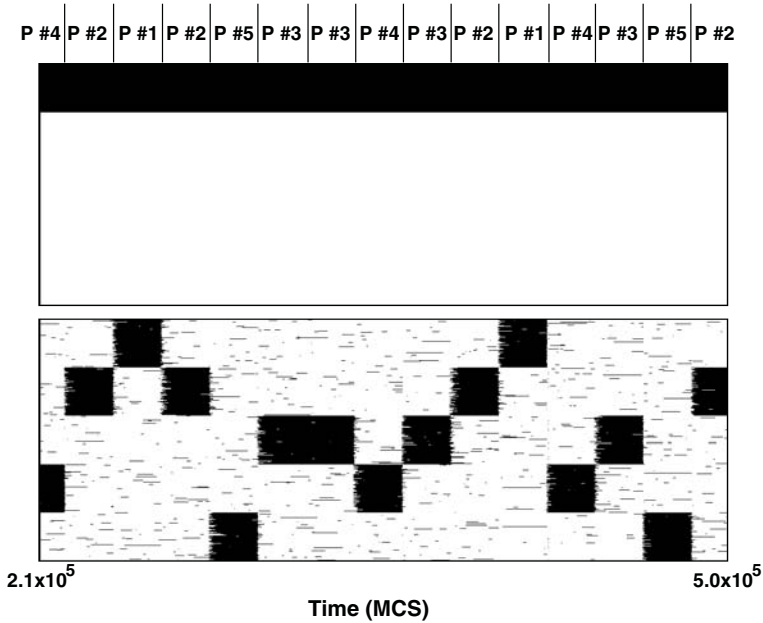
**Fig. 1.** (Left) The function  $F(m, \Phi, \delta = 0)$ , as defined in the main text, for presynaptic noise with  $\Phi = 1$  (solid line) and in absence of noise, i.e.,  $\Phi = -1$  (dashed line). Near the attractor ( $m \approx 1$ ),  $F$  is positive in the absence of noise, which leads stability of the memory. However, the noise makes  $F$  negative, which induces instability of the attractor. (Right) A retrieval experiment in a network of  $N = 3600$  neurons under external stimulation for  $T = 0.1$ ,  $\delta = 0.3$ , and the same values of  $\Phi$  as in the left graph

It follows for  $M = 1$  the stationary overlap  $m = \tanh[T^{-1}F(m, \Phi, \delta)]$  with  $F(m, \Phi, \delta) \equiv m[1 - (m)^2(1 + \Phi) - \delta]$ . The left graph of figure 1 shows this function for  $\delta = 0$  and  $\Phi = 1$  (fast noise) and  $\Phi = -1$  (Hopfield case). Depending on the sign of  $F$ , there are two different types of behavior, namely, (local)stability ( $F > 0$ ) and instability ( $F < 0$ ) of the attractor, which corresponds to  $m = 1$ . That is, the noise induces intrinsic instability resulting in switching between the pattern and the antipattern when a small perturbation  $\delta$  is added.

In general, adding the fast noise destabilizes the fixed point for the interesting case of small  $\delta$  far from criticality ( $T \ll T_c$ ). This is illustrated by Monte Carlo simulations of a network of  $N = 3600$  neurons,  $M = 1$ ,  $\delta = 0.3$  and  $T = 0.1$ , as shown in figure 1 right. Starting from an initial condition near the attractor,

the system jumps to the antipattern when fast noise is present (solid line), and remains in the attractor for the Hopfield case (dashed line).

The switching property remains as the system stores more patterns. In order to illustrate this, we simulated a network of  $N = 400$  neurons with  $M = 5$  overlapping patterns such that  $m^{\nu,\mu} \equiv 1/N \sum_i \xi_i^\nu \xi_i^\mu = 1/5$  for any two of them. The system in this case begins with the first pattern, and then evolves under the effect of a repetitive small stimulus  $+\delta\xi^\sigma$  ( $\delta = 0.3$ ) with  $\sigma$  randomly chosen from 1 to 5 every  $2 \times 10^4$  MCS for each  $\sigma$ . As shown in figure 2, lacking the noise ( $\Phi = -1$ ), the system remains in the initial pattern. However, in the presence of some noise ( $\Phi = 0.05$  in this simulation), there is continuous jumping from one attractor to the other, every time the new attractor is presented in the stimulus. This property is robust with respect to the type of patterns stored [10].



**Fig. 2.** Sensitivity of the system under repetitive external random stimulus, as discussed in the main text. The top graph shows the Hopfield case ( $\Phi = -1$ ). Here, neuron activity is represented at vertical axis and simulation parameters are  $N = 400$ ,  $T = 0.1$  and  $\delta = 0.3$

Summing up, equations (3)–(4) provide a rather general framework to model activity–dependent processes. We here briefly reported on some consequences of adapting this to a specific case. In particular, we studied a case which describes neurobiologically–motivated fast noise, and study how this affects the synapses of an auto–associative neural network with a finite number of stored patterns. Assuming a noise distribution with a global dependence on the activity, (5),

one obtains non-trivial local fields (7) which lead the system to an intriguing emergent phenomenology. We studied this case both analytically and by Monte Carlo simulations using Glauber, *spin-flip* dynamics [6]. We thus show that a tricritical point occurs. That is, one has (in the limit  $\alpha \rightarrow 0$ ) first and second order phase transitions between a ferromagnetic-like, retrieval phase and a paramagnetic-like, non-retrieval phase. The noise also happens to induce a nonequilibrium condition which results in an important intensification of the network sensitivity to external stimulation. We explicitly show that the noise may turn unstable the *attractor* or fixed point solution of the retrieval process, and the system then seeks for another attractor. This behavior improves the network ability to detect changing stimuli from the environment. One may argue that the process of categorization in nature might follow a similar strategy. That is, different attractors may correspond to different objects, and a dynamics conveniently perturbed by fast noise may keep visiting the attractors belonging to a class which is characterized by a certain degree of correlation between its elements. A similar mechanism seems at the basis of early olfactory processing of insects [9], and instabilities of the same sort have been described in the cortical activity of monkeys [7] and other cases [8]. We are presently studying further variations of the model above.

## Acknowledgments

We acknowledge financial support from MCyT and FEDER (project No. BFM2001-2841 and *Ramón y Cajal* contract).

## References

1. Abbott L.F. and Regehr W.G.: Synaptic computation *Nature* **431** (2004) 796–803
2. Allen C. and Stevens C.F.: An evaluation of causes for unreliability of synaptic transmission *Proc. Nat. Acad. Sci.* **91** (1994) 10380–10383
3. Zador A.: Impact of synaptic unreliability on the information transmitted by spiking neurons *J. Neurophysiol.* **79** (1998) 1219–1229
4. Bibitchkov D., Herrmann J.M., and Geisel T.: Pattern storage and processing in attractor networks with short-time synaptic dynamics *Network: Comput. Neural Syst.* **13** (2002) 115–131
5. Tsodyks M., Pawelzik K., and Markram H.: Neural networks with dynamic synapses *Neural Comput.* **10** (1998) 821–835
6. Marro J. and Dickman R., *Nonequilibrium Phase Transitions in Lattice Models*, Cambridge Univ. Press, Cambridge 1999.
7. Abeles M., Bergman H., Gat I., Meilijson I., Seideman E., Tishby N., and Vaadia E.: Cortical activity flips among quasi-stationary states *Proc. Natl. Acad. Sci. USA* **92** (1995) 8616–8620
8. Miller L.M. and Schreiner C.E.: Stimulus-based state control in the thalamocortical system *J. Neurosci.* **20** (2000) 7011–7016
9. Laurent G., Stopfer M., Friedrich R., Rabinovich M., Volkovskii A. and Abarbanel H.: Odor encoding as an active, dynamical process: experiments, computation and theory *Annu. Rev. Neurosci.* **24** (2001) 263–297

10. Cortes J.M., Torres J.J., Marro J., Garrido P.L., and Kappen H.J.: Effects of Fast Presynaptic Noise in Attractor Neural Networks *Neural Comput.* (2004) submitted.
11. Torres J.J., Garrido P.L., and Marro J.: Neural networks with fast time-variation of synapses *J. Phys. A: Math. and Gen.* **30** (1997) 7801–7816
12. Pantic L., Torres J.J., Kappen H.J., and Gielen S.C.A.M.: Associative memory with dynamic synapses *Neural Comput.* **14** (2002) 2903-2923

# Lookup Table Powered Neural Event-Driven Simulator

Richard R. Carrillo<sup>1</sup>, Eduardo Ros<sup>1</sup>, Eva M. Ortigosa<sup>1</sup>,  
Boris Barbour<sup>2</sup>, and Rodrigo Agís<sup>1</sup>

<sup>1</sup> Department of Computer Architecture and Technology,  
E.T.S.I. Informática, University of Granada,  
18071 Granada, Spain

{rcarrillo, eduardo, eva, ragis}@atc.ugr.es

<sup>2</sup> Laboratoire de Neurobiologie (CNRS UMR 8544)  
Ecole Normale Supérieure, 75230 Paris Cedex 05, France  
barbour@ens.fr

**Abstract.** A novel method for efficiently simulating large scale realistic neural networks is described. Most information transmission in these networks is accomplished by the so called action potentials, events which are considerably sparse and well-localized in time. This facilitates a dramatic reduction of the computational load through the application of the event-driven simulation schemes. However, some complex neuronal models require the simulator to calculate large expressions, in order to update the neuronal state variables between these events. This requirement slows down these neural state updates, impeding the simulation of very active large neural populations in real-time. Moreover, neurons of some of these complex models produce firings (action potentials) some time after the arrival of the presynaptic potentials. The calculation of this delay involves the computation of expressions that sometimes are difficult to solve analytically. To deal with these problems, our method makes use of precalculated lookup tables for both, fast update of the neural variables and the prediction of the firing delays, allowing efficient simulation of large populations with detailed neural models.

## 1 Introduction

Recent research projects are modelling neural networks based on specific brain areas. Realistic neural simulators are required in order to evaluate the proposed network models. Some of these models (e.g. related with robot control or image processing [1, 2]) are intended to interface with the real world, requiring real-time neural simulations. This kind of experiments demands efficient software able to simulate large neural populations with moderated computational power consumption.

Traditionally, neural simulations have been based on discrete time step (synchronous) methods [3, 4]. In these simulations, the state variables of each neuron are updated every time step, according to the current inputs and the previous values of these variables. The differential expressions describing the neural model dynamics are usually

computed with numerical integration methods such as Euler or Runge-Kutta. The precision of the numerical integration of these variables depends on the time step discretization. Short time steps are required in order to achieve acceptable precision, which means considerable computational power consumption by each neuron. Thus, simulating large neural population with adequate precision and detailed models using these methods is not feasible in real-time.

One alternative to avoid this problem is the use of event-driven simulators (also known as discrete-event simulators). Most natural network communication is done by means of spikes (action potentials) which are short and considerably sparse in time (not very frequent) events. If the state evolution of a neuron between these spikes is deterministic or the probability of all the target states is known, the number of neural state updates could be reduced, accumulating the entire computational load in the instants in which the spikes are produced or received by a neuron [5, 6] (see Fig. 1).

Mattia & Giudice [7] proposed an event-driven scheme that included dynamical synapses. Reutimann et al extended this approach to include neuron models with stochastic dynamics.

Makino [8] developed an event-driven simulator which uses efficient numerical methods to calculate the neural states evolution from one discrete computed step to the next one. More concretely, the main contribution of this work is the development of an efficient strategy to calculate the delayed firing times that uses the linear envelopes of the state variable of the neuron to partition the simulated time. Contrary to this approach, we avoid this complex calculation by off-line characterization of the firing behaviour of the cell.

Recently, Reutimann et al [9] and Brette [10] proposed the use of pre-calculated lookup tables to speed up simulations to avoid on-line numerical calculations. We adopt this strategy in our event-driven simulator. In these approaches the pre-calculated tables are used to store probability density distributions [9] and partial mathematical functions on which the neuron characteristic dynamics are based [10]. Furthermore, the method proposed by Brette imposes the restriction of equal time constants in all the synaptic types while realistic inhibitory synapses exhibit longer time constants than the excitatory ones. In our approach, the entire cell model is computed off-line, and its behaviour is compiled into characterization tables. Since the cell model is computed off-line, we are able to simulate models of different complexities (with a constraint on the number of parameters defining cell dynamics).

The main innovation with respect to previous similar approaches [5, 7], is the use of characterization tables to describe the cell dynamics between input spikes. A priori, this fact removes the need for many of the simplifying assumptions necessary when the neural models are computed following simple expressions to achieve high computational efficiency.

Another important aspect, that has been included, is the synaptic temporal dynamics (i.e. the gradual injection/extraction of charge). The synaptic conductance evolution due to an input spike is not computed as an instantaneous jump, but as a gradual function. This is important in the study of neural population synchronization processes [11, 12]. The inclusion of temporal dynamics forces the implementation of a prediction and validation strategy, since the output spikes will not be coincident with the input events (variable firing delay). This introduces some more complexity in the simulation algorithm.



## 2 Computation Scheme

The general idea of the event-driven computation scheme is that the neuron states need only to be updated when a new event is received. Thus, the simulation time ( $t$ ) can run discontinuously. In our approach, the new neuron state can be obtained using tables instead of specific analytical equations.

**Event Reordering.** In an event-driven computation scheme, spikes need to be processed in chronological order. Therefore, a spike re-ordering mechanism is required in order to deal with the delayed firing and the propagation delays of the spikes. This data structure is used as an interface between the source neuron events and target neurons.

Complex data structures, such as “balanced trees”, can be used for this purpose, offering  $O(\log(N))$  for insertion and deletion of elements. However, to prevent performance degradation, they require the optimization of their structure after some insertions or deletions. This rebalancing process adds more complexity and additional computational overhead [13].

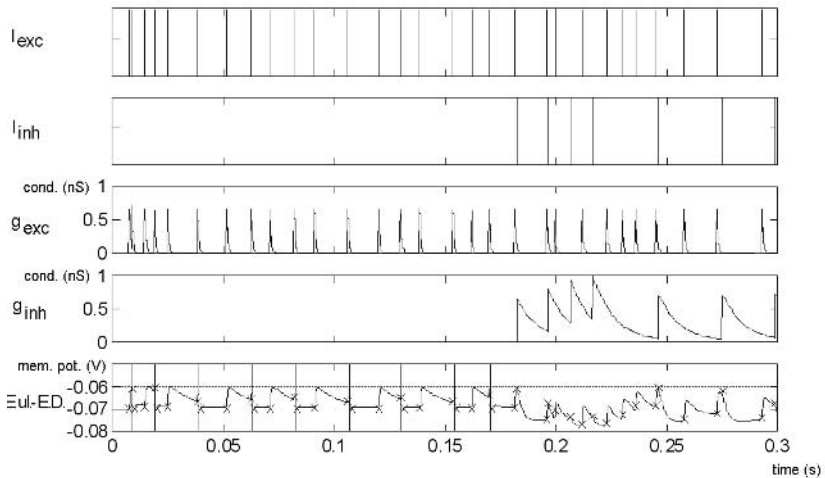
An alternative candidate data structure is the “skip list” [14], but in this case the cost of the worst case may not be  $O(\log(N))$  because the insertion of an input stream can produce an unbalanced structure. This structure offers good performance in searching specific elements. However, this is not needed in our computation scheme as we only need to extract the first element, i.e., the next spike.

Finally, the “heap data structure” (priority queue) [15, 16, 17] offers a stable computational cost of  $O(\log(N))$  in inserting and deleting elements. This a convenient option as it does not require more memory resources than the stored data. This is because it can be implemented as an array, while the “balanced trees” and “skip lists” need further pointers or additional memory resources.

**Delayed Firing.** When a spike reaches a neuron, it affects the membrane potential by gradually injecting or extracting charge. The membrane potential evolution after a spike follows a function that depends on the synaptic dynamics (as illustrated in Fig. 1, for the Spike Response Model [18]).

If we restrict our approach to a one-compartment model and the synaptic dynamics are approximated as single exponentials (see Fig. 1), all the excitatory inputs can be summed in one term and all the inhibitory inputs in another. The whole contribution associated with an input event is not injected (extracted) instantaneously. If a second spike reaches the neuron while the membrane potential is evolving due to a previous spike, the additional contribution can be computed by updating the charge injection (extraction) function.

Since the effect of an input spike extends over a time period, the target neuron membrane potential evolves gradually and it may reach the firing threshold some time after the input is received (this can be critical for certain winner-take-all processing mechanisms [19]). When an input spike reaches a target neuron, the firing time is calculated (assuming that no further inputs are received by this cell). This new event is inserted into the spike heap as a *firing event*. All these *firing events* are predictions; they may or may not happen depending on the other cells activity after the prediction. If a new spike is received before the neuron fires, a new prediction is made and inserted into the spike heap, invalidating the last one, as the firing time label of the corresponding neuron will not match the time label of the first predicted *firing event* when it is received.



**Fig. 1.** Single neuron simulation. Excitatory and inhibitory spikes on the upper plots. Excitatory and inhibitory conductances transients on the middle plots. The bottom plot is a comparison between the neural model simulated with iterative numerical calculation (continuous trace) and the event-driven scheme, in which the membrane potential is only updated when an input spike is received or generated (marked with an x)

**Large Fan Out Optimization.** The algorithm efficiency depends on the size of the event heap. Therefore, the computing scheme is optimized for a reduced number of events and heap load. However, in some cerebellar circuits [20] and other learning areas, huge fan outs can be found in biological neural systems with different delays for different connections. There are two options to deal with this problem. In a first approach, if one neuron fires and the spike reaches target neurons with different delays, one event is inserted into the spike heap for each target neuron. However, this would overload the heap particularly when a set of neurons with large fan outs are very active. In a second approach, each time a neuron fires, a process of several stages is launched. The output connection list of each neuron (which indicates the target cells of each neuron) is sorted by the connection delay. When a source neuron fires, only the event corresponding to the shortest delay connection, is inserted into the spike heap.

This event is linked with the other output spikes of this source neuron. When this first spike is processed and removed from the heap, the next event in the output connection list is inserted into the spike heap together with the updated delay. This is repeated until the last event in the list is processed. In this way, the system can handle large connection fan outs without overloading the spike heap structure.

**Simulation Engine.** The simulation scheme supports the definition of arbitrary connection delays. This implies that each spike transmitted between two cells is represented internally by two events. The first one (called the *firing event*) is marked with the time instant when the source neuron fires the spike. The second one (called the *propagated event*) is marked with the time instant when the spike reaches the target neuron.

The basic computation scheme consists of a processing loop, in which, in each iteration, the event with shortest *time-to-go* is extracted from this spike heap structure, the target neuron variables are updated, and, if the affected neurons generate them, new events are inserted into the spike heap.

Simulation algorithm. This pseudo-code describes the simulation engine. It processes all the events of the spike heap in chronological order.

```

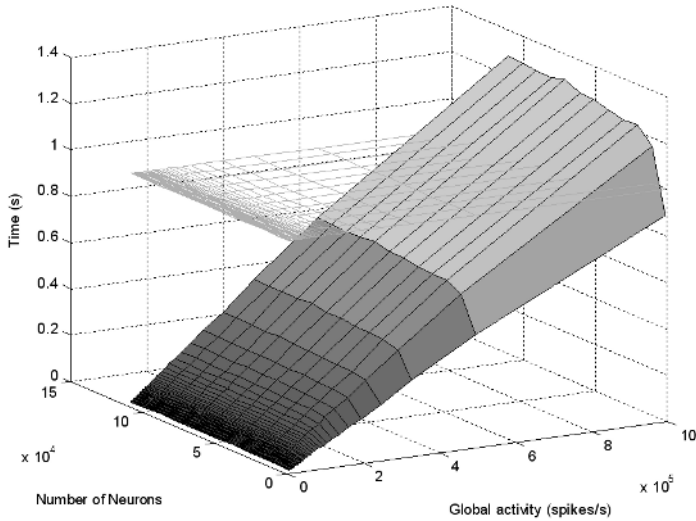
While $t_{sim} < t_{end}$
 Extract the event with a shortest time-to-go in the
 spike heap
 If it is a firing event
 If it is still a valid event and the neuron is not
 under a refractory period
 Update the neuron state: V_m, g_{exc}, g_{inh}
 Prevent this neuron from firing during the refrac-
 tory period, updating the neuron time label to
 $t_{sim} + t_{refrac}$
 Predict if the source neuron will fire again with
 the current neuron state
 If the neuron will fire:
 Insert a new firing event into the spike heap
 Insert the propagated event with the shortest time-
 to-go (looking at the output connection list)
 If it is a propagated event
 Update the target neuron state: V_m, g_{exc}, g_{inh} looking at
 the characterization tables
 Modify the conductances (g_{exc}, g_{inh}) using the connec-
 tion weight ($G_{exc,i}, G_{inh,i}$) for the new spike
 Update the neuron time label to t_{sim}
 Predict if the target neuron will fire
 If it fires
 Insert the firing event into the spike heap with
 the predicted time
 Insert only the next propagated event with the next
 shortest time-to-go (looking at the output connection
 delay table)

```

One important variable, which is updated when a neuron receives a spike, is the time label. This variable indicates the last time this neuron was updated. As described in the simulation algorithm, when a neuron is affected by an event, the time label of this neuron is updated to  $t_{sim}$  if it is an input spike (*propagated event*) or to  $t_{sim} + t_{refrac}$  if it is an output spike (*firing event*), to prevent it from firing again during the refractory period.

### 3 Computer Performance

With the described approach, the simulation time does not depend on the network size but on the rate of events per second that need to be processed. More precisely, the simulation time depends on the network activity as illustrated in Fig. 2. A more extended study can be found in [21].



**Fig. 2.** This figure represents the time taken in a 1 second simulation on a Pentium IV (1.8 GHz) platform. Global activity represents the total number of spikes per second in the network. The network size has not a significant impact on the time required. On the other hand, the computation time depends almost linearly on the network activity. The horizontal grid represents the real-time simulation limit, i.e. one second of simulation requiring one second of computation

## 4 Discussion

To the best of our knowledge, this approach represents the first event-driven simulation scheme entirely based on pre-calculated neural characterization tables. This considerably reduces the a priori assumptions for the cell models. In principle, cell models of considerable complexity can be simulated. The limitation is given by the table dimensions and sizes.

The aim of the described approach is to enable efficient simulation of neural structures of reasonable size, based on cells whose characteristics are not described by simple analytical expressions. This is achieved by defining the neural dynamics using the pre-calculated traces of their internal variables. The proposed scheme efficiently splits the computational load in two different stages:

- Off-line neural model characterization: This preliminary stage requires a massive numerical calculation of the cell model in different conditions, to scan its dynamics. The goal of this stage is to build up the neural characterization tables. In principle, this could even be done compiling electro-physiological recordings of real cells.
- On-line event-driven simulation scheme: The computation of the simulation process jumps from one event to the next, updating the neuron states according to pre-calculated neuron characterization tables and generating new events.

The proposed scheme represents a simulation tool that lies between the very detailed simulators (such as NEURON [22] or GENESIS [3]) and the event-driven simulation schemes based on simple analytically described cell dynamics [23, 4].

The performance (computation speed) of the proposed event-driven simulation scheme is determined by the total event rate and not by the network size. The size of the characterization tables does not affect the computation performance. Therefore, the use of very large characterization tables becomes of interest in order to study the significative properties of the cells in the framework of large-scale simulations. This prompts the parallelization of the off-line computation stage that builds up the characterization tables, which is one of the future lines of work.

The described approach has been applied to simulations based on one-compartmental cell models approximating excitatory AMPA receptor-mediated synaptic dynamics and the GABAergic inhibitory synaptic dynamics (with longer time constants), with single exponential functions. The inclusion of new synaptic mechanisms, such as voltage dependent channels is possible, it only requires the inclusion of new neural variables, and thus, new table dimensions. Different strategies, such as multiple-stage table management, are also being studied to address cell models of larger complexity more efficiently.

## Acknowledgements

This work has been supported by the EU project SpikeFORCE (IST-2001-35271) and the National Spanish Grant (DPI 2004-07032).

## References

1. Van Rullen, R., Gautrais, J., Delorme, A., Thorpe, S.: Face processing using one spike per neurone. *BioSystems*, 48. (1998) 229-239
2. Philipona, D., Coenen, O. J. M. D.: Model of granular layer encoding in the cerebellum. *Neurocomputing*, vol. 58-60. (2004) 575-580
3. Bower, J.M., Beeman, B.: *The book of GENESIS*. New York. Springer Verlag (1998)
4. Delorme, A., Thorpe, S.: SpikeNET: An event-driven simulation package for modelling large networks of spiking neurons. *Network. Computation in Neural Systems*, Vol. 14. (2003) 613-627
5. Watts, L.: Event-driven simulation of networks of spiking neurons. In J.D. Cowan, G. Teesauro, & J. Alspector (Eds.), *Advances in neural information processing systems*, 6. San Mateo CA Morgan Kaufmann (1994) 967-934
6. Graßmann, C., Anlauf, J. K.: Fast digital simulation of spiking neural networks and neuromorphic integration with SPIKELAB. *International Journal of Neural Systems*, 9(5). (1999) 473-478
7. Mattia, M., Del Giudice, P.: Efficient event-driven simulation of large networks of spiking neurons and dynamical synapses. *Neural Computation*, 12(10). (2000) 2305-2329
8. Makino, T.: A Discrete-Event Neural Network Simulator for General Neuron Models. *Neural Comput & Applic*, 11. (2003) 210-223
9. Reutimann, J., Guigliano, M., Fusi, S.: Event-driven simulation of spiking neurons with stochastic dynamics. *Neural Computation*, 15. (2003) 811-830

10. Brette, R.: Event-driven simulation of integrate-and-fire neurons with exponential synaptic conductances. Submitted to *Journal of Computational Neuroscience* (2004)
11. Eckhorn, R., Bauer, R., Jordan, W., Brosh, M., Kruse, W., Munk, M., Reitböck: Coherent oscillations: A mechanism of feature linking in the visual cortex? *Biol. Cyber.*, 60. (1988) 121-130
12. Eckhorn, R., Reitböck, H.J., Arndt, M., Dicke, D.: Feature Linking via Synchronization among Distributed Assemblies: Simulations of Results from Cat Visual Cortex. *Neural Computation*, 2. (1990) 293-307
13. Karlton, P.L., Fuller, S.H., Scroggs, R.E., Kaehler, E.B.: Performance of height-balanced trees. In *Information retrieval and language processing. Communications of ACM.* 19(1). (1976) 23-28
14. Pugh, W.: Skip lists: A probabilistic alternative to balanced trees. *Communications of the ACM*, 33(6). (1990) 668-676
15. Aho, A.V., Hopcroft, J.E., Ullman, J. D.: *The design and analysis of computer algorithms.* Reading, MA, Addison-Wesley (1974)
16. Chowdhury R.A., Kaykobad M.: Sorting using heap structure. *Proceedings of Int. Conf. on Comp. and Inf. Tech. Dhaka, Bangladesh* (2001) 26-30
17. Cormen, T. H., Lierson, C. E., Rivest, R. L.: *Introduction to algorithms.* MIT Cambridge press (1990) 140-152
18. Gerstner, W., Kistler, W.: *Spiking neuron models: Single neurons, populations, plasticity.* Cambridge: Cambridge University (2002)
19. Ros, E., Pelayo, F.J., Martín-Smith, P., Palomar, D., Prieto, A. Competitive and Temporal Inhibition Structures with Spiking Neurons. *Neural Processing Letters.* Vol. 11(3). (2000) 197-208
20. Chez, C.: The Cerebellum. In *Principles of Neural Science (Third Edition)*, Edited by Kandel, E., Schwartz, J.H. and Jessel (1991) 626-645
21. Ros, E., Carrillo, R., Ortigosa, E. M., Barbour, B., Agís, R.: Event-driven Simulation Scheme for Spiking Neural Models based on Characterization Look-up Tables. Submitted to *Neural Computation* in 2004
22. Hines, M.L., Carnevale, N.T.: The NEURON simulation environment. *Neural Computation*, 9. (1997) 1179-1209
23. Delorme, A., Gautrais, J. van Rullen, R., Thorpe, S.: SpikeNET: A simulator for modeling large networks of integrate and fire neurons. In J. M. Bower (Ed.), *Computational Neuroscience: Trends in research 1999, Neurocomputing, Vols. 26-27.* (1999) 989-996

# Joint Kernel Maps

Jason Weston<sup>1</sup>, Bernhard Schölkopf<sup>2</sup>, and Olivier Bousquet<sup>2</sup>

<sup>1</sup> NEC Laboratories, America, Princeton, NJ, USA

<sup>2</sup> Max Planck Institute for Biological Cybernetics, 72076 Tübingen, Germany

**Abstract.** We develop a methodology for solving high dimensional dependency estimation problems between pairs of data types, which is viable in the case where the output of interest has very high dimension, e.g., thousands of dimensions. This is achieved by mapping the objects into continuous or discrete spaces, using joint kernels. Known correlations between input and output can be defined by such kernels, some of which can maintain linearity in the outputs to provide simple (closed form) pre-images. We provide examples of such kernels and empirical results.

## 1 Introduction

We begin by providing some background in kernel methods.

Suppose we are given empirical data

$$(x_1, y_1), \dots, (x_m, y_m) \in \mathcal{X} \times \mathcal{Y}. \quad (1)$$

Here, the domain  $\mathcal{X}$  is some nonempty set that the inputs  $x_i$  are taken from; the  $y_i \in \mathcal{Y}$  are called *targets*. Here and below,  $i, j = 1, \dots, m$ .

Note that we have not made any assumptions on the domain  $\mathcal{X}$  other than it being a set. In order to study the problem of learning, we need additional structure. In learning, we want to be able to *generalize* to unseen data points. In the case of pattern recognition, given some new input  $x \in \mathcal{X}$ , we want to predict the corresponding  $y \in \{\pm 1\}$ . Loosely speaking, we want to choose  $y$  such that  $(x, y)$  is in some sense *similar* to the training examples. To this end, we need similarity measures in  $\mathcal{X}$  and in  $\{\pm 1\}$ . The latter is easier, as two target values can only be identical or different. For the former, we require a similarity measure

$$k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}, \quad (x, x') \mapsto k(x, x') \quad (2)$$

with the property that there exists a map  $\Phi$  into a Hilbert space  $\mathcal{H}$  such that for all  $x, x' \in \mathcal{X}$ ,

$$k(x, x') = \langle \Phi(x), \Phi(x') \rangle. \quad (3)$$

Such a function  $k$  is called a *positive definite (pd) kernel* [1–3],  $\mathcal{H}$  is the *reproducing kernel Hilbert space (RKHS)* associated with it, and  $\Phi$  is called its *feature map*. A popular example, in the case where  $\mathcal{X}$  is a normed space, is the Gaussian

$$k(x, x') = \exp\left(-\frac{\|x - x'\|^2}{2\sigma^2}\right), \quad (4)$$

where  $\sigma > 0$ .

The advantage of using a pd kernel as a similarity measure is that it allows us to construct algorithms in Hilbert spaces. For instance, consider the following simple classification algorithm, where  $\mathcal{Y} = \{\pm 1\}$ . The idea is to compute the means of the two classes in the RKHS,  $\mathbf{c}_1 = \frac{1}{m_1} \sum_{\{i:y_i=+1\}} \Phi(x_i)$ , and  $\mathbf{c}_2 = \frac{1}{m_2} \sum_{\{i:y_i=-1\}} \Phi(x_i)$ , where  $m_1$  and  $m_2$  are the number of examples with positive and negative target values, respectively. We then assign a new point  $\Phi(x)$  to the class whose mean is closer to it. This leads to

$$y = \text{sgn} (\langle \Phi(x), \mathbf{c}_1 \rangle - \langle \Phi(x), \mathbf{c}_2 \rangle + b) \tag{5}$$

with  $b = \frac{1}{2} (\|\mathbf{c}_2\|^2 - \|\mathbf{c}_1\|^2)$ . Rewritten in terms of  $k$ , this reads

$$y = \text{sgn} \left( \frac{1}{m_1} \sum_{\{i:y_i=+1\}} k(x, x_i) - \frac{1}{m_2} \sum_{\{i:y_i=-1\}} k(x, x_i) + b \right) \tag{6}$$

and  $b = \frac{1}{2} \left( \frac{1}{m_2^2} \sum_{\{(i,j):y_i=y_j=-1\}} k(x_i, x_j) - \frac{1}{m_1^2} \sum_{\{(i,j):y_i=y_j=+1\}} k(x_i, x_j) \right)$ .

Let us consider one well-known special case of this type of classifier. Assume that the class means have the same distance to the origin (hence  $b = 0$ ), and that  $k$  can be viewed as a density, i.e., it is positive and has integral 1,  $\int_{\mathcal{X}} k(x, x') dx = 1$  for all  $x' \in \mathcal{X}$ . Then (6) corresponds to the Bayes decision boundary separating the two classes, subject to the assumption that the two classes are equally likely and were generated from two probability distributions that are correctly estimated by the Parzen windows estimators of the two classes,

$$p_1(x) := \frac{1}{m_1} \sum_{\{i:y_i=+1\}} k(x, x_i), \quad p_2(x) := \frac{1}{m_2} \sum_{\{i:y_i=-1\}} k(x, x_i). \tag{7}$$

The classifier (6) is quite close to the *Support Vector Machine (SVM)* that has recently attracted much attention [3-5]. It is linear in the RKHS (see (5)), while in the input domain, it is represented by a kernel expansion (6). It is example-based in the sense that the kernels are centered on the training examples, i.e., one of the two arguments of the kernels is always a training example. This is a general property of kernel methods, due to the Representer Theorem [5, 6]. The main point where SVMs deviate from (6) is in the selection of the examples that the kernels are centered on, and in the weight that is put on the individual kernels in the decision function. The SVM decision boundary takes the form

$$y = \text{sgn} \left( \sum_{i=1}^m \lambda_i k(x, x_i) + b \right), \tag{8}$$

where the coefficients  $\lambda_i$  and  $b$  are computed by solving a convex quadratic programming problem such that the margin of separation of the classes in the RKHS is maximized. It turns out that for many problems this leads to sparse solutions, i.e., often many of the  $\lambda_i$  take the value 0. The  $x_i$  with nonzero  $\lambda_i$  are called *Support Vectors*.

Using methods from statistical learning theory [4], one can bound the generalization error of SVMs. In a nutshell, statistical learning theory shows that it is imperative that



one uses a class of functions whose *capacity* (e.g., measure by the VC dimension) is matched to the size of the training set. In SVMs, the capacity measure used is the size of the margin.

The SV algorithm has been generalized to problems such as regression estimation [4], one-class problems and novelty detection [5], as well as to mappings between general sets of objects [7]. The latter uses two kernels,  $k_{\mathcal{X}}$  and  $k_{\mathcal{Y}}$ , on  $\mathcal{X}$  and  $\mathcal{Y}$ , respectively, and learns a linear map between the associated RKHS's  $\mathcal{H}_{\mathcal{X}}$  and  $\mathcal{H}_{\mathcal{Y}}$ . The feature map of  $k_{\mathcal{X}}$  analyzes the input by computing its feature representation in  $\mathcal{H}_{\mathcal{X}}$ , while the feature map of  $k_{\mathcal{Y}}$  synthesizes the output  $\Psi$  of the linear map in  $\mathcal{H}_{\mathcal{Y}}$ . It can be thought of as inducing a generative model for the outputs; in the algorithm, it is usually employed to compute a pre-image of  $\Psi$  in  $\mathcal{Y}$ .

In the present paper, we will generalize this situation to the case where there is a kernel that jointly compares inputs and outputs.

To this end, first consider the problem of linear regression. Given a training set of paired objects  $\{(\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_m, \mathbf{y}_m)\}$  identically and independently sampled from a distribution  $P$  over the product space  $\mathcal{X} \times \mathcal{Y}$ , we wish to find a function  $W$  that maps from  $\mathcal{X}$  into  $\mathcal{Y}$  such that:

$$\int_{\mathcal{X} \times \mathcal{Y}} \|\mathbf{y} - W\mathbf{x}\|_{\mathcal{Y}}^2 dP(\mathbf{x}, \mathbf{y})$$

is minimized.

This is a classical learning problem that has been widely studied when  $\mathcal{Y} \subset \mathbb{R}^q$  has a small dimension. When the output dimension becomes very high, in order to generalize well one must take into account (i) correlation between output variables (ii) correlation between input variables  $\mathcal{X} \subset \mathbb{R}^p$  and (iii) correlation between input *and* output variables.

If prior knowledge about such correlations exists, it can be encoded into a regularizer. For example, a minimization scheme could be adopted that minimizes

$$\frac{1}{m} \sum_{i=1}^m \|\mathbf{y}_i - W\mathbf{x}_i\| + \sum_{i,j=1}^{\dim(\mathcal{X})} \sum_{s,t=1}^{\dim(\mathcal{Y})} W_{ij} W_{st} S_{ijst}.$$

Here,  $S_{ijst}$  encodes the correlation between inputs  $i, j$  with outputs  $s$  and  $t$ .

For example, suppose one is learning a mapping between two spaces of equal and large dimension, e.g. pairs of images or spectra. Then the most obvious prior knowledge one has is that, e.g., pixels in images that are close in the input are also close in the output. This knowledge can be encoded into  $S$ . The challenge is to rewrite such an optimization problem in the general case so that (i) it can be solved in a dual form to make it tractable for high dimension and (ii) it can be generalized with kernels to also solve nonlinear problems.

In this work we will show how to encode such prior knowledge by defining appropriate joint kernel functions and subsequent minimization in dual variables, building on work such as [7] and [8]. The subsequent algorithm will solve much more than linear regression: it will generalize nonlinear support vector machines for classification *and* regression, and will be also be able to deal with structured outputs such as strings, trees and graphs via kernels [8-10].

## 2 Linear Maps

We start by learning the linear map  $W$  such that a prediction on data is

$$\mathbf{y}(\mathbf{x}) = \operatorname{argmin}_{\mathbf{y} \in \mathcal{Y}} \|\mathbf{W}\mathbf{x} - \mathbf{y}\|^2 = \mathbf{W}\mathbf{x}.$$

Note that if the argmin is taken over a linear space, then  $\mathbf{y}(\mathbf{x}) = \mathbf{W}\mathbf{x}$ , but in more general settings, it will be necessary to compute it using other means. We consider an  $\varepsilon$ -insensitive loss approach, as in support vector regression [11]. We choose the  $W$  that minimizes

$$\|\mathbf{W}\|_{FRO}^2 \tag{9}$$

using the Frobenius norm, subject to

$$\|\mathbf{W}\mathbf{x}_i - \mathbf{y}\|^2 \geq \|\mathbf{W}\mathbf{x}_i - \mathbf{y}_i\|^2 + \varepsilon^2/2, \tag{10}$$

$$\forall i, \{\forall \mathbf{y} \in \mathcal{Y} : \|\mathbf{y}_i - \mathbf{y}\| > \varepsilon\}.$$

We note that this generalizes support vector classification and Regression:

- For  $\mathbf{y} \in \mathbb{R}$  one obtains support vector regression (SVR) [11] without threshold, and for  $\mathbf{y} \in \mathbb{R}^q$  one obtains vector-valued  $\varepsilon$ -insensitive SVR [12]. We rewrite (10) as

$$\min_{\mathbf{y} \in C_\varepsilon(\mathbf{y}_i)} \|\mathbf{W}\mathbf{x}_i - \mathbf{y}\|^2 \geq \|\mathbf{W}\mathbf{x}_i - \mathbf{y}_i\|^2 + \varepsilon^2/2$$

where  $C_\varepsilon(\mathbf{y}_i)$  is the complement of the ball of radius  $\varepsilon$  centered at  $\mathbf{y}_i$ . If  $\mathbf{W}\mathbf{x}_i$  is not in the latter ball, the value of this minimum is zero and the problem does not have any solution. On the other hand, if  $\mathbf{W}\mathbf{x}_i$  is in the ball, then this minimum is not zero and can be computed directly. Its value is attained for the following  $\mathbf{y}$ :

$$\mathbf{y} = \mathbf{y}_i + \frac{\mathbf{W}\mathbf{x}_i - \mathbf{y}_i}{\|\mathbf{W}\mathbf{x}_i - \mathbf{y}_i\|} \varepsilon.$$

The value of the minimum is then  $(\varepsilon - \|\mathbf{W}\mathbf{x}_i - \mathbf{y}_i\|)^2$ . We then have the constraint

$$(\varepsilon - \|\mathbf{W}\mathbf{x}_i - \mathbf{y}_i\|)^2 \geq \|\mathbf{W}\mathbf{x}_i - \mathbf{y}_i\|^2 + \varepsilon^2/2,$$

which gives, after some algebra,  $\|\mathbf{W}\mathbf{x}_i - \mathbf{y}_i\| \leq \varepsilon/4$ .

- For  $y \in \{\pm 1\}$  and  $0 \leq \varepsilon < 2$  we obtain two-class SVMs [11] ( $W$  is a  $1 \times p$  matrix). Expanding the constraint (10) for each  $i$  gives

$$-2y\mathbf{W}x_i + 2y_i\mathbf{W}x_i \geq \varepsilon^2/2.$$

For  $y, y_i \in \{\pm 1\}$ ,  $\|\mathbf{y}_i - \mathbf{y}\| > \varepsilon$  only occurs for  $y = -y_i$ , in which case we have  $y_i\mathbf{W}x_i \geq \varepsilon^2/8$ , the usual SVM constraints, disregarding scaling and threshold  $b$ .

- Similarly, for  $\mathbf{y} \in \{0, 1\}^q$ , where the  $c_i^{th}$  entry is 1 when example  $i$  is in class  $c_i$ , and 0 otherwise, and  $0 \leq \varepsilon < \sqrt{2}$  we can obtain multiclass SVMs [13]. As  $\|\mathbf{y}\| = 1$  we have the constraints

$$\mathbf{y}_i^\top W \mathbf{x}_i - \mathbf{y}^\top W \mathbf{x}_i \geq \varepsilon^2/4,$$

where the  $q$  rows of  $W = \begin{pmatrix} \mathbf{w}_1 \\ \dots \\ \mathbf{w}_q \end{pmatrix}$  correspond to the  $q$  hyperplanes of multi-class SVMs ( $W$  is a  $q \times p$  matrix). Because only one constraint is switched on at one time due to the zeros in  $\mathbf{y}$  we have to minimize  $\|W\|_{FRO}^2 = \sum_i \|\mathbf{w}_i\|^2$  subject to  $\forall i, \mathbf{w}_{c_i} \mathbf{x}_i - \mathbf{w}_j \mathbf{x}_i \geq \varepsilon^2/4, \forall j \in \{1, \dots, q\} \setminus c_i$  which is the same as in [13], again disregarding scaling and thresholds.

Generalizing to the non-separable case in the usual manner [8, 11] should be straightforward. Note that the constraints can also be written as:

$$\forall i, \{\forall \mathbf{y} \in \mathcal{Y} : \|\mathbf{y}_i - \mathbf{y}\| > \varepsilon\} : 2(\mathbf{y}_i - \mathbf{y})W \mathbf{x}_i \geq \varepsilon^2/2 + \|\mathbf{y}_i\|^2 - \|\mathbf{y}\|^2. \quad (11)$$

Let us now restrict ourselves slightly to the situation where the outputs are normalized so  $\forall \mathbf{y} \in \mathcal{Y} : \|\mathbf{y}\| = 1$ . (Obviously this is only useful in the multi-dimensional case.) Hence, we rewrite our optimization problem as: minimize

$$\|W\|_{FRO}^2 \quad (12)$$

subject to

$$\forall i, \{\forall \mathbf{y} \in \mathcal{Y} : \|\mathbf{y}_i - \mathbf{y}\| > \varepsilon\} : \mathbf{y}_i^\top W \mathbf{x}_i - \mathbf{y}^\top W \mathbf{x}_i \geq \varepsilon^2/4. \quad (13)$$

We can regard  $F(\mathbf{x}, \mathbf{y}) = \mathbf{y}^\top W \mathbf{x}$  as a function that returns the degree of fit between  $\mathbf{x}$  and  $\mathbf{y}$ . The output on a test point can now be written

$$\begin{aligned} \mathbf{y}(\mathbf{x}) &= \operatorname{argmin}_{\mathbf{y} \in \mathcal{Y}} \|W \mathbf{x} - \mathbf{y}\|^2 \\ &= \operatorname{argmax}_{\mathbf{y} \in \mathcal{Y}} \mathbf{y}^\top W \mathbf{x} = \frac{W \mathbf{x}}{\|W \mathbf{x}\|}. \end{aligned} \quad (14)$$

because, by Cauchy-Schwarz, the function  $\operatorname{argmax}_{\mathbf{y}} \mathbf{y}^\top W \mathbf{x}$  is maximal if  $\frac{\mathbf{y}}{\|\mathbf{y}\|}$  is parallel to  $W \mathbf{x}^*$ .

With this optimization problem for the case of discrete  $\mathcal{Y}$  and  $\varepsilon \rightarrow 0$ , we obtain the support vector machine for interdependent and structured output spaces (SVM-ISOS) of [8]. In practice, one could relax the restriction upon the normalization of  $\mathbf{y}$  during training because separability could still be obtained. However, if one is dealing with continuous outputs without this restriction then the preimage given by  $\operatorname{argmax}_{\mathbf{y} \in \mathcal{Y}} \mathbf{y}^\top W \mathbf{x}$  would not be well defined. This is the reason why in the work of [8] normalization was not an issue, as only the discrete output case was considered<sup>1</sup>.

We now show how to develop our method for joint kernels.

<sup>1</sup> In practice, in our experiments with joint kernels, we normalize the joint kernel itself, not the outputs, because the output in this case is not easily accessible.

### 3 Joint Kernel Maps

We can rewrite the last optimization problem by considering  $\mathbf{W}$  as a vector  $\mathbf{w}$  of dimension  $\dim(\mathcal{X})\dim(\mathcal{Y})$ , and choosing the feature map

$$\Phi_{\mathcal{X}\mathcal{Y}}(\mathbf{x}, \mathbf{y}) = \langle (\mathbf{x}\mathbf{y}^\top)_{ij} \rangle_{\substack{i=1, \dots, \dim(\mathcal{Y}) \\ j=1, \dots, \dim(\mathcal{X})}}.$$

The optimization problem then consists of minimizing<sup>2</sup>

$$\|\mathbf{w}\|^2 \tag{15}$$

subject to

$$\begin{aligned} \langle \mathbf{w}, \Phi_{\mathcal{X}\mathcal{Y}}(\mathbf{x}_i, \mathbf{y}_i) - \Phi_{\mathcal{X}\mathcal{Y}}(\mathbf{x}_i, \mathbf{y}) \rangle &\geq \varepsilon^2/2, \\ \forall i, \{\forall \mathbf{y} \in \mathcal{Y} : \|\mathbf{y}_i - \mathbf{y}\| > \varepsilon\}. \end{aligned} \tag{16}$$

However, we are free to choose another mapping, as we shall see later (indeed, choosing a mapping which incorporates prior knowledge is the whole point of using this approach).

We call  $\Phi_{\mathcal{X}\mathcal{Y}}$  the *joint kernel map* (JKM), and

$$J((\mathbf{x}, \mathbf{y}), (\hat{\mathbf{x}}, \hat{\mathbf{y}})) = \Phi_{\mathcal{X}\mathcal{Y}}(\mathbf{x}, \mathbf{y})^\top \Phi_{\mathcal{X}\mathcal{Y}}(\hat{\mathbf{x}}, \hat{\mathbf{y}})$$

the *joint kernel*. This relates our method to the work of [14] and [15].

Constructing the corresponding dual problem we obtain: maximize<sup>3</sup>

$$\frac{\varepsilon^2}{4} \sum_{i, \mathbf{y} : \|\mathbf{y}_i - \mathbf{y}\| > \varepsilon} \alpha_{i\mathbf{y}} - \frac{1}{2} \sum_{\substack{i, \mathbf{y} : \|\mathbf{y}_i - \mathbf{y}\| > \varepsilon \\ j, \hat{\mathbf{y}} : \|\mathbf{y}_i - \hat{\mathbf{y}}\| > \varepsilon}} \alpha_{i\mathbf{y}} \alpha_{j\hat{\mathbf{y}}} \langle \Phi_{\mathcal{X}\mathcal{Y}}(\mathbf{x}_i, \mathbf{y}_i) - \Phi_{\mathcal{X}\mathcal{Y}}(\mathbf{x}_i, \mathbf{y}), \Phi_{\mathcal{X}\mathcal{Y}}(\mathbf{x}_j, \mathbf{y}_j) - \Phi_{\mathcal{X}\mathcal{Y}}(\mathbf{x}_j, \hat{\mathbf{y}}) \rangle$$

subject to

$$\alpha_{ij} \geq 0, \quad i = 1, \dots, m, \{\forall \mathbf{y} \in \mathcal{Y} : \|\mathbf{y}_i - \mathbf{y}\| > \varepsilon\}.$$

<sup>2</sup> Note that we could also simplify the optimization problem further by splitting the constraints: i.e. minimize  $\|\mathbf{w}\|^2$  subject to

$$\begin{aligned} \forall i : \langle \mathbf{w}, \Phi_{\mathcal{X}\mathcal{Y}}(\mathbf{x}_i, \mathbf{y}_i) \rangle + b &\geq \varepsilon^2/8 \\ \{\forall \mathbf{y} \in \mathcal{Y} : \|\mathbf{y}_i - \mathbf{y}\| > \varepsilon\} : \langle \mathbf{w}, \Phi_{\mathcal{X}\mathcal{Y}}(\mathbf{x}_i, \mathbf{y}) \rangle + b &\leq -\varepsilon^2/8. \end{aligned}$$

If this problem is linearly separable, then its solution  $\mathbf{w}$  is also a feasible solution of (15)-(16).

<sup>3</sup> Note that with infinitely many constraints, standard duality does not apply for our optimization problem. However, for the purposes of the present paper, we are not concerned with this. For practical purposes, we may assume that for any  $\varepsilon > 0$ , our data domain has a finite  $\varepsilon$ -cover (e.g., our domain could be a compact subset of  $\mathbb{R}^n$ ). Since on a computer implementation, a constraint can only be enforced up to machine precision, we can thus imagine choosing a sufficiently small  $\varepsilon$ , which reduces our setting to one with a finite number of constraints. Furthermore, we find experimentally that the number of active constraints is small and scales sublinearly with the number of examples or output dimension (see Figure 1).

The objective can be rewritten with kernels:

$$\frac{\varepsilon^2}{4} \sum_{i, \mathbf{y}: \|\mathbf{y}_i - \mathbf{y}\| > \varepsilon} \alpha_{i\mathbf{y}} - (1/2) \sum_{\substack{i, \mathbf{y}: \|\mathbf{y}_i - \mathbf{y}\| > \varepsilon \\ j, \hat{\mathbf{y}}: \|\mathbf{y}_j - \hat{\mathbf{y}}\| > \varepsilon}} \alpha_{i\mathbf{y}} \alpha_{j\hat{\mathbf{y}}} [J((\mathbf{x}_i, \mathbf{y}_i), (\mathbf{x}_j, \mathbf{y}_j)) \\ - J((\mathbf{x}_i, \mathbf{y}_i), (\mathbf{x}_j, \hat{\mathbf{y}})) - J((\mathbf{x}_i, \mathbf{y}), (\mathbf{x}_j, \mathbf{y}_j)) \\ + J((\mathbf{x}_i, \mathbf{y}), (\mathbf{x}_j, \hat{\mathbf{y}}))].$$

The standard linear map therefore requires  $J((\mathbf{x}_i, \mathbf{y}_i), (\mathbf{x}_j, \mathbf{y}_j)) = \langle \mathbf{x}_i, \mathbf{x}_j \rangle \langle \mathbf{y}_i, \mathbf{y}_j \rangle = K(\mathbf{x}_i, \mathbf{x}_j)L(\mathbf{y}_i, \mathbf{y}_j)$ , where  $K(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i, \mathbf{x}_j \rangle$  and  $L(\mathbf{y}_i, \mathbf{y}_j) = \langle \mathbf{y}_i, \mathbf{y}_j \rangle$  are kernel maps for input and output respectively.

Now

$$\mathbf{w} = \sum_{i, \mathbf{y}: \|\mathbf{y}_i - \mathbf{y}\| > \varepsilon} \alpha_{i\mathbf{y}} [\Phi_{\mathcal{X}\mathcal{Y}}(\mathbf{x}_i, \mathbf{y}_i) - \Phi_{\mathcal{X}\mathcal{Y}}(\mathbf{x}_i, \mathbf{y})].$$

For certain joint kernels (that are linear in the outputs) we can compute the matrix  $W$  explicitly to calculate the mapping. However, for general nonlinear mappings of the output (or input) we must solve the pre-image problem (cf. (14)):

$$\begin{aligned} \mathbf{y}(\mathbf{x}^*) &= \operatorname{argmax}_{\mathbf{y} \in \mathcal{Y}} \langle W, \Phi_{\mathcal{X}\mathcal{Y}}(\mathbf{x}^*, \mathbf{y}) \rangle \\ &= \operatorname{argmax}_{\mathbf{y} \in \mathcal{Y}} \sum_{i, \mathbf{y}: \|\mathbf{y}_i - \mathbf{y}\| > \varepsilon} \alpha_{i\mathbf{y}} J((\mathbf{x}_i, \mathbf{y}_i), (\mathbf{x}^*, \mathbf{y}^*)) \\ &\quad - \alpha_{i\mathbf{y}} J((\mathbf{x}_i, \mathbf{y}), (\mathbf{x}^*, \mathbf{y}^*)). \end{aligned}$$

In the next section we discuss joint kernels, and consider several examples that do not require one to solve the general pre-image problem. First, let us discuss related work, and practical implementation considerations.

*Optimization.* Finding a solution to the above equations, which contain an infinite number of constraints, is feasible because in practice the solution tends to be very sparse. In fact, the solution can be found in polynomial time if the pre-image can be computed in polynomial time. An efficient method for the SVM for Interdependent and Structured Output Spaces was developed in [8] and can be analogously implemented for Joint Kernel Maps by using an iterative scheme: add the most violating example to the working set and reoptimize, repeating until completion. One can then show that on each iteration the objective function strictly improves and is guaranteed to terminate if the problem is separable. In practice, in our experiments we also start with  $\varepsilon$  large, and decrease it upon separability.

*Related Algorithms.* The idea of learning maps by embedding both input and output spaces using kernels was first employed in the Kernel Dependency Estimation algorithm [7], where the kernels were defined separately. This allowed correlations to be encoded between output features, nonlinear loss functions to be defined, and for outputs to be structured objects such as strings and trees [8-10] (however, one must then solve an often difficult pre-image problem). The method first decorrelates the outputs

via performing a kernel principal component analysis (kPCA). kPCA yields principal components  $v_l \in \mathbb{R}^q, l = 1 \dots n$  and corresponding variances  $\lambda_l$ . Henceforth the output labels  $\{y_i\}_{i=1}^m$  are projected to the column vectors  $v_l$  to retrieve the  $m$  principal coordinates  $z_i \in \mathbb{R}^n$ . This projection results in the new estimation task

$$\arg \min_{W \in \mathbb{R}^{n \times p}} \sum_{i=1}^m \|z_i - Wx_i\|^2.$$

KDE for example performs a ridge regression on each component  $z_{ij}, 1 \leq j \leq n$  to overcome overfitting. Predictions for a new point  $x^*$  are made via predicting first the principal coordinates  $z^* = Wx^*$ , and then using the principal components.

$$y^* = Vz^*.$$

Here  $V \in \mathbb{R}^{q \times n}$  consists of the  $n$  principal components  $v_l$ . In the case where  $n = q$  the prediction performance will only depend on the basic regression used for estimating  $z^*$  since  $V$  acts as a basis transformation.

If one assumes that the main variation in the output are according to signal and the small variances according to noise, then it is reasonable to take the first  $n$  principal components corresponding to the largest variance  $\lambda_l$ . Alternatively, instead of cutting off it is also possible to *shrink* the directions according their variance.

Compared to the current work and work such as SVM-ISOS [8], KDE has the advantage during training of not requiring the computation of pre-images. On the other hand, it requires an expensive matrix inversion step, and does not give sparse solutions. The inability to use Joint Kernels in KDE means that prior knowledge cannot be so easily encoded into the algorithm. In our experiments (see Section 5) the difference between using this prior knowledge or not in real applications can be large, at least for small sample size.

The authors of [16] also provide a method of using kernels to deal with high-dimensional output regression problems using vector-valued kernel functions. One defines a prediction function as follows:

$$f(\mathbf{x}) = \sum_{i=1}^m K(\mathbf{x}_i, \mathbf{x}) \mathbf{c}_i$$

where  $K(\mathbf{x}_i, \mathbf{x}_j)$  is a  $q$  by  $q$  matrix which in position  $K_{s,t}$  encodes the similarity between training points  $i$  and  $j$  with respect to outputs  $s$  and  $t$ . The weights  $\mathbf{c}_i$  are hence  $q$  by 1 vectors. Although at first sight this approach seems very complicated in terms of defining kernels, there are some natural examples where known correlation across outputs can be encoded. However, simply minimizing  $\sum_i \|y_i - f(\mathbf{x}_i)\|^2$  yields a large, non-sparse optimization problem with  $qm$  variables.

Considering once again classification problems, the current work also turns out to have strong relations with the work of [15] who employed a ranking perceptron algorithm and a specific joint kernel on the natural language problem of parsing (outputting a parse tree). In this case, the difficult pre-image problem was avoided by only selecting among  $n$  pre-selected experts (parsing algorithms). The algorithm they used is

thus similar to the one given in footnote 2, except in their case not all possible negative constraints are enforced, but only  $n - 1$  per example. Using the multi-class SVM formulation of [11, 13]:

$$f(\mathbf{x}_i, \mathbf{y}_i) > f(\mathbf{x}_i, \mathbf{y}), \quad \forall \{\mathbf{y} \in \mathcal{Y} \setminus \mathbf{y}_i\} \quad (17)$$

and considering  $\mathcal{Y}$  as some large set, e.g. of structured objects, one arrives at the formulation of SVM-ISOS [8]. Essentially, this is a special case of our algorithm, where the output is structured (discrete  $\mathcal{Y}$ ) and  $\varepsilon = 0^4$ . The authors apply the algorithm to problems of label sequence learning, named entity recognition and others. Our work complements this last one in helping to understand the role of joint kernels in learning problems where one can supply prior knowledge by way of the similarity measure. The authors of [17] also provide a similar formulation to [8] but with a probabilistic interpretation.

Although in this paper we do not consider structured output problems, the algorithm we develop could indeed be applied to such problems. Let us consider one such problem, machine translation: translating a sentence into another language. The relation between regression and classification in this framework is an interesting one. On the one hand, one could argue that one desires separability, to return the correct pre-image (sentence) on the training set. This is the approach of [8]. On the other hand, to classify one sentence as correct, and all others as wrong as in the constraints of (17) could be dangerous because it ignores the distance measure in the output space (other sentences may also be plausible.) Thus even when the embedding is discrete, it may make sense to treat it as regression if outputs close in output space have the same ‘label’. Although the authors of [8] try to fix this problem with an adaptive soft margin approach, the  $\varepsilon$ -insensitive approach of the current paper would preserve sparsity.

## 4 Joint Kernels

A joint kernel is a nonlinear similarity measure between input-output pairs, i.e.,  $J((\mathbf{x}, \mathbf{y}), (\mathbf{x}', \mathbf{y}'))$  where  $(\mathbf{x}, \mathbf{y})$  and  $(\mathbf{x}', \mathbf{y}')$  are labeled training examples,<sup>5</sup>

$$J((\mathbf{x}, \mathbf{y}), (\mathbf{x}', \mathbf{y}')) = \langle \Phi_{\mathcal{X}\mathcal{Y}}(\mathbf{x}, \mathbf{y}), \Phi_{\mathcal{X}\mathcal{Y}}(\mathbf{x}', \mathbf{y}') \rangle,$$

where  $\Phi_{\mathcal{X}\mathcal{Y}}$  is a map into a dot product space. All functions  $J((\mathbf{x}, \mathbf{y}), (\mathbf{x}', \mathbf{y}'))$  that take this form are positive definite, and all positive definite kernels  $J((\mathbf{x}, \mathbf{y}), (\mathbf{x}', \mathbf{y}'))$  can be written in this form. This follows directly from the corresponding statements for kernels  $k(\mathbf{x}, \mathbf{x}')$  (see, for example, [5]). The point of a joint kernel is to describe the similarity between input-output pairs by mapping pairs into a joint space. A joint kernel can encode more than just information about inputs or outputs independent of each other: it can also encode known dependencies/correlations between inputs and outputs.

<sup>4</sup> Ignoring the normalization conditions on the output which come from our original derivation, as discussed previously.

<sup>5</sup> Note there is nothing stopping us considering not just pairs here but also kernels on  $n$ -tuples, e.g., of the form  $(\mathbf{x}, \mathbf{y}, \mathbf{z})$ .

Joint Kernels have already begun to be studied ([14],[8]); however, so far only discrete output spaces and structured outputs (such as sequences) were considered. One of the problems with Joint Kernels is that only for a subset of possible kernels can one compute the pre-image easily. In [8] kernels on sequences are chosen that are amenable to dynamic programming. Although some methods for speeding up pre-image computations exist [18, 19], this remains a difficult problem. In the following we describe some kernels which avoid complex pre-image problems.

*Tensor Product Kernels.* A kernel that does not encode any correlations can be obtained by using the product

$$J_{\text{LINEAR}}((\mathbf{x}, \mathbf{y}), (\mathbf{x}', \mathbf{y}')) = K(\mathbf{x}, \mathbf{x}')L(\mathbf{y}, \mathbf{y}') \\ = \langle \Phi_{\mathcal{X}}(\mathbf{x}), \Phi_{\mathcal{X}}(\mathbf{x}') \rangle \langle \Phi_{\mathcal{Y}}(\mathbf{y}), \Phi_{\mathcal{Y}}(\mathbf{y}') \rangle$$

where  $K$  and  $L$  are respectively kernels on the inputs and outputs. If  $K$  and  $L$  are positive definite, then  $J$  will be, too; moreover, the associated feature space is known to be the tensor product of the individual feature spaces.

An interesting special case is when  $L$  is a linear kernel. In that case

$$W_{\text{LINEAR}} = \sum_{i, \mathbf{y}: \|\mathbf{y}_i - \mathbf{y}\| > \epsilon} \alpha_{ij} \Phi_{\mathcal{X}}(\mathbf{x}_i) \mathbf{y}_i^{\top} - \alpha_{ij} \Phi_{\mathcal{X}}(\mathbf{x}_i) \mathbf{y}^{\top}.$$

When  $\dim(\mathcal{X})$  or  $\dim(\mathcal{Y})$  are very large it can be more efficient to avoid the calculation of  $W$  and calculate a test prediction directly:

$$W_{\text{LINEAR}} \mathbf{x} = \sum_{i, \mathbf{y}: \|\mathbf{y}_i - \mathbf{y}\| > \epsilon} \alpha_{ij} K(\mathbf{x}_i, \mathbf{x}) \mathbf{y}_i^{\top} - \alpha_{ij} K(\mathbf{x}_i, \mathbf{x}) \mathbf{y}^{\top}.$$

Hence we avoid difficult pre-image problems in this case.

*Diagonal Regularization.* Consider the case where  $\dim(\mathcal{X}) = \dim(\mathcal{Y})$ , and it is known that one is looking for a linear map where the true matrix  $W$  is close to the identity map. Slightly more generally, one may know that the  $n^{th}$  dimension of the input is correlated with the  $n^{th}$  dimension of the output. Instances of such problems include decoding mass spectrometry (mapping from observed to theoretical spectra) and image mapping problems (deblurring, morphing, etc.). This correlation can be directly encoded:

$$J_{\text{DIAG}}((\mathbf{x}, \mathbf{y}), (\mathbf{x}', \mathbf{y}')) = \\ (1 - \lambda)K(\mathbf{x}, \mathbf{x}')\langle \mathbf{y}, \mathbf{y}' \rangle + \lambda \left[ \sum_{k=1}^q x_k x'_k y_k y'_k \right] \tag{18}$$

where  $\lambda$  controls the amount of encoded correlation. If  $\lambda$  is large, then the  $n^{th}$  dimension in the input is presumed highly correlated with the  $n^{th}$  dimension in the output, and the similarity measure is dominated by these relationships. Algorithms that minimize



the Frobenius norm choose these dimensions as relevant. Furthermore, the solution is still linear (does not require a pre-image) because we can write

$$W_{\text{DIAG}}\mathbf{x} = (1 - \lambda)W_{\text{LINEAR}}\mathbf{x} + \lambda \sum_{i, \mathbf{y}: \|\mathbf{y}_i - \mathbf{y}\| > \varepsilon} \alpha_{ij} [\text{DIAG}(\mathbf{x}_i \mathbf{y}_i^\top) - \text{DIAG}(\mathbf{x}_i \mathbf{y}^\top)] \mathbf{x}.$$

where  $D = \text{DIAG}(M)$  is a diagonal matrix with  $D_{ii} = M_{ii}$ .

*Patch-Wise Correlation.* The natural generalization of the previous kernel is when you know that the  $n^{\text{th}}$  dimension of the output is strongly correlated with a known set of dimensions in the input; e.g., for mappings between images, one could know that a region in the output image is strongly correlated with a region in the input image. This knowledge can be encoded with the kernel

$$J_{\text{PATCH}}((\mathbf{x}, \mathbf{y}), (\mathbf{x}', \mathbf{y}')) = (1 - \lambda)K(\mathbf{x}, \mathbf{x}')\langle \mathbf{y}, \mathbf{y}' \rangle + \lambda \sum_{k=1}^{|\mathcal{P}|} \left[ \sum_{p \in \mathcal{P}_k} \mathbf{x}_p \mathbf{x}'_p \sum_{p \in \mathcal{P}_k} \mathbf{y}_p \mathbf{y}'_p \right] \quad (19)$$

where  $\mathcal{P}$  is the set of known correlated patches. This encodes patch correlation between dimensions in  $\mathbf{x}$ , between dimensions in  $\mathbf{y}$ , and correlation between input and output, i.e. between  $\mathbf{x}$  and  $\mathbf{y}$ .<sup>6</sup> The evaluation on a test example can be expressed as:

$$W_{\text{PATCH}}\mathbf{x} = (1 - \lambda)W_{\text{LINEAR}}\mathbf{x} + \lambda \sum_{i, \mathbf{y}: \|\mathbf{y}_i - \mathbf{y}\| > \varepsilon} \alpha_{ij} \left[ \sum_{k=1}^{|\mathcal{P}|} P_k(\mathbf{x}_i \mathbf{y}_i^\top) - \sum_{k=1}^{|\mathcal{P}|} P_k(\mathbf{x}_i \mathbf{y}^\top) \right] \mathbf{x}$$

where  $P = P_k(M)$  is a matrix such that  $P_{ij} = M_{ij}$  if  $i \in \mathcal{P}_k$  or  $j \in \mathcal{P}_k$  (if  $i$  or  $j$  are in the  $k^{\text{th}}$  patch), or  $P_{ij} = 0$ , otherwise.

*Image Reconstruction.* Consider the problem of image reconstruction. For example, in a problem of digit reconstruction one should predict the bottom half of a digit given its top half. The authors of [7] solved such a problem with the KDE algorithm. The input and output kernels,  $K$  and  $L$ , used by that algorithm are separate and the algorithm is not given in advance prior knowledge that the two images are related, i.e. that their concatenation creates a single image. The kernels used were

$$K(\mathbf{x}, \mathbf{x}') = \exp(-\|\mathbf{x} - \mathbf{x}'\|^2 / (2\sigma^2))$$

$$L(\mathbf{y}, \mathbf{y}') = \exp(-\|\mathbf{y} - \mathbf{y}'\|^2 / (2(\sigma^*)^2)) \quad (20)$$

<sup>6</sup> One can introduce a weighting function over the patches, corresponding to the assumption that the closer the pixels are, the more reliable is their correlation, cf. [5, Eq. (13.21)].

In that work it was apparent that sometimes in the middle of the digit this approach can cause some ‘glitches’ when the two halves are connected together. A simple joint kernel such as

$$J_{\text{RBF}}((\mathbf{x}, \mathbf{y}), (\mathbf{x}', \mathbf{y}')) = \exp(-\|(\mathbf{x}, \mathbf{y}) - (\mathbf{x}', \mathbf{y}')\|^2 / (2\sigma^2))$$

(i.e. concatenating the images together, and then taking the RBF kernel) could capture more of the problem than taking the product of the kernels in (20). The joint kernel given here would take into account nonlinearities between pixels of input and output dimensions. To improve this method further, invariances could also be encoded into the kernel, e.g. by concatenating the input and output images and then taking into account rotations, translations, etc. A local polynomial kernel [11] which takes encodes spatial information within the image would also help to encode the mapping between input and output; i.e., it would encode that the pixels at the very bottom of the input are highly correlated with the pixels on the top of the output, as before.

## 5 Experiments

As said before, JKM reduces to support vector classification and regression for particular  $\mathcal{Y}$ . We therefore only test our algorithm on regression problems of multiple outputs, and show how employing joint kernels can benefit in this case.

### 5.1 Artificial Problem: The Identity Map

We performed a first experiment on toy data to demonstrate the potential of the approach. We chose a very simple problem: the input are  $x_i \in R^p$ , each dimension drawn independently from a normal distribution of mean 0, standard deviation 1. The output is the same as the input,  $y_i = x_i$ , i.e. the task is to learn the identity map.

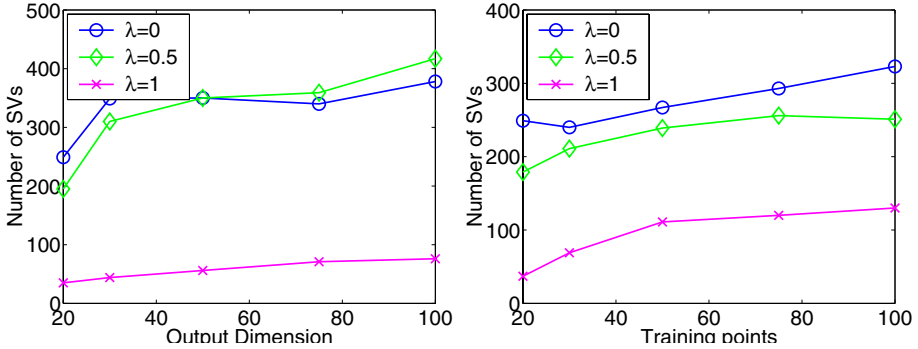
**Table 1.** Mean squared error for different joint kernels encoding the identity map (first three rows) compared to ridge regression (RR) and  $k$ -nearest neighbors. Incorporating prior knowledge in the joint kernel approach ( $\lambda > 0$ ) improves generalization performance

| $\dim(\mathcal{X}) = \dim(\mathcal{Y})$ | 20   | 30   | 50   | 75   | 100  |
|-----------------------------------------|------|------|------|------|------|
| JKM <sub>DIAG</sub> ( $\lambda = 1$ )   | 0.00 | 0.00 | 0.01 | 0.02 | 0.02 |
| JKM <sub>DIAG</sub> ( $\lambda = 0.5$ ) | 0.03 | 0.14 | 0.34 | 0.50 | 0.62 |
| JKM <sub>DIAG</sub> ( $\lambda = 0$ )   | 0.06 | 0.40 | 0.78 | 1.00 | 1.14 |
| RR (best $\gamma$ )                     | 0.06 | 0.43 | 0.82 | 1.07 | 1.21 |
| $k$ -NN (best $k$ )                     | 0.92 | 1.09 | 1.27 | 1.40 | 1.47 |

We compared  $k$ -nearest neighbor and ridge regression with our approach. For the former ( $k$ -NN and RR) we chose the best possible parameters, for the latter (JKM) we show the results for the identity-map regularizing joint kernel (18) for  $\lambda = 0, \frac{1}{2}$  and 1, with  $\varepsilon = \frac{0.5}{\sqrt{p}}$ . For  $\lambda = 0$  the set of possible linear maps is free; for  $\lambda = 1$  only linear maps that are diagonal matrices are considered.

The mean squared error for  $p = 20, \dots, 100$  features are given in Table 1, with 20 examples for training and 100 for testing, averaged over 20 runs. A Wilcoxon signed

ranked test confirms that the two kernels with  $\gamma > 0$  outperform the other techniques. Further experiments adding noise to the dataset (not shown) yielded similar conclusions. Figure 1 shows the number of active constraints (support vectors) for varying output dimensions with training size 20 (left) and varying training set sizes with output dimension 20 (right). The solutions are relatively sparse (consider that dual ridge regression [20] uses  $pm$  variables for  $p$  outputs and  $m$  examples). Note that larger values of  $\lambda$  (where the capacity of the set of functions is lower) have less active constraints.



**Fig. 1. Number of Active Constraints (Support Vectors) on Artificial data** varying output dimension (left) and training set size (right)

## 5.2 Image Mapping: Learning to Smile

We consider the problem of mapping from the image of a face with a plain expression to an image of the same person smiling using images from the MPI face database [21, 22]. We use 20 examples for training, and 50 for testing. The images are  $156 \times 176 = 27456$  pixels. We selected a small number of training examples because in this setting the weakness of existing methods was further exposed.

We applied a joint kernel mapping using the tensor product (linear) kernel ( $\epsilon = 0.05$ ) and the patch-wise kernel (19) with  $\gamma = 0.95$ ,  $\epsilon = 0.1$  and patches of size  $10 \times 10$  which overlap by 5 pixels. Training took 344 and 525 steps of adding a single violating example for the linear and patch kernels, resulting in 150 and 162 support vectors, respectively. Again, we compared with conventional regression techniques, choosing their best possible hyperparameters. A naive employment of ridge regression on this task fails, outputting a kind of “average” face image, independent of the input, see Figure 2. The large dimensionality means there are many solutions with low empirical error, RR (after choosing the optimal regularization constant) selects one that uses many (irrelevant) inputs due to its regularizer. Similarly,  $k$ -NN cannot solve this problem well for small sample size. See Figure 2 for example images, and Table 2 for mean squared error rates comparing all these methods. By way of comparison, the baseline of simply predicting the input image as the output (the plain expression) gives a test error of  $0.1823 \pm 0.003$ .



**Fig. 2.** Prediction of smiling face given plain expression by joint kernel maps (patch and linear) and ridge regression and  $k$ -NN. The large dimensionality means there are many solutions with low empirical error, RR (after choosing the optimal regularization constant) selects one that uses many (irrelevant) inputs due to its regularizer  $\|w\|^2$  which favors non-sparse solutions. Only the Patch-Kernel Joint Kernel Map is successful, as the choice of (joint) kernel limits the possible choice of functions to ones which are close to the identity map

**Table 2.** Test error on the smiling problem of the MPI face database

|            | JKM–<br>PATCH<br>( $\varepsilon = 0.1$ ) | JKM–<br>LINEAR<br>( $\varepsilon = 0.05$ ) | RR<br>(best $\gamma$ ) | $k$ -NN<br>(best $k$ ) |
|------------|------------------------------------------|--------------------------------------------|------------------------|------------------------|
| Test error | 0.142                                    | 0.227                                      | 0.222                  | 0.244                  |
| Test error | $\pm 0.002$                              | $\pm 0.006$                                | $\pm 0.006$            | $\pm 0.006$            |

### 5.3 Conclusions

In this work we presented a general method of supervised learning via joint kernel mappings, and showed how such kernels can encode certain regularization properties which reflect prior knowledge in mappings. While the experiments shown here used only simple types of joint kernels taking advantage of patch-wise information, these examples are only an instantiation of our approach, to show its validity and to bring insight into why and how joint kernels are useful. Joint kernels are mainly useful in cases where their pre-image is easily computable, and are extendable to complex outputs such as strings, trees and graphs. Indeed, we believe the gain of joint kernel methods is in employing such complex structured outputs that go beyond standard classification and regression such as in parsing, machine translation and other applications. In those cases the difference between coding prior knowledge into a joint kernel and using two separate kernels for input and output could potentially be large, at least in the small sample size case. Although first studies in some of these areas have been completed [8, 15], no study that we know of has yet directly compared this benefit.

Future work should also address issues of efficiency (efficiency of training, pre-images for more complex nonlinear and structured kernels), and to more deeply explore applications of these results.

**Acknowledgments.** We thank Christian Wallraven for providing the MPI face data. We thank André Elisseeff, Goekhan Bakır, Jan Eichorn, Olivier Chapelle, Arthur Gretton, Tobias Mann, William Stafford Noble, Massimiliano Pontil, Fabian Sinz and Thomas Hofmann for useful discussions.

### References

1. J. Mercer. Functions of positive and negative type and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society, London*, A 209: 415446, 1909.
2. M. A. Aizerman, E. M. Braverman, and L. I. Rozonoer. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821837, 1964.
3. B. E. Boser, I. M. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In D. Haussler, editor, *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pages 144152, Pittsburgh, PA, July 1992. ACM Press.
4. V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer Verlag, New York, 1995.
5. B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

6. G. S. Kimeldorf and G. Wahba. Some results on Tchebycheffian spline functions. *Journal of Mathematical Analysis and Applications*, 33:8295, 1971.
7. J. Weston, O. Chapelle, A. Elisseeff, B. Schölkopf, and V. Vapnik. Kernel dependency estimation. *Neural Processing Information Systems* 15, 2002.
8. I. Tschantaris, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. *ICML*, 2004.
9. D. Haussler. Convolution kernels on discrete structure. Technical report, UC Santa Cruz, 1999.
10. C. Watkins. Dynamic alignment kernels. In A. J. Smola, P. L. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 3950, Cambridge, MA, 2000. MIT Press.
11. V. N. Vapnik. *Statistical Learning Theory*. Springer, 1998.
12. F. Pérez-Cruz, G. Camps, E. Soria, J Pérez, A.R. Figueiras-Vidal, and A. Artés-Rodríguez. Multi-dimensional function approximation and regression estimation. In *ICANN*, 2002.
13. J. Weston and C. Watkins. Multi-class support vector machines. *Royal Holloway Technical Report CSD-TR-98-04*, 1998.
14. T. Hofmann, I. Tschantaris, and Y. Altun. Learning over discrete output spaces via joint kernel functions. *Kernel Methods Workshop, Neural Processing Information Systems* 15, 2002.
15. M. Collins and N. Duffy. Convolution kernels for natural language. *Neural Processing Information Systems* 14, 2001.
16. C.A. Micchelli and M. Pontil. On learning vectorvalued functions. *Research Note RN/03/08*, Dept of Computer Science, UCL, 2003.
17. C. Guestrin B. Taskar and D. Koller. Max-margin markov networks. *Neural Information Processing Systems* 16, 2003.
18. J. T. Kwok and I. W. Tsang. Finding the pre-images in kernel principal component analysis. *6th Annual Workshop On Kernel Machines*, Whistler, Canada, 2002.
19. G .H. Bakir, J. Weston, and B. Schölkopf. Learning to find pre-images. *Advances in Neural Information Processing Systems* 16, 2004.
20. Craig Saunders, Alexander Gammerman, and Volodya Vovk. Ridge regression learning algorithm in dual variables. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 51521. Morgan Kaufmann Publishers Inc., 1998.
21. Max Planck Institute Face Database. <http://faces.kyb.tuebingen.mpg.de/>.
22. V. Blanz and T. Vetter. A morphable model for the synthesis of 3d faces. *SIGGRAPH99*, pages 187194, 1999.

# Statistical Ensemble Method (SEM): A New Meta-machine Learning Approach Based on Statistical Techniques

Andrés Yáñez Escolano, Pedro Galindo Riaño, Joaquin Pizarro Junquera,  
and Elisa Guerrero Vázquez

Universidad de Cádiz, Departamento de Lenguajes y Sistemas Informáticos,  
Grupo de "Sistemas Inteligentes de Computación",  
C.A.S.E.M. 11510 – Puerto Real (Cádiz), Spain  
{andres.yaniez, pedro.galindo, joaquin.pizarro,  
elisa.guerrero}@uca.es

**Abstract.** The goal of combining the outputs of multiple models is to form an improved meta-model with higher generalization capability than the best single model used in isolation. Most popular ensemble methods do specify neither the number of component models nor their complexity. However, these parameters strongly influence the generalization capability of the meta-model. In this paper we propose an ensemble method which generates a meta-model with optimal values for these parameters. The proposed method suggests using resampling techniques to generate multiple estimations of the generalization error and multiple comparison procedures to select the models that will be combined to form the meta-model. Experimental results show the performance of the model on regression and classification tasks using artificial and real databases.

## 1 Introduction

The aim of machine learning is to make a good model based on a set of examples. The goal is not to learn an exact representation of the training data itself, but rather to build a statistical model of the process which generates the data [1]. Classic methods for model building choose a model from a set of competing alternatives, assigning a single measure of generalization error to each candidate. The model which minimizes this value is selected and the rest is discarded.

However, when several models show similar generalization errors, we should not conclude that the model having the best performance on the validation set will achieve the best performance on new test data, given that we are working with a noisy, finite learning dataset. Therefore, any chosen hypothesis will be only an estimate of the real target and, like any estimate, will be affected by a bias and a variance term. Furthermore, there is another disadvantage with such approach: all the effort involved in generating the remaining near-optimal models is wasted. These drawbacks can be overcome by combining these models.

Model combination approach leads to significant improvements of new prediction with a little additional computational effort. It is possible to identify two main

approaches to combining models: multiexpert and multistage methods. Multiexpert methods work in parallel, while multistage methods use a serial approach where the next model is trained/consulted only for examples rejected by the previous models. Two main groups of meta-machine learning methods which work in parallel exist: mixture of experts [7] and ensemble methods. While mixture of experts approach divides the input space with a gating network and allocates the subspaces to different experts (models), the output of an ensemble is generated by the weighted outputs of each model. The performance of an ensemble can be better than the performance of the best single model used in isolation when the models are accurate enough and fairly independent in the errors they make [6], [9]. The meta-machine learning method proposed in this paper is based on the last approach.

## 2 The Statistical Ensemble Method

In general, an ensemble is built in two steps: a) Generation/selection of a number of component models. b) Combination of their predictions.

The most prevailing approaches for generating component models are based on varying the topology, the algorithm, the set of initial parameters to be used in the iterative learning process (i.e. random weights for MLPs) or the data itself (subsampling the training examples, manipulating the input features or the output targets and injecting randomness) [3],[16]. However, most popular ensemble methods specify neither the number of component models nor their complexity, and, obviously, these parameters strongly influence the generalization capability of the ensemble. In this paper we propose a methodology to generate a meta-model with optimal values for these parameters.

The steps of the proposed methodology may be outlined as follow:

1. Obtain multiple generalization error measures for each model using resampling techniques. The use of a set of estimations instead of a single measure of generalization error for model selection was proposed in [14],[5].
2. Determine the set of models whose errors are not significantly different from the model with minimum estimated generalization error using statistical tests [18] for comparing groups of paired samples (multiple comparison procedures).
3. Combine these near-optimal models using ensemble methods.

### 2.1 Multiple Error Measures Using Resampling Techniques

Resampling methods for estimating the generalization error generate multiple test-and-train datasets, and estimate the generalization error as the average of the validation errors.

The main approaches to resampling are the following:

- Random hold-out: many randomly train-and-test sets are generated. The examples are selected without replacement.
- K-fold cross-validation: examples are randomly divided into  $k$  mutually exclusive partitions of approximately equal size. Each model is trained and tested  $k$  times; each time tested on a fold and trained on the dataset minus the fold.



- Leave-one-out: it is a special case of  $k$ -fold cross validation, where  $k$  equals the sample size.
- Bootstrapping: instead of repeatedly analyzing subsets of the data, you repeatedly analyze subsamples of the data. Each subsample is a random sample with replacement from the full dataset and constitutes a training set. Examples not found in the training set form the validation set.

In the above methods, the estimate of generalization error is taken as the average of the estimated accuracies (validation errors) from the different train/test sets. In the proposed methodology, this estimate will be used to determine the reference model, but all validation errors obtained from each train/test pair will be kept in order to be able to apply statistical tests to compare groups of related samples, instead of comparing a single estimate of the generalization error. The whole process may be described in more detail as follows:

1. Take the whole data set and create  $m$  resampled data sets ( $m$  train/test pairs) using any of the approaches described above.
2. For each resampled train/test set ( $m$  pairs), and for each model ( $k$  models), obtain a validation error. This allows us to obtain an array of  $m \times k$  validation errors.
3. Determine the class ( $S_i$ ) with minimum estimated generalization error, that is the class with minimum validation error mean.

## 2.2 Multiple Model Selection Using Statistical Tests

The second step in the methodology consists on the selection of a set of models to be combined. The best conditions for combining occur when the learned models are accurate enough, but fairly independent in the errors they make. The first condition will be guaranteed by determining those models not significantly different from the model with minimum estimated generalization error using statistical tests for comparing  $k$  groups of related samples. The second condition is much more difficult to ensure, and is approximated considering different architectures, learning paradigms, model complexities, etc.

The proposed methodology determines a subset of models having *similar* error measures that the model with minimum estimated generalization error as follows:

1. Apply a medium power test (i.e. Nemenyi) to obtain the models which are not significantly different from the model with minimum estimated generalization error.
2. Apply an omnibus test for related samples (repeated measures ANOVA test, if the assumptions are met or Friedman test in different case).
  - 2.1. If the global null hypothesis is true (that is, all model classes of this set are not significantly different), finish the process.
  - 2.2. If the global null hypothesis is false, apply more powerful multiple comparison procedures (t or Wilcoxon paired tests with Bonferroni method for p-values adjustment) and obtain a subset with the model classes which are not significantly different from the model class with minimum estimated generalization error.

Some remarks about the method should be done. When omnibus tests are significant, it indicates that at least two of the model classes are significantly different, but we don't know which could be. At this point, multiple comparison procedures, which are usually less powerful, are applied.

Nemenyi test is a medium power multiple comparison procedure. It may even accept model classes that should be rejected. It is a good procedure to generate an initial but not definitive set of *non-significant* model classes.

Finally, the results may improve with a large number of resampled sets: resampling methods estimate better the generalization error and parametric tests [4], which are more powerful, may be applied on step 2. We suggest  $m \geq 30$ .

### 2.3 Model Combination Using Ensemble Methods

Model combination starts with the determination of a model for each near-optimal class as determined in the previous step of the methodology. For each class, we should select the member  $f_i(x, w^*)$  whose parameter vector  $w^*$  minimizes the empirical risk for the whole dataset.

Once a set of component models has been generated, they must be combined. This combination consists of a weighted combination of models. For combining the outputs of component models, the most prevailing approaches are majority weighted voting for classification tasks and weighted averaging for regression tasks [6],[13].

## 3 Experimental Results

In this section we shall describe the experiments carried out with our methodology, the obtained results, and a comparative study with other strategies. A number of simulations have been conducted to evaluate the efficiency of SEM method using Radial Basis Function networks (RBF). In our experiments, we have used several databases from the UCI repository [2], StatLib repository [12], Donoho-Johnstone benchmarks [15] and the ELENA Project [8] in order to test the performance of the method on regression and classification tasks using artificial and real databases.

We have repeatedly extracted (100 times) from each database a small number of examples (sample size column in tables 2 to 5) for model estimation, while the remaining ones were used to get a precise estimation of the expected generalization error for each trained model. For the block function, gaussian noise has been added to the outputs and the generalization error is estimated with 10000 previously unseen examples.

In order to compare the performance of different networks, we define the observed efficiency of model  $m_i$  as the ratio of the lowest estimated generalization error to that of model  $m_i$ . Thus, observed efficiencies range from 0 to 1. An observed efficiency equal to 1.0 would correspond to a model always having the lowest generalization error.

We have considered an initial set of RBF models with complexities ranging from 1 to  $n$  ( $n = 20$  or  $30$ , depending on the database) where  $n$  is defined as the number of kernels. The width of the basis functions has been set to

$$\sigma = \frac{\|\max(x_i - x_j)\|}{\sqrt{2n}}$$

All statistical tests have been applied using a level of significance  $\alpha = 5\%$ .

For SEM, we suggest three methods which require some restrictions on the weights (the weights must be greater than zero and sum to one) and fix the weights at the end of training. First, an unweighted average is computed (Basic Ensemble Method-BEM)[13]. Second, the weights are inversely proportional to estimated generalization error [11]. Finally, the weights are proportional to the number of times that each model has been selected as the model with minimum validation error [17]. A comparative of these methods is shown in [17]. In this paper, the simplest method (BEM) is applied.

Table 2 shows observed efficiency values for three regression tasks and for different sample sizes: Block function (25 y 50 data) , Abalone data set (50 and 100 data) and California housing (250 and 500 data). Statistical measures (mean, median and standard deviation) of the observed efficiency for the different methodologies are shown, as well as methodologies are ranked from the highest mean of the observed efficiency to the lowest mean (from 1 to 4 respectively). We have considered four different model building strategies: a) an ensemble using all the models, b) SEM using only Nemenyi test, c) SEM using Bonferoni test, and d) the model with the lowest estimated error.

Table 3 shows the number of component networks per meta-model on average for the different regression tasks considered.

**Table 2.** Observed efficiency for three regression tasks

| Database           | Sample size | Statistical measures | All models | SEM using Nemenyi | SEM using Bonferroni | Model selection |
|--------------------|-------------|----------------------|------------|-------------------|----------------------|-----------------|
| Block function     | 25          | Mean                 | 0.4019     | 0.8657            | 0.8308               | 0.7793          |
|                    |             | Median               | 0.2156     | 1.0000            | 0.9044               | 0.7670          |
|                    |             | Stand. dev.          | 0.4053     | 0.2554            | 0.2214               | 0.1436          |
|                    |             | Rank                 | 3          | 1                 | 2                    | 4               |
|                    | 50          | Mean                 | 0.7855     | 0.8955            | 0.8915               | 0.7260          |
|                    |             | Median               | 0.9629     | 0.9728            | 0.9555               | 0.7104          |
|                    |             | Stand. dev.          | 0.3110     | 0.2032            | 0.1869               | 0.1606          |
|                    |             | Rank                 | 3          | 1                 | 2                    | 4               |
| Abalone            | 50          | Mean                 | 0.9220     | 0.9789            | 0.9584               | 0.8638          |
|                    |             | Median               | 0.9290     | 0.9968            | 0.9746               | 0.8661          |
|                    |             | Stand. dev.          | 0.0702     | 0.0318            | 0.0486               | 0.1036          |
|                    |             | Rank                 | 3          | 1                 | 2                    | 4               |
|                    | 100         | Mean                 | 0.9797     | 0.9826            | 0.9727               | 0.9305          |
|                    |             | Median               | 0.9882     | 0.9882            | 0.9828               | 0.9503          |
|                    |             | Stand. dev.          | 0.0234     | 0.0192            | 0.0315               | 0.0703          |
|                    |             | Rank                 | 2          | 1                 | 3                    | 4               |
| California housing | 250         | Mean                 | 0.9815     | 0.9893            | 0.9804               | 0.9063          |
|                    |             | Median               | 0.9881     | 0.9978            | 0.9958               | 0.9111          |
|                    |             | Stand. dev.          | 0.0198     | 0.0167            | 0.0280               | 0.0582          |
|                    |             | Rank                 | 2          | 1                 | 3                    | 4               |
|                    | 500         | Mean                 | 0.9881     | 0.9970            | 0.9876               | 0.9152          |
|                    |             | Median               | 0.9887     | 1.0000            | 0.9939               | 0.9176          |
|                    |             | Stand. dev.          | 0.0086     | 0.0070            | 0.0213               | 0.0430          |
|                    |             | Rank                 | 2          | 1                 | 3                    | 4               |

**Table 3.** Average of the number of component networks per meta-model

| Database           | Sample size | All models | SEM using Nemenyi | SEM using Bonferroni | Model Selection |
|--------------------|-------------|------------|-------------------|----------------------|-----------------|
| Block function     | 25          | 30         | 7,37              | 4,08                 | 1               |
|                    | 50          | 30         | 13,87             | 8,67                 | 1               |
| Abalone            | 50          | 20         | 11,27             | 6,47                 | 1               |
|                    | 100         | 20         | 12,51             | 6,72                 | 1               |
| California housing | 250         | 20         | 15,12             | 8,07                 | 1               |
|                    | 500         | 20         | 17,35             | 11,26                | 1               |
|                    | 250         | 30         | 29,05             | 23,69                | 1               |

Table 4 shows results for 3 different classification tasks: Clouds data set with 50 and 250 experimental data, Gauss 2D with 50 and 100 experimental data and Phoneme data set with 100 and 250. Table 5 shows the number of component networks per meta-model on average.

**Table 4.** Observed efficiencies for three binary classification tasks

| Database | Sample size | Statistical measures | All models | SEM using Nemenyi | SEM using Bonferroni | Model selection |
|----------|-------------|----------------------|------------|-------------------|----------------------|-----------------|
| Clouds   | 50          | Mean                 | 0.9193     | 0.9587            | 0.9125               | 0.7475          |
|          |             | Median               | 1.0000     | 0.9778            | 0.9330               | 0.7416          |
|          |             | Stand. dev.          | 0.1384     | 0.0506            | 0.0871               | 0.1008          |
|          |             | Rank                 | 2          | 1                 | 3                    | 4               |
|          | 250         | Mean                 | 0.9156     | 0.9827            | 0.9955               | 0.9479          |
|          |             | Median               | 0.9182     | 0.9857            | 1.0000               | 0.9481          |
|          |             | Stand. dev.          | 0.0340     | 0.0168            | 0.0103               | 0.0393          |
|          |             | Rank                 | 3          | 2                 | 1                    | 4               |
| Gauss 2D | 50          | Mean                 | 0.7390     | 0.9624            | 0.9782               | 0.9395          |
|          |             | Median               | 0.7786     | 0.9772            | 1.0000               | 0.9618          |
|          |             | Stand. dev.          | 0.1446     | 0.0535            | 0.0464               | 0.0674          |
|          |             | Rank                 | 4          | 2                 | 1                    | 3               |
|          | 100         | Mean                 | 0.9155     | 0.9842            | 0.9913               | 0.9681          |
|          |             | Median               | 0.9265     | 0.9919            | 1.0000               | 0.9771          |
|          |             | Stand. dev.          | 0.0514     | 0.0210            | 0.0164               | 0.0332          |
|          |             | Rank                 | 4          | 2                 | 1                    | 3               |
| Phoneme  | 100         | Mean                 | 0.9918     | 0.9935            | 0.9813               | 0.9129          |
|          |             | Median               | 1.0000     | 1.0000            | 0.9955               | 0.9110          |
|          |             | Stand. dev.          | 0.0149     | 0.0116            | 0.0290               | 0.0460          |
|          |             | Rank                 | 1          | 2                 | 3                    | 4               |
|          | 250         | Mean                 | 0.9994     | 0.9986            | 0.9903               | 0.9014          |
|          |             | Median               | 1.0000     | 1.0000            | 0.9977               | 0.9083          |
|          |             | Stand. dev.          | 0.0017     | 0.0031            | 0.0154               | 0.0467          |
|          |             | Rank                 | 1          | 2                 | 3                    | 4               |

**Table 5.** Average of the number of component networks per meta-model

| Database | Sample size | All models | SEM using Nemenyi | SEM using Bonferroni | Model Selection |
|----------|-------------|------------|-------------------|----------------------|-----------------|
| Clouds   | 50          | 20         | 18,34             | 12,02                | 1               |
|          | 250         | 20         | 18,91             | 13,34                | 1               |
| Gauss 2D | 50          | 20         | 14,03             | 6,22                 | 1               |
|          | 100         | 20         | 14,83             | 6,85                 | 1               |
| Phoneme  | 100         | 30         | 25,96             | 18,72                | 1               |
|          | 250         | 30         | 29,05             | 23,69                | 1               |

Experimental results from the simulations (tables 2 and 4) suggest that generalization capability of SEM is higher (or similar in the worst case) than the model with minimum expected generalization error, and better than the ensemble obtained combining all component networks. Similar results are obtained applying only Nemenyi test or applying Bonferroni correction after it, but the ensembles generated after Bonferroni correction are less complex (tables 3 and 5). The SEM model selects the optimal cardinality for the ensemble and the appropriate complexity for their component networks.

## 4 Conclusions

It is known that combining networks improve the generalization ability. The number of component networks and their complexity are free parameters and usually must be fixed before the training process begins, but there is no standard procedure to fix these parameters. In this paper we have proposed a new ensemble method based on statistical techniques (SEM) which fixes these parameters in order to obtain a low generalization error with a small set of optimal component networks. Experimental results have shown that SEM improves the performance when compared to the strategy which selects the model with the lowest estimated generalization error and the strategy which combines all the networks.

Finally, other simulation results obtained applying our method [17] show that:

1. With other families of models (eg. linear models, polynomials, MLP networks,...), SEM always reduced the generalization error.
2. More powerful multiple comparison procedures based on Bonferroni correction [10] are not necessary, because a set of models with similar cardinality is selected.
3. Similar results are obtained applying random hold-out technique, but leave-one-out or k-fold cross-validation techniques make the results worse, because they select set of models with high cardinality.

## References

1. Bishop, C. M.: Neural network for pattern recognition. Clarendon Press-Oxford (1995)
2. Blake, C.L. y Merz, C.J. UCI Repository of machine learning databases <http://www.ics.uci.edu/~mlearn/MLRepository.html>. Irvine, CA: University of California, Department of Information and Computer Science (1998).
3. Dietterich, T. G. Machine Learning Research: Four Current Directions. *Artificial Intelligence Magazine*, 18(4), pp. 97-136 (1997).
4. Don Lehmkuhl, L.: Nonparametric statistics: methods for analyzing data not meeting assumptions required for the application of parametric tests. *Journal of prosthetics and orthotics* Vol. 8, num. 3, pp.105-113 (1996)
5. Guerrero, E., Yáñez, A., Galindo, P. and Pizarro, J. Repeated measures multiple comparison procedures applied to model selection in neural network. *Proceeding of the 6<sup>th</sup>. Int. Conf. on Artificial Neural Network (IWANN)*, vol. 2, pp. 88-95 (2001).
6. Hansen, L. K., Salamon, P. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10), pp. 993-1001 (1990).
7. Jacobs, R. A., Jordan, M. I., Nowlan, S. J. e Hinton, G. E. Adaptive mixtures of local experts. *Neural Computation*, 3 (1), pp. 79-87 (1991).
8. Jutten, C. et al. ESPIRIT basic research project number 689 ELENA. <ftp.dice.ucl.ac.be/pub/neural-net/ELENA/databases>
9. Krogh, A., Vedelsby, J. Neural networks ensembles, cross validation and active learning. In Tesauro, G., Touretzky, D. and Leen, T. (Eds.). *Advances in Neural Information Processing Systems*, vol. 7, pp. 231-238. The MIT Press (1995).
10. Lasarev, M. R.: Methods for p-value adjustment, Oregon Health & Science University, [http://medir.ohsu.edu/~geneview/education/dec19\\_h.pdf](http://medir.ohsu.edu/~geneview/education/dec19_h.pdf) (2001).
11. Optiz, D. W. y Shavlik, J. W. (1996). Generating accurate and diverse members of a neural-network ensemble. *Advances in Neural Information Processing Systems*, 8, págs. 535-541. Ed. D. S. Touretzky, M. C. Mozer y M. E. Hasselmo. The MIT Press.
12. Pace, R. K. y Barry, R. Sparse Spatial Autoregressions. *Statistics and Probability Letters*, 33, pp. 291-297. <http://lib.stat.cmu.edu/> (1997).
13. Perrone, M.P., Cooper, L.N. When networks disagree: ensemble method for neural networks, in: R.J.Mammone (Ed.), *Artificial Neural Networks for Speech and Vision*, Chapman & Hall, New York, pp.126-142 (1993).
14. Pizarro, J., Guerrero, E. and Galindo, P. Multiple comparison procedures applied to model selection. *Neurocomputing* 48, pp. 152-159 (2001).
15. Sarle, W. Donoho-Johnstone benchmarks: neural nets results. <ftp://ftp.sas.com/pub/neural/dojo/dojo.html> (1999).
16. Scharkey, A. J. C. On Combining Artificial Neural Nets. *Connection Science*, 8, 3/4, pp. 299-314 (1996).
17. Yáñez, A. Regresión mediante la combinación de modelos seleccionados mediante técnicas de remuestreo y procedimientos de comparación múltiple. Thesis. University of Cádiz.
18. Zar, J. H.: *Biostatistical analysis*, Prentice Hall (1996)

# Neural Network Modeling by Subsampling

Michele La Rocca and Cira Perna\*

Dept. of Economics and Statistics, University of Salerno,  
Via Ponte don Melillo, 84084 Fisciano (SA), Italy  
{larocca, perna}@unisa.it

**Abstract.** The aim of the paper is to develop hypothesis testing procedures both for variable selection and model adequacy to facilitate a model selection strategy for neural networks. The approach, based on statical inference tools, uses the subsampling to overcome the analytical and probabilistic difficulties related to the estimation of the sampling distribution of the test statistics involved. Some illustrative examples are also discussed.

## 1 Background and Motivations

Artificial neural networks are widely accepted as a potentially useful way of modeling complex non linear and dynamic systems. Their success is due to the great flexibility and capability of providing a model which fits any kind of data with an arbitrary degree of accuracy. This universal approximation property, together with the absence of the curse of dimensionality and the good predictive performances, has made the neural networks extensively used in a variety of statistical applications. A crucial point, when using a neural network model, is the choice of a proper topology which is basically related to the specification of the type and the number of the input variables and to the selection of the hidden layer size. The “atheoretical” nature of the tool, employed for the lack of knowledge about the functional form of the data generating process and the intrinsic misspecification of the model, make the choice of an adequate neural network an hard task. The most used approaches to this problem are based on pruning and regularization, information criteria and cross-validation methods [10]. Although these techniques may lead to satisfactory results, they lack of an inferential statistical perspective and usually contain a strong judgemental component which makes the model building process difficult to reconstruct.

To circumvent these drawbacks, alternative procedures based on hypothesis testing have been proposed (see [3],[1] inter alia). These techniques are based on a strong statistical approach but they focus on single weights and this could be misleading due to the black-box nature of the neural network model. Indeed, they do not give any information on the most “significant” variables, which is useful in any model building strategy. Moreover, different topologies can achieve the

---

\* We gratefully acknowledge the comments of two anonymous referees.

same approximation accuracy. As a consequence, a proper choice of the network topology cannot be just based on complexity reasons and it should also take into account model plausibility.

Therefore, a model selection strategy should emphasize the role of the explanatory variables (useful for the identification and interpretation of the model) and it should treat the hidden layer size as a smoothing parameter, taking into account the trade-off between estimation bias and variability. The solutions proposed in the statistical literature, following this spirit, for model identification, diagnostic testing and model adequacy, basically deal with *iid* case and so they are not suitable for time series data [11], [13].

The aim of the paper is to develop hypothesis testing procedures, both for variable selection and model adequacy, to facilitate neural network model selection strategies based on statical inference tools in the dependent data framework. The approach uses the subsampling which gives consistent results under quite general and weak assumptions and allows to overcome the analytical and probabilistic difficulties related to the estimation of the sampling distribution of the test statistics [9]. Basically the scheme just requires the existence of a limiting law for the sampling distribution of the statistics involved. It does not require any knowledge of the specific structures of the time series other than its asymptotic stationarity and strong mixing properties.

The paper is organized as follows. Section 2 describes the data generating process and the neural network model employed. Section 3 discusses the use of hypothesis testing procedures in neural network modeling focusing on the use of subsampling. Finally, section 4 reports the results of two illustrative examples along with some concluding remarks.

## 2 The Data Generating Process and the Neural Network Model

Let  $\{Y_t, t \in \mathbb{Z}\}$  be a process modeled as:

$$Y_t = g(\mathbf{X}_t) + \varepsilon_t \quad (1)$$

where  $\{Y_t, \mathbf{X}'_t\}$  is a stationary,  $\alpha$ -mixing sequence and  $\mathbf{X}_t = (X_{1t}, \dots, X_{dt})'$  is a vector of  $d$  random variables possibly including explanatory variables, lagged explanatory variables and lagged values of  $Y_t$ . The unknown function  $g(\cdot)$  is a continuously differentiable function defined on a compact subset of  $\mathbb{R}^d$ .

The function  $g$  can be approximated by a single hidden layer feed-forward neural network  $NN(d, r)$  defined as

$$f(\mathbf{x}_t, \theta) = \sum_{k=1}^r c_k \phi \left( \sum_{j=1}^d a_{kj} x_{jt} + a_k \right) + c_0 \quad (2)$$

where  $\mathbf{x} = (x_1, \dots, x_d)$  is the vector of the  $d$  input variables,  $a_{kj}$  is the weight of the connection between the  $j$ -th input neuron and the  $k$ -th neuron in the



hidden level;  $c_k, k = 1, \dots, r$  is the weight of the link between the  $k$ -th neuron in the hidden layer and the output;  $a_{k0}$  and  $c_0$  are respectively the bias term of the hidden neurons and of the output;  $\phi(\cdot)$  is the activation function of the hidden layer. We define  $\theta = (c_0, c_1, \dots, c_r, \mathbf{a}'_1, \mathbf{a}'_2, \dots, \mathbf{a}'_r)'$  where  $\mathbf{a}'_i = (a_{i0}, a_{i1}, \dots, a_{id})$  with  $\theta \in \Theta \subset \mathbb{R}^{r(d+2)+1}$ .

A well known result [2] guarantees that, under quite general conditions, there exists a parameter vector  $\theta^*$  such that

$$\|g(\mathbf{x}) - f(\mathbf{x}, \theta^*)\| \leq \frac{(2C_g)^2}{r} \tag{3}$$

where  $C_g > 0$  is a proper chosen constant.

Once fixed the network topology, the parameter vector  $\theta^*$  can be estimated by solving

$$T^{-1} \sum_{t=1}^T \psi(\mathbf{Z}_t, \theta) = 0. \tag{4}$$

where the function  $\psi(\cdot)$  can generate different classes of estimators such as least squares, maximum likelihood and generalized method of moments.

The universal approximation property and the wide choice of algorithms available for the estimation procedure have made the neural networks widely used in a variety of statistical applications. Unfortunately, this class of models is not yet supported by the rich collection of specification and diagnostic tests usually employed in statistical and econometric modeling.

A crucial point, when using a  $NN(d, r)$  model, is the choice of a proper topology which, basically, is related to the specification of the type and the number of the input variables and to the selection of the hidden layer size.

The problem is not a novel one and a number of different and effective solutions have been proposed in the machine learning framework (see [5]). Here, we focus on an econometric like approach. In this perspective, a neural network model selection strategy should emphasize the different role of the input and the hidden layer neurons. In a regression framework, input neurons are related to the explanatory variables (useful for identification and interpretation of the model) while the hidden layer size has no clear interpretation and it should be considered basically as a smoothing parameter, taking into account the trade-off between estimation bias and variability.

Therefore, the input selection step should be addressed by referring to statistical test procedures for variable selection in dynamic regression models. While the hidden layer size could be chosen by considering one of the many results available in the statistical literature, ranging from the information criteria based on fitting to the indexes based on prediction accuracy. In this framework, a model is selected if it has the smallest value of the chosen index within a class of models. If two models have the same value of the index, the simpler one is selected. In any case, the criteria based on the traditional maximum likelihood approach have not been clarified completely (see [4] for a theoretical discussion).

Moreover, often the differences among indexes are not so large and so a test procedure for the equivalence of two competing models could be useful.

### 3 Hypothesis Testing in Neural Networks by Subsampling

To select a proper set of input variables, we focus on a stepwise selection rule which involves: (i) definition of variable's relevance to the model; (ii) estimation of the sampling distribution of the relevance measure; (iii) testing the hypothesis that the variable is irrelevant [11].

The hypothesis that a set of independent variables  $\mathcal{X}_0 = \{x_i, i \in I_0\}$  has no effect on  $Y$  can be formulated as [13],

$$f_i(x; \theta^*) = \frac{\partial f(x; \theta^*)}{\partial x_i} = 0, \forall x, i \in I_0. \quad (5)$$

and tested by referring to the null

$$H_0 : m^* = \sum_{i \in I_0} E[f_i^2(\mathbf{X}_t, \theta^*)] = 0 \quad (6)$$

The null  $H_0$  can be tested by using the statistic,

$$\hat{m}_T = T^{-1} \sum_{i \in I_0} \sum_{t=1}^T f_i^2(\mathbf{X}_t, \hat{\theta}_T) \quad (7)$$

where the parameter vector  $\hat{\theta}_T$  is a consistent estimator of the unknown parameter vector  $\theta^*$ .

The distribution of the test statistic under: (i) stationarity, mixing and moment condition on the data generating process, (ii) smooth condition on the function  $\psi(\cdot)$  involved in the estimation step, (iii) sigmoidal activation functions, can be consistently approximated by using the subsampling [6].

The choice of this resampling technique is related to the probabilistic complexity of neural network modeling for dependent data. The subsampling method estimates the sampling distribution of the test statistic by evaluating it on blocks of consecutive observations in order to take into account the dependence structure of the observed time series. This basic resampling scheme does not change dramatically when moving from *iid* to dependent data. Moreover, the method does not require any knowledge of the specific structures of the time series other than its stationarity and strong mixing property, so it is robust against misspecified models, a key property when dealing with artificial neural network models which are intrinsically misspecified. Finally, the subsampling is by no means restricted to stationary series but it gives asymptotically correct inference even for heteroskedastic time series under general minimal assumptions [8]. Basically the scheme requires that  $\frac{\delta_b}{\delta_T} \rightarrow 0$ ,  $\frac{b}{T} \rightarrow 0$  and  $b \rightarrow \infty$  as  $T \rightarrow \infty$  and the existence of a limiting law for the sampling distribution of the test statistic [9].

The main issue when applying the subsampling procedure lies in choosing the length of the block, a problem which is common to all blockwise resampling techniques. Even if the conditions on  $b$  are quite weak, they do not give any guidelines for its choice and this parameter, which is related to the amount of dependence assumed in the series, has to be chosen on the data at hand. Nevertheless, Politis et al. [9] proposed a number of strategies to select  $b$  and theorems that ensure that the asymptotic results are still valid for a broad range of choices for the subsample size.

A somewhat similar test procedure, based on the comparison of competing neural network models, can be used to verify if there are any omitted variables or if a different hidden layer size can improve the performance of the model [13].

Let  $f_1(x, \theta_1^*)$  and  $f_2(x, \theta_2^*)$  be two nested competing neural network models. The idea is that if the two networks are equivalent  $f_1$  is capable of producing an output identical to that of the network  $f_2$ . A discrepancy measure between the outputs of the two competing neural network models can be defined as

$$m^* = E \left[ (f_1(x, \theta_1^*) - f_2(x, \theta_2^*))^2 \right] \quad (8)$$

Therefore, the hypothesis that the two models are equivalent can be written as  $H_0: m^* = 0$  and tested by using the statistic

$$\hat{m}_T = \frac{1}{T} \sum_{t=1}^T \left( f_1(\mathbf{X}_t, \hat{\theta}_{T1}) - f_2(\mathbf{X}_t, \hat{\theta}_{T2}) \right)^2 \quad (9)$$

where  $\hat{\theta}_{T1}$  and  $\hat{\theta}_{T2}$  are consistent estimators of, respectively,  $\theta_1^*$  and  $\theta_2^*$ .

Again, the distribution of the test statistic can be consistently estimated by using the subsampling, thus extending straightforwardly the procedure to dependent data.

## 4 Illustrative Examples and Concluding Remarks

To illustrate how the proposed test procedures work, the results of some illustrative examples on simulated data will be reported. The experimental setup is based on datasets generated by the following models

$$\begin{aligned} \text{(M1)} \quad Y_t &= -0.3 - 0.5Y_{t-1}(1 - \mathcal{G}(Y_{t-1})) + 0.1 + 0.5Y_{t-1}\mathcal{G}(Y_{t-1}) + \varepsilon_t \\ \text{(M2)} \quad Y_t &= (0.5 + 0.9 \exp(-Y_{t-1}^2)) Y_{t-1} + (0.8 - 1.8 \exp(-Y_{t-1}^2)) Y_{t-2} + \varepsilon_t \end{aligned}$$

where the innovations  $\varepsilon_t$  are distributed as standard normal,  $\mathcal{G}(\cdot)$  is the logistic cumulative function. These models, very popular in several applications in different fields, have been chosen as representing different kind of dynamic behaviour since their flexibility allows generation of quite different time series structures. Model (M1) is a Logistic Smooth Transition Model (LSTAR) while M2 is an Exponential Autoregressive model of order 2 used in several studies on the bootstrap for its peculiar nonlinear structure.

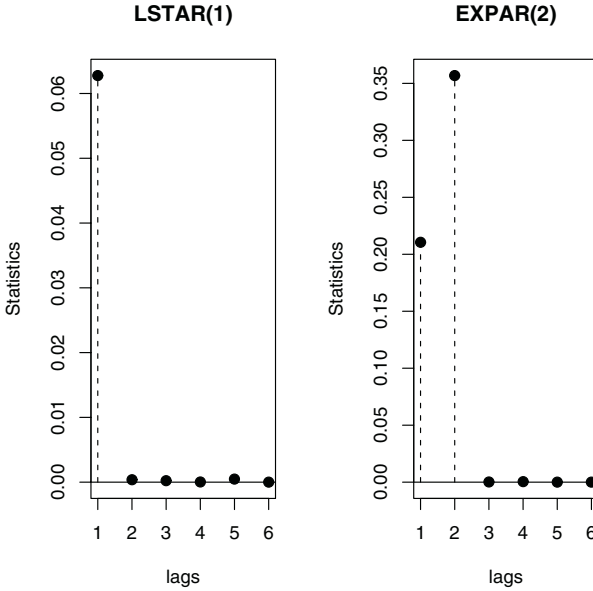


Fig. 1. Plots of the values of the relevance measure for different lags

Table 1. Values of the test statistics and  $p$ -values (estimated by subsampling) in parenthesis for the selection of the relevant input set

| Model            | $I_0 = \{1\}$     | $I_0 = \{2\}$     | $I_0 = \{3, 4, 5, 6\}$ |
|------------------|-------------------|-------------------|------------------------|
| <i>LSTAR</i> (1) | 64.6314 (0.0009)  | 0.1982 (0.7702)   | 0.7466 (0.9589)        |
| <i>EXPAR</i> (2) | 212.1005 (0.0000) | 359.6342 (0.0000) | 0.5749 (0.9378)        |

The nonlinear dynamical structure of the data can be modeled by approximating the DGP by a proper chosen neural network model  $NN(d, r)$ .

To select the set of variables to be tested as irrelevant, we use simple graphical exploratory tools based on the plots of the relevance measures for each single lag. Values of a given relevance measure close to zero candidate the corresponding lag to be in the set of irrelevant ones. In this exercise we start with a tentative model  $NN(6, 1)$ . By the plots in Figure 1 we identify, as possible relevant variables lag 1 for the *LSTAR* model and lags 1 and 2 for the *EXPAR* model. The results of the test based identification procedure are reported in Table 1 where the  $p$ -values have been estimated by subsampling with the subseries length  $b = 180$  identified by using the Variance Inflation Index [9]. Clearly, as expected, we identify a neural network model with only one input neuron (corresponding to the first lag) for the M1 model and a neural network model with two input neurons (corresponding to the first two lags) for the model M2.

Following a common practice and for computational reasons, the hidden layer size has been selected by using the Akaike (AIC), the Hannan-Quinn (HQIC) and the Schwartz (SIC) information criteria (see Table 2).

**Table 2.** Values of the information criteria to compare networks with different hidden layer size. In bold the minimum value for each index. Input set chosen according to Table 1

| Model            | $r$ | AIC            | HQIC           | SIC            |
|------------------|-----|----------------|----------------|----------------|
| <i>LSTAR</i> (1) | 1   | -3.6369        | -3.6393        | -3.6194        |
|                  | 2   | <b>-3.6825</b> | <b>-3.6828</b> | <b>-3.6482</b> |
|                  | 3   | -3.6766        | -3.6772        | -3.6274        |
|                  | 4   | -3.6705        | -3.6714        | -3.6067        |
|                  | 5   | -3.6645        | -3.6656        | -3.5860        |
| <i>EXPAR</i> (2) | 1   | -4.8626        | -4.8629        | -4.8380        |
|                  | 2   | -4.9617        | -4.9623        | -4.9175        |
|                  | 3   | -5.1006        | -5.1014        | <b>-5.0367</b> |
|                  | 4   | <b>-5.1192</b> | <b>-5.1203</b> | -5.0356        |
|                  | 5   | -5.1142        | -5.1157        | -5.0110        |

**Table 3.** Values of the test statistics and  $p$ -values (estimated by subsampling) in parenthesis to compare different models. Input set chosen according to Table 1

| Model            | $r = 1 \leftrightarrow r = 2$ | $r = 2 \leftrightarrow r = 3$ | $r = 3 \leftrightarrow r = 4$ | $r = 4 \leftrightarrow r = 5$ |
|------------------|-------------------------------|-------------------------------|-------------------------------|-------------------------------|
| <i>LSTAR</i> (1) | 1.1736 (0.0000)               | 0.0017 (0.1221)               | 0.0051 (0.0888)               | < 0.0001 (0.1953)             |
| <i>EXPAR</i> (2) | 0.1750 (0.0000)               | 1.9943 (0.0000)               | 0.0814 (0.0056)               | 0.0194 (0.1210)               |

**Table 4.** Teraesvirta and White neural network tests for neglected nonlinearity, Jarque and Bera Test for normality, on the original data and on the residuals from the “optimal” estimated neural network models

| Model            | Test               | ORIGINAL DATA |            | RESIDUALS |            |
|------------------|--------------------|---------------|------------|-----------|------------|
|                  |                    | Statistic     | $p$ -value | Statistic | $p$ -value |
| <i>LSTAR</i> (1) | <i>Teraesvirta</i> | 80.8153       | 0.0000     | 0.3757    | 0.8287     |
|                  | <i>White</i>       | 80.8030       | 0.0000     | 0.2578    | 0.8790     |
|                  | <i>Jarque-Bera</i> | 0.1078        | 0.9475     | 0.2623    | 0.8771     |
| <i>EXPAR</i> (2) | <i>Teraesvirta</i> | 109.8451      | 0.0000     | 4.8964    | 0.6726     |
|                  | <i>White</i>       | 14.2297       | 0.0000     | 0.7716    | 0.6799     |
|                  | <i>Jarque-Bera</i> | 124.2957      | 0.0000     | 2.5579    | 0.2783     |

The values of the indexes point towards a network with two hidden neurons for the *LSTAR* model and 3 or possibly 4 hidden neurons for the *EXPAR* model. In Table 3 we reported the results of the test procedures comparing competing models with different hidden layer sizes. Again the  $p$ -values have been computed by using the subsampling. The input neurons have been fixed according to the results of Table 1.

Clearly, for the *LSTAR* model neural networks with more than two neurons in the hidden layer seem to be equivalent to the  $NN(1, 2)$  model. So, the “optimal” model seems to be the  $NN(1, 2)$ . For the *EXPAR* case, the test procedure suggests that the  $NN(2, 3)$  and the  $NN(2, 4)$  model cannot be considered equivalent, while there is no improvement in the performances by considering more

than four neurons in the hidden layer. So the “optimal” identified model seems to be  $NN(2, 4)$ .

The neural network tests for neglected nonlinearity by Teraesvirta [12] and White [7] on the residuals from the identified optimal models, along with the Jarque-Bera test for normality, are reported in Table 4. The tests on the residuals do not refuse the null and so the nonlinear structure of the data seems to be correctly modeled. Moreover, the residuals can be considered as realization of a gaussian process.

The proposed test procedures based on the subsampling seem able to detect correctly the set of input variables and to discriminate among alternative models. Clearly, joint usage of neural network models and subsampling is usually quite demanding from a computational point of view. In any case, it is worthwhile to underline that both neural networks and subsampling are suitable to be implemented on parallel and cluster computers almost without any modification of the computing algorithms.

## References

1. Anders, U., Korn, O.: Model selection in neural networks. *Neural Networks*. **12** (1999) 309–323
2. Barron, A. R.: Universal approximation bounds for superposition of a sigmoidal function. *IEEE Transactions on Information Theory* **39** (1993) 930–945
3. Cottrell, M., Girard, B., Girard, Y., Mangeas, M., Muller, C.: Neural modeling for time series: a statistical stepwise method for weight elimination. *IEEE Transaction on Neural Networks*. **6** (1995) 1355 – 1364
4. Fukumizu, K.: Likelihood Ratio of Unidentifiable Models and Multilayer Neural Networks. *The Annals of Statistics*. **31** (2003) 833 – 851
5. Hastie, T., Tibshirani, R., Friedman, J.: *The Elements of Statistical Learning*. Springer.
6. La Rocca, M, Perna C.: Variable selection in neural network regression models with dependent data: a subsampling approach. *Computational Statistics and Data Analysis*. **48** (2005) 415 – 419
7. Lee, T. H., White, H., Granger C. W. J.: Testing for neglected nonlinearity in time series models. *Journal of Econometrics*. **56** (1993) 269–290
8. Politis, D. N., Romano, J. P., Wolf, M.: Subsampling for heteroscedastic time series. *Journal of Econometrics*. **81** (1997) 281–317
9. Politis, D. N., Romano, J. P., Wolf, M.: *Subsampling*. Springer NY (1999)
10. Reed, R.: Pruning algorithms—a survey. *Neural Networks*. **4** (1993) 740–747
11. Refenes, A. P. N., Zapranis, and A. D.: Neural model identification, variable selection and model adequacy. *Journal of Forecasting*. **18** (1999) 299–332
12. Teraesvirta, T., Lin, C. F., Granger, C. W. J.: Power of the Neural Network Linearity Test. *Journal of Time Series Analysis*. **14** (1993) 209–220
13. White, H., Racine, J.: Statistical Inference, The Bootstrap, and Neural-Network Modeling with Application to Foreign Exchange Rates. *IEEE Transactions on Neural Networks*. **12** (2001) 657–673

# Balanced Boosting with Parallel Perceptrons

Iván Cantador and José R. Dorronsoro\*

Dpto. de Ingeniería Informática and Instituto de Ingeniería del Conocimiento,  
Universidad Autónoma de Madrid, 28049 Madrid, Spain

**Abstract.** Boosting constructs a weighted classifier out of possibly weak learners by successively concentrating on those patterns harder to classify. While giving excellent results in many problems, its performance can deteriorate in the presence of patterns with incorrect labels. In this work we shall use parallel perceptrons (PP), a novel approach to the classical committee machines, to detect whether a pattern's label may not be correct and also whether it is redundant in the sense of being well represented in the training sample by many other similar patterns. Among other things, PP allow to naturally define margins for hidden unit activations, that we shall use to define the above pattern types. This pattern type classification allows a more nuanced approach to boosting. In particular, the procedure we shall propose, balanced boosting, uses it to modify boosting distribution updates. As we shall illustrate numerically, balanced boosting gives very good results on relatively hard classification problems, particularly in some that present a marked imbalance between class sizes.

## 1 Introduction

As it is well known, boosting constructs a weighted classifier out of possibly weak learners by successively concentrating on those patterns harder to classify. More precisely, it keeps on each iteration a distribution  $d_t(X)$  of the underlying  $X$  patterns, and after a new hypothesis  $h_t$  has been constructed in the  $t$ -th iteration,  $d_t(X)$  is updated to

$$d_{t+1}(X) = \frac{1}{Z_t} d_t(X) e^{-\alpha_t y_X h_t(X)}, \quad (1)$$

where  $y_X = \pm 1$  is the class label associated to  $X$ ,  $Z_t$  is a probability normalization constant and  $\alpha_t$  is related to the training error  $\epsilon_t$  of  $h_t$  (more details in the third section). Therefore, after each iteration boosting concentrates on the patterns harder to classify, as we have  $e^{-\alpha_t y_X h_t(X)} > 1$  if  $y_X h_t(X) < 0$ , i.e.,  $X$  has been incorrectly classified; as a consequence, the training error  $\epsilon_t$  will tend to 0 under mild hypothesis on the weak learner [6]. The final hypothesis is the average  $h(X) = \sum_t \alpha_t h_t(X)$  of the successively built weak hypotheses  $h_t$ .

Boosting has been used with great success in several applications and over various data sets [2]. However, its has also been shown that it may not yield

---

\* With partial support of Spain's CICYT, TIC 01-572.

such good results when applied to noisy datasets. In fact, assume that a given pattern has label noise, that is, although clearly being a member of one class, its label corresponds to the alternate class. Such a label noisy pattern is likely to be repeatedly misclassified by the successive hypotheses which, in turn, would increase its sampling probability and cause boosting to hopelessly concentrate on it. Although this fact may be useful in some instances, its most likely consequence is to deteriorate the final hypothesis. The just described situation may very well happen when dealing with imbalanced data sets, where the number of patterns from one class (that we term the positive one) is much smaller than that from others. There are many examples of this situation, as well as a large literature on this topic, with many techniques having been applied [3, 7]. Most real world classification problems involve imbalanced samples and for them we should expect patterns to fall within three categories: redundant (i.e., easy to classify and likely to be overrepresented in the sample), the just described label noisy and, finally, borderline patterns, i.e., those whose classification could be different after small perturbations and upon which classifier construction should concentrate. To successfully deal with imbalanced data sets it is quite important to detect and handle these three pattern categories correctly.

In this work we introduce a new technique for redundant, label noisy and borderline pattern detection that, in turn, will suggest a new procedure for boosting's probability update (1) depending on what pattern category  $X$  is in. The assignment of  $X$  to one of these types is based on another concept of margin that arises naturally in the training of parallel perceptrons (PP), a type of committee machines introduced by Auer et al. in [1] and that will be described in section 2. A key part of the PP training procedure is an output stabilization technique that tries to augment the distance of the activation of a perceptron to its decision hyperplane, i.e., its activation margin, so that small random changes on an input pattern do not cause its being assigned to another class. The activation margins are also learned in some sense during training and can be used for the above classification of training patterns, as it will be described in section 3. In turn, knowing which kind of pattern a given one is can be used to adjust boosting's probability updates. We will do so here by changing the exponent in (1) to  $\alpha_t R(X) y_X h_t(X)$ , where the  $R(X)$  factor will reflect the nature of the pattern  $X$ . More precisely,  $R(X)$  will be 1 for redundant patterns and  $-1$  for noisy ones. We shall consider in section 3 several options for choosing  $R(X)$  for borderline patterns; as we shall see in section 4, best results will be obtained by what we shall call balanced boosting, whose results are comparable to those of boosted multilayer perceptrons (MLPs) but with much smaller training times. Finally, the paper will close with a brief summary section and a discussion of further work.

## 2 Parallel Perceptron Training

PPs have the same structure of the well known committee machines [5], that is, they are made up of an odd number of standard perceptrons  $P_i$  with  $\pm 1$  outputs,



and the machine's one dimensional output is simply the sign of the sum of these perceptrons' outputs (that is, the sign of the overall perceptron vote count). They are thus well suited for 2-class discrimination problems, but it is shown in [1] that they can also be used in regression problems. In more detail, assume we are working with  $D$  dimensional patterns  $X = (x_1, \dots, x_D)^t$ , where the  $D$ -th entry has a fixed 1 value to include bias effects. If the committee machine (CM) has  $H$  perceptrons, each with a weight vector  $W_i$ , for a given input  $X$ , the output of perceptron  $i$  is then  $P_i(X) = s(W_i \cdot X) = s(\text{act}_i(X))$ , where  $s(\cdot)$  denotes the sign function and  $\text{act}_i(X) = W_i \cdot X$  is the activation of perceptron  $i$  due to  $X$ . The final output  $h(X)$  of the CM is  $h(X) = s\left(\sum_1^H P_i(X)\right)$  where we take  $H$  to be odd to avoid ties. We will assume that each input  $X$  has an associated  $\pm 1$  label  $y_X$  and take the output  $h(X)$  as correct if  $y_X h(X) > 0$ . If this is not the case, i.e. whenever  $y_X h(X) = -1$ , parallel perceptron training applies the well known Rosenblatt's rule

$$W_i := W_i + \eta y_X X. \quad (2)$$

to all wrong perceptrons, i.e. those  $P_i$  verifying  $y_X P_i(X) = -1$  ( $\eta$  denotes a possibly varying learning rate). Moreover, when a pattern  $X$  is correctly classified, PP training also applies a margin-based output stabilization procedure to those perceptrons for which  $0 < y_X \text{act}_i(X) < \gamma$ . Notice that for them a small perturbation could cause a wrong class assignment.

The value of the margin  $\gamma$  is also adjusted dynamically from a starting value. More precisely, as proposed in [1], after a pattern  $X$  is processed correctly,  $\gamma$  is increased to  $\gamma + 0.25\eta$  if for all correct perceptrons we have  $y_X \text{act}_i(X) > \gamma$ , while we decrease  $\gamma$  to  $\gamma - 0.75\eta$  if  $0 < y_X \text{act}_i(X) < \gamma$  for at least one correct perceptron. PPs can be trained either on line or in batch mode; since we will use then in a boosting framework, we shall use this second procedure. Notice that for the margin to be meaningful, weights have to be normalized somehow; we will make its euclidean norm to be 1 after each batch pass. In spite of their very simple structure, PPs do have a universal approximation property. Moreover, as shown in [1], PPs provide results in classification and regression problems quite close to those offered by C4.5 decision trees and only slightly weaker than those of standard multilayer perceptrons (MLPs). Finally, their training is extremely fast, specially when compared to that of MLPs, something quite useful in boosting, where repeated batch trainings will have to be performed.

### 3 Boosting Parallel Perceptrons

As mentioned in the introduction, boosting constructs after each iteration a weak hypothesis  $h_t$  over the current distribution  $d_t$ , and updates it according to the rule (1), in which  $Z_t = \sum_X d_{t+1}(X)$  is a probability normalization,  $\alpha_t = \ln((1 - \epsilon_t)/\epsilon_t)/2$ , and  $\epsilon_t$  is the iteration error with respect to  $d_t$ , i.e.,

$$\epsilon_t = \sum_{\{X : y_X h_t(X) = -1\}} d_t(X).$$

**Table 1.** The table gives the  $R(X)$  labels for training patterns for negative, positive and balanced boosting. All tend to avoid label noisy patterns and their main difference is in the handling of near noisy borderline patterns. Standard boosting sets  $R(X) = 1$  in all cases

| Pattern set | neg. boostPP | pos. boostPP | bal. boostPP |
|-------------|--------------|--------------|--------------|
| $R$         | 1            | 1            | 1            |
| $N$         | -1           | -1           | -1           |
| $nB^-$      | 1            | -1           | 0            |
| other $B$   | 1            | 1            | 1            |

As mentioned in the introduction, boosting may not yield good results when applied to noisy datasets, as these will be repeatedly misclassified by the successive hypotheses, increasing their sampling probability and causing boosting to hopelessly concentrate on them. In it. On the other hand, PP's activation margins can be used to detect not only label noisy patterns but also those that are redundant and borderline. In more detail, PPs adaptively adjust these margins, making them to converge to a final value  $\gamma$ . If for a pattern  $X$  its  $i$ -th perceptron activation verifies  $|act_i(X)| > \gamma$ ,  $s(act_i(X))$  is likely to remain unchanged after small perturbations of  $X$ . Thus if for all  $i$  we have  $y_X act_i(X) > \gamma$ ,  $X$  is likely to be also correctly classified later on. Those patterns are natural choices to be taken as redundant. Similarly, if for all  $i$  we have  $y_X act_i(X) < -\gamma$ ,  $X$  is likely to remain wrongly classified, and we will take such patterns as label noisy. The remaining  $X$  will be the borderline patterns. We shall use the notations  $R_t$ ,  $N_t$  and  $B_t$  for the redundant, noisy and borderline training sets at iteration  $t$ . To take into account this categorization, we may introduce a pattern dependent factor  $R(X)$  in the boosting probability actualization procedure as follows

$$d_{t+1}(X) = \frac{1}{Z'_t} d_t(X) e^{-\alpha_t R(X) y_X h_t(X)},$$

with  $Z'_t$  again a normalization constant. If we set the factor  $R(X)$  to be 1, we just recapture standard boosting, while if we want to diminish the influence of label noisy patterns  $X \in N_t$ , we put  $R(X) = -1$ ; since they are not correctly classified, then  $\alpha_t R(X) y_X h_t(X) > 0$  and hence,  $d_{t+1}(X) < d_t(X)$ . Moreover, we would like to keep boosting focused on borderline patterns, even if they are temporarily misclassified. To do so, we have several options. First we can just proceed as in standard boosting, setting  $R(X) = 1$  when  $X \in B$ ; for borderline patterns incorrectly classified this will augment their subsequent probability, while it will diminish it for those well classified. Notice that if the latter are close to the separating hyperplane, they may not be correctly classified afterwards, causing boosting to refocus on them.

However, when dealing with unbalanced datasets, accuracy, that is the percentage of correctly classified patterns, may not be a relevant criterium, as it would be fulfilled by the simple procedure of assigning all patterns to the (possibly much larger) negative class. It may thus be convenient to lessen the impact

**Table 2.** Accuracies and  $g$  values for the standard boosting procedures over 7 UCI datasets using MLPs and PPs as learning algorithms. The more complex structure of MLPs gives better accuracies, although those of PPs are quite close in all problems except two

|            |             | std. boostMLP   |       |       |                | std. boostPP    |       |       |                |
|------------|-------------|-----------------|-------|-------|----------------|-----------------|-------|-------|----------------|
| Dataset    | % positives | $a$             | $a^+$ | $a^-$ | $g$            | $a$             | $a^+$ | $a^-$ | $g$            |
| Ionosphere | 35.9        | 88.40<br>(0.32) | 73.53 | 96.71 | 84.3<br>(1.89) | 85.49<br>(1.51) | 68.91 | 94.85 | 80.8<br>(2.02) |
| Diabetes   | 34.9        | 73.80<br>(0.25) | 60.63 | 80.84 | 70.0<br>(1.75) | 73.16<br>(0.97) | 57.99 | 81.42 | 68.7<br>(1.06) |
| Cancer     | 34.5        | 95.82<br>(0.13) | 94.19 | 96.68 | 95.4<br>(0.55) | 95.64<br>(0.25) | 93.32 | 96.86 | 95.1<br>(0.52) |
| Vehicle    | 25.7        | 83.16<br>(0.10) | 65.35 | 89.30 | 76.4<br>(0.60) | 79.04<br>(1.15) | 55.33 | 87.30 | 69.5<br>(1.05) |
| Glass      | 13.6        | 95.82<br>(0.22) | 88.00 | 97.01 | 92.4<br>(1.52) | 95.71<br>(1.06) | 86.83 | 97.13 | 91.8<br>(1.84) |
| Vowel      | 9.1         | 99.66<br>(0.01) | 98.22 | 99.80 | 99.0<br>(0.12) | 99.46<br>(0.28) | 96.56 | 99.76 | 98.1<br>(0.80) |
| Thyroid    | 7.4         | 98.73<br>(0.05) | 91.30 | 99.32 | 95.2<br>(0.32) | 97.33<br>(0.19) | 80.14 | 98.85 | 89.0<br>(1.20) |

**Table 3.** Accuracies for the boosting procedures over 7 UCI datasets (the lower values give the standard deviations of 10 times 10-fold cross validation); best values in bold face. While it only gives the best result in the glass problem, the overall accuracy of balanced boost is quite close to the best one, while giving a good balance between  $a^+$  and  $a^-$

|            | std. boostPP           |       |       | neg. boostPP           |       |       | pos. boostPP           |       |       | bal. boostPP           |       |       |
|------------|------------------------|-------|-------|------------------------|-------|-------|------------------------|-------|-------|------------------------|-------|-------|
| Dataset    | $a$                    | $a^+$ | $a^-$ | $a$                    | $a^+$ | $a^-$ | $a$                    | $a^+$ | $a^-$ | $a$                    | $a^+$ | $a^-$ |
| Ionosphere | <b>85.49</b><br>(1.51) | 68.91 | 94.85 | 85.09<br>(1.21)        | 67.45 | 95.09 | 65.97<br>(1.40)        | 89.85 | 52.50 | 84.97<br>(1.20)        | 71.44 | 92.65 |
| Diabetes   | 73.16<br>(0.97)        | 57.99 | 81.42 | <b>73.29</b><br>(0.88) | 59.07 | 81.04 | 57.76<br>(0.96)        | 94.85 | 37.56 | 72.08<br>(0.88)        | 73.18 | 71.49 |
| Cancer     | 95.64<br>(0.25)        | 93.32 | 96.86 | 96.03<br>(0.18)        | 94.27 | 97.02 | <b>96.39</b><br>(0.22) | 99.15 | 94.91 | 96.32<br>(0.16)        | 96.07 | 96.50 |
| Vehicle    | 79.04<br>(1.15)        | 55.33 | 87.30 | <b>79.68</b><br>(0.97) | 55.04 | 88.27 | 72.09<br>(0.46)        | 95.52 | 63.93 | 78.61<br>(0.75)        | 72.41 | 80.78 |
| Glass      | 95.71<br>(1.06)        | 86.83 | 97.13 | 96.09<br>(0.78)        | 87.33 | 97.51 | 94.76<br>(0.39)        | 90.17 | 95.53 | <b>96.33</b><br>(0.88) | 89.33 | 97.46 |
| Vowel      | 99.46<br>(0.28)        | 96.56 | 99.76 | <b>99.52</b><br>(0.35) | 97.44 | 99.73 | 95.56<br>(0.13)        | 99.89 | 95.12 | 99.39<br>(0.24)        | 98.33 | 99.50 |
| Thyroid    | 97.33<br>(0.19)        | 80.14 | 98.85 | <b>97.73</b><br>(0.32) | 83.42 | 98.88 | 94.66<br>(0.21)        | 99.56 | 94.27 | 97.66<br>(0.27)        | 96.60 | 97.74 |

**Table 4.**  $g$  values over 7 UCI datasets for the various boosting procedures (the lower values give the standard deviations of 10 times 10-fold cross validation). Best values (in bold face) are given by MLPs and positive boosting in two cases. Balanced boost gives best values in the other three, and it is the second best in the other 4

| Dataset    | std. boostMLP         | std. boostPP   | neg. boostPP   | pos. boostPP          | bal. boostPP          |
|------------|-----------------------|----------------|----------------|-----------------------|-----------------------|
| Ionosphere | <b>84.3</b><br>(1.89) | 80.8<br>(2.02) | 80.1<br>(1.52) | 68.7<br>(1.85)        | 81.4<br>(2.01)        |
| Diabetes   | 70.0<br>(1.75)        | 68.7<br>(1.06) | 69.2<br>(0.92) | 59.7<br>(0.92)        | <b>72.3</b><br>(1.15) |
| Cancer     | 95.4<br>(0.55)        | 95.1<br>(0.52) | 95.6<br>(0.49) | <b>97.0</b><br>(0.25) | 96.3<br>(0.38)        |
| Vehicle    | 76.4<br>(0.60)        | 69.5<br>(1.05) | 69.7<br>(0.99) | <b>78.1</b><br>(0.67) | 76.5<br>(0.88)        |
| Glass      | 92.4<br>(1.52)        | 91.8<br>(1.84) | 92.3<br>(1.38) | 92.8<br>(1.32)        | <b>93.3</b><br>(1.96) |
| Vowel      | <b>99.0</b><br>(0.12) | 98.1<br>(0.80) | 98.1<br>(0.66) | 97.5<br>(0.25)        | 98.9<br>(0.48)        |
| Thyroid    | 95.2<br>(0.32)        | 89.0<br>(1.20) | 90.8<br>(0.50) | 96.9<br>(0.18)        | <b>97.2</b><br>(0.35) |

in training of the more abundant majority negative class. Redundant pattern removal partially takes care of this but it is also interesting to avoid mistraining effects by near noisy label negative patterns, that is, the set  $nB_t^-$  of those negative patterns  $X$  with a wrong margin  $y_X act_i(X) < 0$  in all perceptrons in the  $t$  iteration. This can be done by lowering their  $d_{t+1}(X)$  probabilities, for which one option is to set  $R(X) = -1$ ; we should expect this to augment the accuracy  $a^+$  of the positive class, while lowering the accuracy  $a^-$  of the negative class. We shall call the resulting procedure positive boosting. Of course, we may do the opposite, applying what we may call negative boosting by setting  $R(X) = 1$  for  $X \in nB_t^-$ , which in turn should increase the accuracy  $a^-$  of the negative class. A third, more balanced option is to augment the probability of positive borderline patterns (i.e., to set  $R(X) = 1$ ) but to be more “neutral” on the  $nB_t^-$  patterns, setting  $R(X) = 0$  for them, which will essentially leave their previous probabilities unchanged. While  $a^+$  would then be smaller than in positive boosting, the overall classification should be more balanced. We shall call the resulting procedure balanced boosting. We will measure the balance of positive and negative accuracies using the  $g$  coefficient, i.e., the geometric ratio  $g = \sqrt{a^+ a^-}$  of the positive  $a^+$  and negative  $a^-$  accuracies, first proposed in [7]. We shall report next numerical results over seven datasets.

## 4 Numerical Results

We shall use 7 problem sets from the well known UCI database (listed in table 2) referring to the UCI database documentation [4] for more details on these

problems. Some of them (glass, vowel, vehicle, thyroid) are multi-class problems; to reduce them to 2-class problems, we are taking as the minority classes the class 1 in the vehicle dataset, the class 0 in the vowel data set, and the class 7 in the glass domains (as done in [3]), and merged in a single class both sick thyroid classes. In general they can be considered relatively hard problems and, moreover, some of these problems provide well known examples of highly imbalanced positive and negative patterns, that make difficult classifier construction, as discriminants may tend to favor the (much) larger negative patterns over the less frequently positive ones. This is the case of the glass, vowel, thyroid and, to a lower extent, vehicle problems. In all problems we will take the minority class as the positive one.

PP training has been carried out as a batch procedure. In all examples we have used 3 perceptrons and parameters  $\gamma = 0.05$  and  $\eta = 10^{-2}$ ; for the thyroid dataset, we have taken  $\eta = 10^{-3}$ . As proposed in [1], the  $\eta$  rate does not change if the training error diminishes, but is decreased to  $0.9\eta$  if it augments. Training epochs have been 250 in all cases; thus the training error evolution has not been taken into account to stop the training procedure. Anyway, this error has an overall decreasing behavior. We have performed 10 boosting iterations. In all cases we have used 10-times 10-fold cross validation. That is, the overall data set has been randomly split in 10 subsets, 9 of which have been combined to obtain the initial training set, the size of which has. To ensure an appropriate representation of positive pattern, stratified sampling has been used. The final PPs' behavior has been computed on the remaining, unchanged subset, that we keep for testing purposes.

As mentioned before, accuracy is a first measure of a classifier's efficiency. Table 3 gives overall, positive and negative accuracies for the four construction procedures (best values are in bold face). It can be seen that negative boosting gives the best results in 4 cases. Standard and balanced boosting give the best accuracy on one problem each, but they are quite close in all others anyway. On the other hand, while positive boosting gives the best accuracy for the cancer dataset, the accuracy it achieves is the lowest in all the other. As it may be expected, it also achieves the highest  $a^+$  values, while standard and, of course, negative boosting strongly favor the negative class. In any case, table 3 also shows that balanced boosting gives the best balance between positive  $a^+$  and negative  $a^-$  accuracies. The better inter-class performance of balanced boosting can also be seen in table 4. Balanced boosting achieves the best  $g$  values for 5 datasets and is a close second in the other two. Positive boosting gives the highest  $g$  for the cancer and vehicle datasets and comes second in two other problems. However, its  $g$  performance is quite poor for the diabetes and ionosphere problems. For their part, the  $g$  performance of standard and negative boosting comes behind, and while closer for other datasets, it is clearly poorer on the diabetes, vehicle and thyroid cases. The boosting performance of PPs is further compared in tables 2 and 4 with that of standard multilayer perceptrons (MLPs). As they are more powerful, boosted MLPs give clearly better accuracies than boosted PPs in

the ionosphere and vehicle problems, worse in the cancer and glass problems and similar in the others. When  $g$  values are considered in table 4, balanced boosting gives the best results in 3 problems, while MLP and positive boosting are better over 2 each; in all these 4 cases, balanced boosting  $g$  values are second best. In other words, balanced boosting gives the best overall performance among PP boosting methods, and that performance is comparable to that of MLP boosting, that has a much greater complexity and is considerably costlier to train.

## 5 Conclusions and Further Work

In this paper we have discussed how the concept of activation margin that arises very naturally on parallel perceptron training can be used to provide a more nuanced approach to boosting. This is done adding an extra factor  $R(X)$  to boosting's exponential probability update, whose values depend on the categorization of a given pattern  $X$  as redundant, label noisy or borderline obtained in terms of  $X$ 's activation margins. We set  $R(X) = 1$  for redundant and  $R(X) = -1$  for label noisy patterns, which causes boosting to lower their subsequent probabilities. Within borderline patterns we consider separately the near label noisy negative patterns. Setting  $R(X) = 1$  for them would increase their subsequent probabilities, augmenting thus the negative accuracy  $a^-$ , while  $a^+$  would increase if we set  $R(X) = 1$ . An equilibrium can be obtained setting  $R(X) = 0$ , thus keeping the probability of  $X$  essentially unchanged. The resulting procedure, balanced boosting, gives better PP classifiers in terms of the equilibrium between positive and negative accuracies, while achieving absolute accuracies close to the best achieved by the other methods. This performance is comparable to that of MLP boosting, while PP complexity is lower and training times much shorter than those of MLPs. Further work will concentrate on the effectiveness of general joint PP-boosting approach to malicious noise problems.

## References

1. P. Auer, H. Burgsteiner, W. Maass, *Reducing Communication for Distributed Learning in Neural Networks*, Proceedings of ICANN'2002, Lecture Notes in Computer Science 2415 (2002), 123–128.
2. E. Bauer, R. Kohavi, *An empirical comparison of voting classification algorithms: Bagging, boosting and variants*, Machine Learning 36 (1999), 105–139.
3. M. Kubat, S. Matwin, *Addressing the Curse of Imbalanced Training Sets: One-Sided Selection*, Proceedings of the 14th International Conference on Machine Learning, ICML'97 (pp. 179-186), Nashville, TN, U.S.A.
4. P. Murphy, D. Aha, *UCI Repository of Machine Learning Databases*, Tech. Report, University of California, Irvine, 1994.
5. N. Nilsson, **The Mathematical Foundations of Learning Machines**, Morgan Kaufmann, 1990.

6. R.E. Schapire, Y. Freund, P. Bartlett, W.S. Lee, *Boosting the margin: a new explanation for the effectiveness of voting methods*, *Annals of Statistics*, 26 (1998), 1651–1686.
7. J.A. Swets, *Measuring the accuracy of diagnostic systems*, *Science* 240 (1998), 1285–1293.

# A Reinforcement Learning Algorithm Using Temporal Difference Error in Ant Model

SeungGwan Lee<sup>1</sup> and TaeChoong Chung<sup>2</sup>

<sup>1</sup> School of Computer Science and Information Engineering, Catholic University, 43-1, Yeokgok 2-Dong, Wonmi-Gu, Bucheon-Si, Gyeonggi-Do, 420-743, Korea  
leesg@catholic.ac.kr

<sup>2</sup> School of Electronics and Information, KyungHee University, 1 Seocheon-Ri, Kiheung-Up, Yongin-Si, Gyeonggi-Do, 449-701, Korea  
tcchung@khu.ac.kr

**Abstract.** When agent chooses some action and does state transition in present state in reinforcement learning, it is important subject to decide how will reward for conduct that agent chooses. In this paper, we suggest multi colony interaction ant reinforcement learning model using TD-error to original Ant-Q learning. This method is a hybrid of multi colony interaction by elite strategy and reinforcement learning applying TD-error to Ant-Q. We could know through an experiment that proposed reinforcement learning method converges faster to optimal solution than original ACS and Ant-Q.

## 1 Introduction

Recently, ant model[1],[2],[3],[4],[5] is introducing by a special of reinforcement learning[6],[7]. When agent chooses some action and does state transition in present state in reinforcement learning, it is important subject to decide how will reward for conduct that agent chooses. In this paper, we suggest multi colony interaction ant reinforcement learning model using TD-error[8],[9](MCIARLM-TD) to Ant-Q to solve temporal-credit assignment problems.

The reinforcement learning that use TD-learning without waits final result. At each learning step, training error uses difference with prediction for output of present state and prediction for output of next state. The prediction for output of present state is updated to approximate with prediction for output of next state in TD-learning.

MCIARLM-TD learning is method that is proposed newly to improve Ant-Q, this method is a hybrid of multi colony interaction by elite strategy and reinforcement learning that apply TD-error that is proposed by C.J.C.H.Watkins to Ant-Q.

## 2 Apply TD-Error to Ant-Q

TD-learning that use TD-error calculates Q-function value of present state with Eq.(1) using difference with prediction for output of present state and prediction for output of next state.



$$Q(s_t, a_t) \leftarrow (1 - \alpha) \cdot Q(s_t, a_t) + \alpha \cdot TD\ error \quad (1)$$

Here,  $\alpha$  is learning rate, TD-error calculates with Eq.(2) as difference with prediction of present state and prediction of next state.

$$TD\ error = r_{t+1} + \gamma [ \underset{a \in A(s_t)}{Max} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) ] \quad (2)$$

$r_t$  is reinforcement value and  $\gamma$  is discount rate. The goal of Ant-Q is to learn AQ-values to find better solution as stochastic. AQ-values are updated by the following Eq.(3). Applying Eq.(2) to Ant-Q, it is expressed with Eq.(4).

$$AQ(r, s) \leftarrow (1 - \alpha) \cdot AQ(r, s) + \alpha \cdot (\Delta AQ(r, s) + \gamma \cdot \underset{z \in J_k(s)}{Max} AQ(s, z)) \quad (3)$$

$$TD\ error = \Delta AQ(r, s) + \gamma [ \underset{z \in J_k(s)}{Max} AQ(s, z) - AQ(r, s) ] \quad (4)$$

Finally, ant reinforcement learning model that apply TD-error in Ant-Q calculates Q-function value for node( $r, s$ ) of present state with Eq.(5).

$$AQ(r, s) \leftarrow (1 - \alpha) \cdot AQ(r, s) + \alpha \cdot (\Delta AQ(r, s) + \gamma \cdot [ \underset{z \in J_k(s)}{Max} AQ(s, z) - AQ(r, s) ]) \quad (5)$$

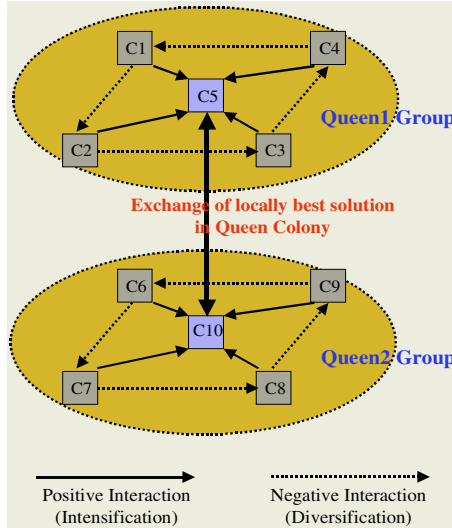
$$\begin{aligned} \text{where } \Delta AQ(r, s) &= 0 & , \text{ if Local updating} \\ \underset{z \in J_k(s)}{Max} AQ(s, z) - AQ(r, s) &= 0 & , \text{ if Global updating} \end{aligned}$$

$\alpha(0 < \alpha < 1)$  is the pheromone decay parameter,  $MaxAQ(s, z)$  is reinforcement value that receive from external environment to maximum, global reinforcement is zero, and  $\gamma$  is discount rate.

### 3 Multi Colony Ant Reinforcement Learning Using TD-Error

Multi colony interaction ant reinforcement learning model using TD-error have some independent ACS colonies, and interaction achieves search according to elite strategy between the colonies. Elite strategy is divided into intensification strategy and diversification strategy. Intensification strategy enables to select of good path to use heuristic information of other agent colony. This makes to select the high frequency of the visit of a edge by agents through positive interaction of between the colonies. Diversification strategy makes to escape selection of the high frequency of the visit of a edge by agents achieve negative interaction by search information of other agent colony. Network structure by elite strategy in multi colony interaction ant model is consisted of dual network structure. In Queen1 group, positive interaction by intensification strategy achieves

between (C1,C5), (C2,C5), (C3,C5), and (C4,C5) colonies and negative interaction by diversification strategy is achieved between other colonies. In Queen2 group, positive interaction achieves between (C6,C10), (C7,C10), (C8,C10), and (C9,C10) colonies and negative interaction is achieved between other colonies. Here, C5 and C10 are center agent colony(Queen Colony) and other colony are ergate colonies. And center colonies(C5, C10) achieve exchange of locally best solution. This means that achieve different duty between each colonies.



**Fig. 1.** Elite strategy interaction in multi colony ant model

All colonies independently achieve search each other in multi colony interaction ant model. First, examining the action of ergate colonies, agent( $k$ ) in colony( $l$ ) uses following Eq.(6) to move to node( $u$ ) at node( $r$ ).

$$s = \begin{cases} \arg \max_{u \in J_k^l(r)} \{ [AQ^l(r, u)]^{\delta(l)} \cdot [HE^l(r, u)]^{\beta(l)} \} , & \text{if } q \leq q_0(\text{exploitation}) \\ S & , \text{otherwise}(\text{exploration}) \end{cases} \quad (6)$$

$AQ^l(r, u)$  is Ant-Q value, be a positive real value associated to the edge( $r, u$ ) in colony( $l$ ), It is counterpart of Q-learning Q-values, and is intended to indicate how useful it is make to move node( $u$ ) when in node( $r$ ).  $AQ^l(r, u)$  is changed at run time.  $HE^l(r, u)$  is a heuristic value associated to edge( $r, u$ ) in colony( $l$ ) which allows an heuristic evaluation of which moves are better(in the TSP, the inverse of the distance). Let  $k$  be an agent whose task is to make a tour.  $J_k^l(r)$  are nodes still to be visited in colony( $l$ ), where  $r$  is the current node. Where  $\delta(l)$  and  $\beta(l)$  is parameters which weigh the relative importance of the learned AQ-values and the heuristic values.  $q$  is a value chosen randomly with uniform probability in

$[0,1]$ ,  $q_0(0 \leq q_0 \leq 1)$  is a parameter, and  $S$  is a random variable selected according to the distribution given by Eq.(7) which gives the probability with which an agent in node( $r$ ) choose the node( $s$ ) to move to.

$$p_k(r, s) = \begin{cases} \frac{[AQ^l(r, s)]^{\delta(l)} \cdot [HE^l(r, s)]^{\beta(l)}}{\sum_{u \in J_k^l(r)} [AQ^l(r, u)]^{\delta(l)} \cdot [HE^l(r, u)]^{\beta(l)}} & , \text{ if } s \in J_k^l(r) \\ 0 & , \text{ otherwise} \end{cases} \quad (7)$$

$$\text{where } [AQ^l(r, s)]^{\delta(l)} = \begin{cases} \sum_{n=1}^M [AQ^n(r, s)]^{\delta(l, n)} & , \text{ Positive} \\ \sum_{n=l-1}^l [AQ^n(r, s)]^{\delta(l, n)} & , \text{ Negative} \end{cases}$$

Where  $M$  is the total number of colony,  $\delta(l, n)$  displays degree of interaction that colony( $l$ ) receives from colony( $n$ ). Once, Interaction between the ergate colonies achieves negative interaction by diversification strategy. In case of negative interaction, it is influenced by total of the reciprocal value of negative of the frequency of the visit to relevant edge in previous colony and the reciprocal value of negative of the frequency of the visit to relevant edge on present colony. Through this negative interaction, agents of ergate colonies achieve various search by new region. And, Interaction between ergate colonies and center colony achieve positive interaction by intensification strategy. In case of positive interaction, center colony is influenced by the reciprocal value of positive of the frequency of the visit of relevant edge from all ergate colonies. Therefore, agents of center colony select good edge with search result of all ergate colonies, and intensify relevant edge. This differs with that assign foundation of a degree of interaction by fixing values[10], we dynamically allocate using the frequency of the visit.

If apply Ant-Q ant model that using TD-learning to multi colony interaction ant model, TD-error is expressed with Eq.(8).

$$TD \text{ error} = \Delta AQ^l(r, s) + \gamma \cdot [ \underset{z \in J_k^l(s)}{Max} AQ^l(s, z) - AQ^l(r, s) ] \quad (8)$$

Finally, ant reinforcement learning model that apply TD-error in Ant-Q calculates Q-function value for node( $r, s$ ) of present state with Eq.(9).

$$AQ^l(r, s) \leftarrow (1 - \alpha) \cdot AQ^l(r, s) + \alpha \cdot (\Delta AQ^l(r, s) + \gamma \cdot [ \underset{z \in J_k^l(s)}{Max} AQ^l(s, z) - AQ^l(r, s) ]) \quad (9)$$

where  $\Delta AQ^l(r, s) = 0$  , if Local updating

$$\underset{z \in J_k^l(s)}{Max} AQ^l(s, z) - AQ^l(r, s) = 0 \quad , \text{ if Global updating}$$

$\alpha(0 < \alpha < 1)$  is the pheromone decay parameter,  $Max AQ^l(s, z)$  is reinforcement value that receive from external environment to maximum, global reinforcement

is zero,  $\gamma$  is discount rate. Also,  $\Delta AQ^l(r,s)$  is reinforcement value, local reinforcement is always zero, while global reinforcement, which is given after all the agents have finished their tour, is computed by the following Eq.(10).

$$\Delta AQ^l(r,s) = \begin{cases} \frac{W}{L_{k_{ib}}^l} & , \text{if } (r,s) \in \text{tour done by the agent } k_{ib}^l \\ 0 & , \text{otherwise} \end{cases} \quad (10)$$

Where  $L_{k_{ib}}^l$  is the length of the tour done by the best agent, that is the agent which did the shortest tour in the current iteration, and  $W$  is a parameter, set to 10.

Also, Interaction between center colony(C5) of Queen1 group and center colony(C10) of Queen2 group select strategy that global updating about excellent solution, compares local best solution of two colonies.

### 4 Experimental Results

To prediction performance of multi colony interaction ant reinforcement learning model that apply TD-error(MCIARLM-TD) in Ant-Q, we measure performance through comparison with original ant model(ACS and Ant-Q).

We experimented the proposed model by using TSPLIB[12] which is a famous TSP example. Basis environment parameter for an experiment was decided as following, and optimum value decided by an experiment usually are  $\beta(l)=2$ ,  $\alpha=0.1$ ,  $q_0=0.9$ ,  $\gamma=0.3$ ,  $W=10$ ,  $\delta(l,n)=\pm(1/\text{the frequency of the visit of a edge})$  and the agent number used in each colony  $M$  was set to 10.

The initial position of agents assigned one agent in an each node at randomly, and the termination condition is that a fixed number of cycles or the value known as the optimum value was found.

Figure(2,3) shows the performance by the learning rate( $\alpha$ ) and discount rate( $\gamma$ ) in MCIARLM-TD ant model. Experiment was achieved increasing the

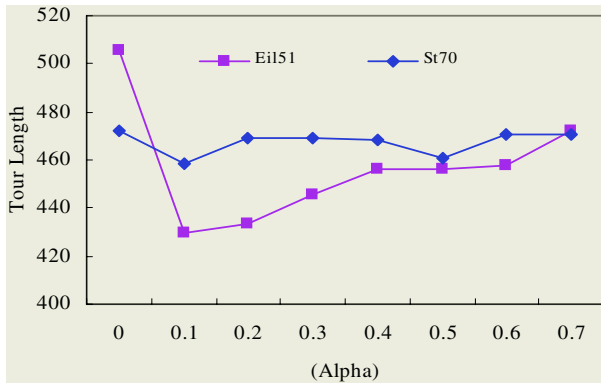


Fig. 2. Performance evaluation by the learning rate( $\alpha$ )

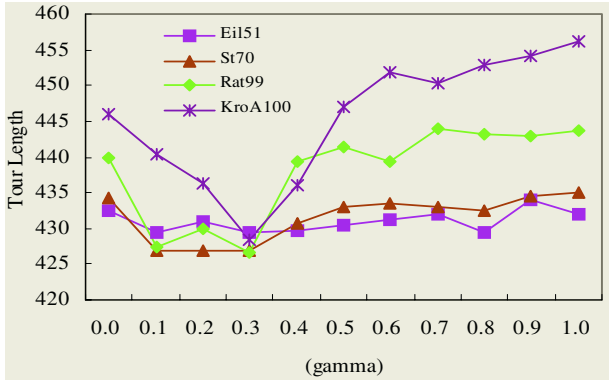


Fig. 3. Performance evaluation by the discount rate( $\gamma$ )

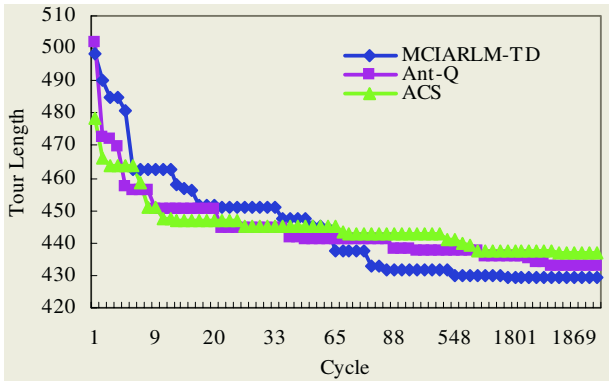


Fig. 4. Convergence speed estimation of each method

learning rate and discount rate by 0.1 step by step and the number of cycles was 2000.

Figure 2 shows the results by the learning rate( $\alpha$ ). We can see that Eil51 and St70 problems display almost similar graph shape according to the learning rate, and the performance reduced as the learning rate increases. When the learning rate is 0.1, shown almost similar optimal tour length. But, learning rate between 0.8 and 1.0 can not achieve learning in proposed model. The reason is that pheromone amount of each edge makes of negative number according as the learning rate is increased(more than learning rate 0.8). Therefore, we experimented in learning rate between 0.0 and 0.7 in proposed model. Here, so that we express two problems in uniformity graph, Eil51 problem displays original value that is worked, and St70 problem displays 62% value of result.

In Figure 3, when discount rate( $\gamma$ ) is 0.1 and 0.3 is showing satisfactory results. Specially, when discount rate is 0.3, satisfactory results is shown. Eil51 problem is original value of creation result, St70 problem is 63%, Rat99 problem is 35%, and KroA100 problem displays 2% value of result.

**Table 1.** Performance evaluation of MCIARLM-TD

| Node    | ACS[11]        |             | Ant-Q[6]       |             | MCIARLM-TD     |             |
|---------|----------------|-------------|----------------|-------------|----------------|-------------|
|         | Average length | Best length | Average length | Best length | Average length | Best length |
| KroA150 | 28908.8        | 27824       | 28761.12       | 27231       | 26576.04       | 26524       |
| Rat195  | 2571.63        | 2461        | 2514.08        | 2397        | 2490.25        | 2338        |
| Gil262  | 2636.75        | 2526        | 2592.35        | 2493        | 2481.68        | 2389        |
| A280    | 2892.58        | 2768        | 2840.52        | 2758        | 2690.61        | 2585        |
| Pr299   | 53497.8        | 51395       | 52714.41       | 50278       | 49143.87       | 48312       |
| Lin318  | 46244.4        | 44837       | 45318.53       | 43832       | 44571.98       | 43041       |

Figure 4 shows the convergence speed in case of repeated 2000 cycles to use Eil51.TSP. The convergence speed of ACS and Ant-Q are faster in beginning. However, according as search is proceeded, we can see that MCIARLM-TD converges faster.

Table 1 shows best tour length and average tour length that are achieved by each algorithms(ACS, Ant-Q and MCIARLM-TD) in case of repeated 20000 cycles in 10th trials about TSPLIB problems, the performance of proposed method is excellent.

## 5 Conclusion and Future Work

In this paper, we suggested multi colony interaction ant reinforcement learning model using TD-error(MCIARLM-TD) to original Ant-Q learning to solve temporal-credit assignment problems.

MCIARLM-TD ant model is method that is proposed newly to improve Ant-Q, this method is a hybrid of multi colony interaction by elite strategy and reinforcement learning that apply TD-error to Ant-Q, and converged faster to optimal solution by solving temporal-credit assignment problems to use TD-error while agents accomplish tour cycle.

MCIARLM-TD ant model using TD-error uses difference with prediction for output of present state and prediction for output of next state at each learning step, and updated to approximate with prediction for output of present state and prediction for output of next state in present state. And, the balance of exploration and exploitation that use elite strategy in multi colony interaction ant model is very important. Intensification strategy in elite strategy is equal with role of exploitation, and diversification strategy is equal with role of exploration. That is, exploitation selects continually historically good path, and exploration extends path search by new region. This enables various searching of new region, and also intensification role for specification edges.

Forward, we need research about reinforcement learning method that apply Eligibility factor that is measure that mean how is suitable about node selecting in present state in MCIARLM-TD ant model.

## References

1. A. Colorni, M. Dorigo, and V. Maniezzo.: An investigation of some properties of an ant algorithm. Proceedings of the Parallel Problem Solving from Nature Conference(PPSn 92), R. Manner and B. Manderick (Eds.), Elsevier Publishing, (1992) 509-520
2. A.Colorni, M. Dorigo, and V. Maniezzo.: Distributed optimization by ant colonies. Proceedings of ECAL91 - European Conference of Artificial Life, Paris, France, F.Varela and P.Bourgine(Eds.), Elsevier Publishing, (1991) 134-144
3. L.M. Gambardella and M. Dorigo.: Solving symmetric and asymmetric TSPs by ant colonies. Proceedings of IEEE International Conference of Evolutionary Computation, IEEE-EC 96, IEEE Press, (1996) 622-627
4. M. Drigo, V.Maniezzo, and A.Colorni.: The ant system: optimization by a colony of cooperation agents. IEEE Transactions of Systems, Man, and Cybernetics-Part B, vol. 26, No.2, (1996) 29-41
5. T. Stutzle and H. Hoos.: The ant system and local search for the traveling salesman problem. Proceedings of ICEC '97 IEEE 4th International Conference of Evolutionary, (1997)
6. L. M. Gambardella and M. Dorigo.: Ant-Q: a reinforcement learning approach to the traveling salesman problem. Proceedings of ML-95, Twelfth International Conference on Machine Learning, A. Prieditis and S. Russell (Eds.), Morgan Kaufmann, (1995) 252-260
7. M. Dorigo and L.M. Gambardella.: A study of some properties of Ant-Q. Proceedings of PPSN IVFourth International Conference on Parallel Problem Solving From Nature, H.M.Voigt, W. Ebeling, I. Rechenberg and H.S. Schwefel (Eds.), Springer-Verlag, Berlin, (1996) 656-665
8. C.N.Fiecher.: Efficient reinforcement learning. In Proceedings of the Seventh Annual ACM Conference On Computational Learning Theory, (1994) 88-97
9. E.Barnald.: Temporal-difference methods and markov model. IEEE Transactions on Systems, Man, and Cybernetics, 23, (1993) 357-365
10. Kawamura, H., Yamamoto, M., Suzuki, K. and Ohuchi, A.: Multiple Ant Colonies Algorithm Based on Colony Level Interactions, IEICE Transactions, Vol. E83-A, No.2, (2000) 371-379
11. L.M. Gambardella and M. Dorigo.: Ant Colony System: A Cooperative Learning approach to the Traveling Salesman Problem. IEEE Transactions on Evolutionary Computation, vol.1, No.1, (1997)
12. <http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95>

# Selection of Weights for Sequential Feed-Forward Neural Networks: An Experimental Study

Enrique Romero

Dept. de Llenguatges i Sistemes Informàtics,  
Universitat Politècnica de Catalunya  
eromero@lsi.upc.edu

**Abstract.** The selection of the frequencies of the new hidden units for sequential Feed-forward Neural Networks (FNNs) usually involves a non-linear optimization problem that cannot be solved analytically. Most models found in the literature choose the new frequency so that it matches the previous residue as best as possible. Several exceptions to the idea of matching the residue perform an (implicit or explicit) orthogonalization of the output vectors of the hidden units. An experimental study of the aforementioned approaches to select the frequencies in sequential FNNs is presented. Our experimental results indicate that the orthogonalization of the hidden vectors outperforms the strategy of matching the residue, both for approximation and generalization purposes.

## 1 Introduction

The selection of a proper number of hidden units for Feed-forward Neural Networks (FNNs) is a very important issue in practical applications, and it has been widely discussed through the literature. In terms of the Bias/Variance decomposition, as far as the number of hidden units of an FNN grows, bias decreases and variance increases [1]. This happens because the flexibility of the model grows with the number of hidden units [2]. Sequential approximations (also named constructive or incremental) with FNNs allow to dynamically construct the network, starting from scratch, without setting *a priori* the architecture [3]. Hidden units are sequentially added until a satisfactory solution is found.

The well-known architecture of a fully connected FNN with one output linear unit and one hidden layer of units computes a function  $f : \mathbb{R}^I \rightarrow \mathbb{R}$  defined as

$$f(x) = b_0 + \sum_{k=1}^N \lambda_k \varphi_k(\omega_k, x, b_k) \quad \omega_k \in \mathbb{R}^I \quad \lambda_k, b_0, b_k \in \mathbb{R},$$

where  $N$  is the number of units in the hidden layer and  $\varphi_k$  is the activation functions of the  $k$ -th hidden unit. For convenience, we refer to the weights in the first layer  $\{\omega_k\}_{k=1}^N$  (the non-linear weights) as *frequencies* and to the weights in the second layer  $\{\lambda_k\}_{k=1}^N$  (the linear weights) as *coefficients*. Regarding the *biases*,  $\{b_k\}_{k=1}^N$  can be considered as part of the frequencies and  $b_0$  as a coefficient.



Sequential algorithms have several shortcomings. First, it is not clear when to stop the addition of hidden units. Second, obtaining the frequencies of the new hidden units, although in theory is easier to solve than for a non-sequential procedure, is still a difficult non-linear optimization problem [4]. The coefficients can usually be computed analytically.

This work deals with the selection of weights of the new hidden units for sequential FNNs. In particular, we will focus on sequential approximations that obtain the new frequencies while keeping the previously selected frequencies fixed. That is, the previously obtained frequencies in the network are frozen in order to obtain the new one [5]. Most models found in the literature choose the new frequency so that it matches the previous residue as best as possible. Several exceptions to the idea of matching the residue perform an (implicit or explicit) orthogonalization of the output vectors of the hidden units, so that the new frequency is selected taking into account the interactions with the previously selected ones in order to minimize the global error. Theoretical results show that these two approaches may construct a sequence of output functions convergent to the target. However, the optimization problems posed at every step cannot be solved analytically in the general case. The difficulty lies on the selection of the frequencies, since the coefficients can usually be computed analytically.

An experimental study of the aforementioned approaches to select the frequencies in sequential FNNs is presented. The results in our experiments indicate that the orthogonalization of the hidden vectors outperforms the strategy of matching the residue, both for approximation and generalization purposes.

The rest of the paper is organized as follows. Several approaches to obtain the weights in sequential FNNs are discussed in section 2. The experiments can be found in section 3. Finally, section 4 outlines some directions for further research.

## 2 Selection of Weights in Sequential FNNs

Most of the sequential FNN models found in the literature keep the previously selected frequencies fixed and search only for the new frequency and the coef-

---

### Algorithm

**repeat**

- Increase by 1 the number of hidden units  $N$
- Pick an activation function for the new hidden unit
- Obtain the frequency of the new hidden unit
- Fix the frequency in the network
- Compute the coefficients

**until** a certain stopping criterion is satisfied

**end Algorithm**

---

**Fig. 1.** A general algorithm for sequential FNNs

ficients. Figure 1 shows the general structure of these sequential FNN schemes. Their output can be defined as ( $X_0 = 0$ ):

$$X_N = \sum_{k=1}^{N-1} \lambda_k^N h_{\omega_k} + \lambda_N^N h_{\omega_N}.$$

where  $X_N$  is a vector in  $\mathbb{R}^L$ ,  $N$  is the number of hidden units,  $L$  is the number of patterns in a data set  $D = \{x_1, \dots, x_L\}$ ,  $h_{\omega_k}$  is the output vector of the  $k$ -th hidden unit (with frequency  $\omega_k$ ) and  $\lambda_k^N$  is the coefficient of that hidden unit at step  $N$ . Output units are linear. The objective is to minimize the sum-of-squares error function  $\|f - X_N\|_2^2$ , where  $f$  is the target vector ( $f = \{f_1, \dots, f_L\}$ ).

The frequencies are obtained by optimizing a certain objective function, which somehow depends on the previously obtained frequencies:

1. Most of them choose the new frequency  $\omega_N$  so that it matches the previous residue as best as possible [5]. By inner product properties, it is equivalent to say that

$$\omega_N = \arg \max_{\omega} |\langle f - X_{N-1}, h_{\omega} \rangle|^2 / \|h_{\omega}\|_2^2 \tag{1}$$

where  $\langle \cdot, \cdot \rangle$  is the dot product in  $\mathbb{R}^L$  and  $h_{\omega}$  is the output vector of a hidden unit with frequency  $\omega$ . After the selection of every frequency, the coefficients of the whole network can be either optimized or not. By inner product properties, the optimal coefficients can be obtained solving a linear equations system [6]. When the whole set of coefficients are not optimized, the coefficient of the new hidden unit with frequency  $\omega_N$  is

$$\lambda_N^N = \langle f - X_{N-1}, h_{\omega_N} \rangle / \|h_{\omega_N}\|_2^2. \tag{2}$$

Some models sharing these underlying ideas are Projection Pursuit Regression [7] originally described in the Statistics field, Matching Pursuit [8] in the context of Signal Processing, or Projection Pursuit Learning Network [9] in the Neural Networks framework. Sometimes, convex approximations are constructed, as in the Incremental Linear Quasi-Parallel algorithm [10].

2. Several exceptions to the idea of matching the residue are the Orthogonal Least Squares Learning algorithm [11], Kernel Matching Pursuit with *pre-fitting* [12] or the Sequential Approximation with Optimal Coefficients and Interacting Frequencies [13], where an (implicit or explicit) orthogonalization of the output vectors of the hidden units is performed. The frequency  $\omega_N$  is selected taking into account the interactions of  $h_{\omega_N}$  with  $h_{\omega_1}, \dots, h_{\omega_{N-1}}$  in order to minimize  $\|f - X_N\|_2^2$ . There is no explicit intention to match the residue. In other words, a frequency  $\omega_1$  is considered better than  $\omega_2$  if  $h_{\omega_1}$  allows, together with the previously selected frequencies (and after computing the optimal coefficients of the whole network), a better approximation of  $f$  than  $h_{\omega_2}$ . The coefficients  $\{\lambda_k^N\}_{k=1}^N$  are always optimal.

In the following, we will refer to *MPR*, *OCMPR* and *SAOCIF* as:

- *MPR* (Matching the Previous Residue): The new frequency  $\omega_N$  is selected according to (1). The coefficients of the whole network are not optimized. That is,  $X_N = X_{N-1} + \lambda_N^N h_{\omega_N}$ , with  $\lambda_N^N$  computed according to (2).

- *OCMPR* (Optimal Coefficients after Matching the Previous Residue): The new frequency  $\omega_N$  is selected according to (1). Subsequently, the coefficients of the whole network are optimized by solving the associated linear equations system.
- *SAOCIF* (Sequential Approximation with Optimal Coefficients and Interacting Frequencies): The new frequency  $\omega_N$  is selected taking into account the interactions of  $h_{\omega_N}$  with  $h_{\omega_1}, \dots, h_{\omega_{N-1}}$  in order to minimize  $\|f - X_N\|_2^2$ . The coefficients are always optimal.

As it is well-known, an approximation scheme is said to have the convergence property if it is able to produce a sequence of output functions convergent to the target. *MPR*, *OCMPR* and *SAOCIF* have been proved to have the convergence property (see [14, 8] for *MPR*, [10, 5] for *OCMPR* or [13] for *SAOCIF*, for example). Unfortunately, these theoretical results cannot be directly applied in practice. The optimization problem posed in the general case cannot be solved analytically, and a suboptimal solution is searched heuristically. Note that the difficulty lies on the selection of the frequencies, since the coefficients can always be analytically computed.

### 3 Experiments

The experiments were performed on several artificial data sets. The main objectives of these experiments were to compare *MPR*, *OCMPR* and *SAOCIF* (see section 2). In particular, we studied the effect of their respective criteria of selection of frequencies both for approximation and generalization.

To that end, the algorithms in figure 2 were designed, as particular cases of that in figure 1. In the proposed algorithms, a number of candidate frequencies

---

#### Algorithm

##### repeat

Increase by 1 the number of hidden units  $N$

Pick an activation function for the new hidden unit

##### repeat

Assign a candidate frequency  $\omega$  to the new hidden unit

*MPR/OCMPR*: Set  $\omega_N := \omega$  if  $|\langle f - X_{N-1}, h_\omega \rangle|^2 / \|h_\omega\|_2^2$  is maximized

*SAOCIF*: Compute the optimal coefficients  $\{\lambda_k^N\}_{1 \leq k \leq N}$

*SAOCIF*: Set  $\omega_N := \omega$  if  $\|f - X_N\|_2^2$  is minimized

**until** there are no more candidate frequencies

Fix the frequency  $\omega_N$  in the network

*MPR*: Compute the coefficient  $\lambda_N^N = \langle f - X_{N-1}, h_{\omega_N} \rangle / \|h_{\omega_N}\|_2^2$

*OCMPR/SAOCIF*: Compute the optimal coefficients  $\{\lambda_k^N\}_{1 \leq k \leq N}$

**until** a certain stopping criterion is satisfied

**end Algorithm**

---

**Fig. 2.** Algorithms for *MPR*, *OCMPR* and *SAOCIF*

are selected at every step. The frequency finally chosen depends on the different objective functions of every algorithm.

5,000 random frequencies in  $[-0.5, +0.5] \times [-0.5, +0.5]$  were assigned as candidate frequencies for every model, and a maximum of 50 hidden units were added to the initial architecture. The rest of parameters (activation functions, gain factor and other internal parameters) were chosen in a preprocessing procedure (see [15] for details).

### 3.1 Data Sets

We used the same data sets described in [9], where five non-linear functions  $g_i : [0, 1]^2 \rightarrow \mathbb{R}$  were defined as follows:

- $g_1(x_1, x_2) = 10.391 ((x_1 - 0.4)(x_2 - 0.6) + 0.36)$ .
- $g_2(x_1, x_2) = 24.234 (r^2(0.75 - r^2))$ , where  $r^2 = (x_1 - 0.5)^2 + (x_2 - 0.5)^2$ .
- $g_3(x_1, x_2) = 42.659 ((2 + x_1)/20 + Re(z^5))$ , where  $z = x_1 + ix_2 - 0.5(1 + i)$ .
- $g_4(x_1, x_2) = 1.3356(1.5(1 - x_1) + e^{2x_1 - 1} \sin(3\pi(x_1 - 0.6)^2) + e^{3(x_2 - 0.5)} \sin(4\pi(x_2 - 0.9)^2))$ .
- $g_5(x_1, x_2) = 1.9 (1.35 + e^{x_1} \sin(13(x_1 - 0.6)^2)e^{-x_2} \sin(7x_2))$ .

225 pairs  $(x_1, x_2)$  of values were generated from the uniform distribution in  $[0, 1]^2$ . These data were used for all five functions in order to generate five noise-free training sets:

$$HEAn\text{-NF} = \{(x_1^i, x_2^i, g_n(x_1^i, x_2^i))\}_{i=1, \dots, 225}$$

where  $n \in \{1, 2, 3, 4, 5\}$ . In addition, another five training data sets were generated adding independent and identically distributed Gaussian noise:

$$HEAn\text{-WN} = \{(x_1^i, x_2^i, g_n(x_1^i, x_2^i) + 0.25\epsilon^i)\}_{i=1, \dots, 225}$$

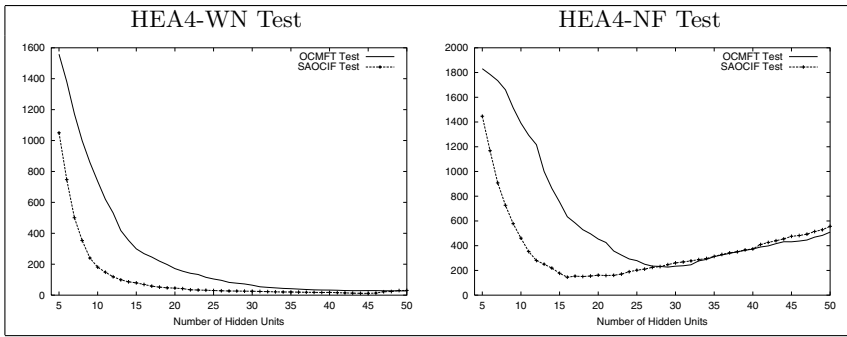
where  $\epsilon^i \sim \mathcal{N}(0, 1)$ . The test set was built sampling every function on a regularly spaced grid on  $[0, 1]^2$  with 10,000 points. In summary, 10 training sets (5 noise-free and 5 noisy versions) and 5 test sets were generated in [9] for the 5 aforementioned functions. These data sets have been widely used in the literature (see [5, 16, 17], for example).

In our experiments, we constructed 10 training sets for every function, each containing 225 points, changing the initial seed of the random function for the uniform distribution (the noise-free data sets). Similar to [9], 10 noisy training data sets were generated in the same way. For every function, the test set in [9] was used as a validation set for the adjustment of the parameters in the preprocessing procedure. For the final results, a new test set was constructed, with an offset of 0.0025 with respect to the input points in the original test set.

In summary, 100 training sets (50 noise-free and 50 noisy versions), 5 validation sets and 5 test sets were generated for the five aforementioned functions. For every function and version (noise-free and noisy), every model was trained with every one of the 10 different training sets, and tested on the test set.

**Table 1.** HEA data sets: Squared test set error and number of hidden units for *MPR*, *OCMPR* and *SAOCIF* in the tested data sets

| Data Set | Test Error    |                    |                      | Num. Hidden Units |              |               |
|----------|---------------|--------------------|----------------------|-------------------|--------------|---------------|
|          | <i>MPR</i>    | <i>OCMPR</i>       | <i>SAOCIF</i>        | <i>MPR</i>        | <i>OCMPR</i> | <i>SAOCIF</i> |
| HEA1-NF  | 0.06 (0.1)    | <b>0.00 (0.0)</b>  | <b>0.00 (0.0)</b>    | 49.3              | 20.3         | 9.6           |
| HEA2-NF  | 29.81 (2.6)   | 1.18 (0.5)         | <b>0.18 (0.1)</b>    | 49.1              | 47.1         | 44.1          |
| HEA3-NF  | 694.11 (27.2) | 31.41 (6.9)        | <b>5.57 (2.2)</b>    | 49.2              | 45.0         | 49.1          |
| HEA4-NF  | 69.29 (6.9)   | 24.32 (1.8)        | <b>9.80 (3.7)</b>    | 49.4              | 45.8         | 47.3          |
| HEA5-NF  | 49.22 (7.8)   | 14.79 (3.4)        | <b>7.76 (1.0)</b>    | 49.8              | 49.4         | 46.2          |
| HEA1-WN  | 14.37 (3.1)   | <b>10.70 (1.6)</b> | 15.37 (2.9)          | 39.1              | 4.5          | 4.4           |
| HEA2-WN  | 110.85 (6.8)  | 99.13 (7.1)        | <b>89.47 (6.3)</b>   | 30.9              | 19.1         | 15.4          |
| HEA3-WN  | 704.35 (50.5) | 303.65 (18.9)      | <b>256.01 (32.6)</b> | 49.9              | 33.6         | 31.6          |
| HEA4-WN  | 213.67 (8.0)  | 202.91 (14.7)      | <b>120.58 (11.5)</b> | 47.0              | 27.5         | 17.4          |
| HEA5-WN  | 174.63 (9.0)  | 211.82 (18.0)      | <b>173.35 (5.7)</b>  | 35.0              | 23.7         | 19.4          |

**Fig. 3.** Evolution of the average test error of *OCMPR* and *SAOCIF* with respect to the number of hidden units for the HEA4-NF (left) and HEA4-WN (right) data sets

### 3.2 Results

Results are shown in table 1 as the average of the minimum squared test set errors. Figures in boldface indicate the best results. Numbers in brackets are  $\hat{\sigma}_n/\sqrt{n}$ , the standard errors<sup>1</sup> estimated from the sample standard deviation  $\hat{\sigma}_n$ . The average number of hidden units where these minima are achieved is also shown. For the HEA1-NF data set, *SAOCIF* stopped with a fewer number of hidden units because the training error was less than 0.000001, so that it made no sense adding new hidden units. As an example, figure 3 shows a comparison of the evolution of the average test errors of *OCMPR* and *SAOCIF* with respect to the number of hidden units for the HEA4 data sets. The training errors were similar, although in a different scale, to left plot in figure 3.

<sup>1</sup> Under normality assumptions, the confidence interval can be computed from this value. For example, the deviation of the true value from the observed mean  $\bar{x}_n$  will be less than  $1.96\hat{\sigma}_n/\sqrt{n}$  with a probability of 0.95.

Regarding the overall behavior, *SAOCIF* obtains better results than *OCMPR*, which in turn compares favorably with *MPR*. This fact can be understood by looking at the number of hidden units of the obtained results:

1. For noise-free data sets the number of hidden units is around 50, the maximum number of allowed hidden units. This is due to the fact that overfitting was not observed during the learning process with these data sets (see left plot in figure 3). Therefore, the best results are obtained by those models that are able to fit more accurately the data. The scheme of selection of frequencies of *SAOCIF* allows to find better approximations with the same number of hidden units as *OCMPR* or *MPR*. The same happens when *OCMPR* is compared to *MPR*.
2. For noisy data sets, there is a high correlation between the number of hidden units and the goodness of the model: those models that attain their minima with less hidden units usually obtain better results. *SAOCIF* obtains simpler models (in terms of the number of hidden units), with the same training error, than *OCMPR*. According to the Bias/Variance trade-off, the minimum test set error should be smaller for *SAOCIF* than for *OCMPR*, as observed. The same happens with respect to *MPR*.

Therefore, the interacting frequencies selected by *SAOCIF* show a better behavior than frequencies selected so as to match the previous residue, both for approximation and generalization purposes. It can be observed that the differences among methods are greater for noise-free data sets than for noisy ones.

## 4 Future Work

New experiments can be done by changing the heuristic to select the new frequencies in the tested models. It would be interesting to test if other heuristics also maintain the observed differences. Some variation of the Breeder Genetic Algorithm [18], for example, could be useful to that end.

A different point of view can be introduced in sequential FNNs if we reconsider the goodness of the previously selected frequencies. Comparing the addition of hidden units in a sequential FNN with the selection of features in the Sequential Forward Selection procedure for Feature Selection [19], we can see that they share the same general ideas, although applied to different objects. Whereas sequential FNNs applies the forward selection to the hidden units, Sequential Forward Selection applies it to the features. Therefore, other Feature Selection search procedures can be applied to the construction of FNNs.

## Acknowledgements

This work was supported by Consejo Interministerial de Ciencia y Tecnología (CICYT), under projects DPI2002-03225 and CGL2004-04702-C02-02.

## References

1. Geman, S., Bienenstock, E., Doursat, R.: Neural Networks and the Bias/Variance Dilemma. *Neural Computation* **4** (1992) 1–58
2. Barron, A.R.: Approximation and Estimation Bounds for Artificial Neural Networks. *Machine Learning* **14** (1994) 115–133
3. Kwok, T.Y., Yeung, D.Y.: Constructive Algorithms for Structure Learning in Feedforward Neural Networks for Regression Problems. *IEEE Transactions on Neural Networks* **8** (1997) 630–645
4. Kůrková, V.: Incremental Approximation by Neural Networks. In Karny, M., Warwick, K., Kůrková, V., eds.: *Dealing With Complexity: A Neural Network Approach*. Springer-Verlag, London (1998) 177–188
5. Kwok, T.Y., Yeung, D.Y.: Objective Functions for Training New Hidden Units in Constructive Neural Networks. *IEEE Transactions on Neural Networks* **8** (1997) 1131–1148
6. Achieser, N.I.: *Theory of Approximation*. Frederick Ungar Pub. Co., New York (1956)
7. Friedman, J.H., Stuetzle, W.: Projection Pursuit Regression. *Journal of the American Statistical Association* **76** (1981) 817–823
8. Mallat, S.G., Zhang, Z.: Matching Pursuits with Time-Frequency Dictionaries. *IEEE Transactions on Signal Processing* **41** (1993) 3397–3415
9. Hwang, J.N., Ray, S.R., Maechler, M., Martin, D., Schimert, J.: Regression Modelling in Back-Propagation and Projection Pursuit Learning. *IEEE Transactions on Neural Networks* **5** (1994) 342–353
10. Kůrková, V., Beliczyński, B.: Incremental Approximation by One-Hidden-Layer Neural Networks. In: *International Conference on Artificial Neural Networks*. Volume 1. (1995) 505–510
11. Chen, S., Cowan, C.F.N., Grant, P.M.: Orthogonal Least Squares Learning Algorithm for Radial Basis Function Networks. *IEEE Transactions on Neural Networks* **2** (1991) 302–309
12. Vincent, P., Bengio, Y.: Kernel Matching Pursuit. *Machine Learning* **48** (2002) 165–187 Special Issue on New Methods for Model Combination and Model Selection.
13. Romero, E., Alquézar, R.: A New Incremental Method for Function Approximation using Feed-forward Neural Networks. In: *International Joint Conference on Neural Networks*. Volume 2. (2002) 1968–1973
14. Jones, L.K.: On a Conjecture of Huber Concerning the Convergence of Projection Pursuit Regression. *The Annals of Statistics* **15** (1987) 880–882
15. Romero, E.: *Learning with Feed-forward Neural Networks: Three Schemes to Deal with the Bias/Variance Trade-off*. PhD thesis, Departament de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, Spain (2004)
16. Treadgold, N.K., Gedeon, T.D.: Exploring Constructive Cascade Networks. *IEEE Transactions on Neural Networks* **10** (1999) 1335–1350
17. Ma, L., Khorasani, K.: New Training Strategies for Constructive Neural Networks with Application to Regression Problems. *Neural Networks* **17** (2004) 589–609
18. Mühlenbein, H., Schlierkamp-Voosen, D.: Predictive Models for the Breeder Genetic Algorithm I. Continuous Parameter Optimization. *Evolutionary Computation* **1** (1993) 25–49
19. Liu, H., Motoda, H.: *Feature Selection for Knowledge Discovery and Data Mining*. Kluwer Academic Publishers (1998)

# Exploiting Multitask Learning Schemes Using Private Subnetworks

Pedro J. García-Laencina<sup>1</sup>, Aníbal R. Figueiras-Vidal<sup>2</sup>, Jesús Serrano-García<sup>1</sup>,  
and José-Luis Sancho-Gómez<sup>1</sup>

<sup>1</sup> Dpto. Tecnologías de la Información y las Comunicaciones,  
Universidad Politécnica de Cartagena, Cartagena-Murcia, Spain

<sup>2</sup> Dpto. Teoría de la Señal y Comunicaciones,  
Universidad Carlos III de Madrid, Leganés-Madrid, Spain

**Abstract.** Many problems in pattern recognition are focused to learn one main task, *Single Task Learning* (STL). However, most of them can be formulated from learning several tasks related to the main task at the same time while using a shared representation, *Multitask Learning* (MTL). In this paper<sup>1</sup>, a new MTL architecture is proposed and its performance is compared with those obtained from other previous schemes used in MTL. This new MTL scheme makes use of private subnetworks to induce a bias in the learning process. The results provided from artificial and real data sets show how the use of this private subnetworks in MTL produces a better generalization capabilities and a faster learning.

## 1 Introduction

Human learning frequently involves learning several tasks simultaneously; in particular, humans compare and contrast similar tasks for solving a problem. For example, if you want to learn periodic table, its easier learning groups of related elements than learning the complete table. Nevertheless, most approaches to machine learning focus on the learning of a single isolated task, *Single Task Learning* (STL). This type of framework has been achieved great success, but it overlooks basic details and advantages of human learning.

In the recent last years, several works have extended the STL adding extra tasks related to the main one, learning them at same time [1, 2, 3]. The task which is desired to be learnt better is called the main task and the task whose training data are used as hints by the main task are called as the extra tasks. This approach to learning is known as *Multitask Learning* (MTL) [2].

In this article, we study several MTL schemes and a new proposal is introduced. We analyze its performance on different artificial and real problems. Results show how the use of this private subnetworks in MTL produces a better

---

<sup>1</sup> This work is partially supported by Ministerio de Educación y Ciencia under grant TIC2002-03033.



generalization capabilities and a faster learning. The rest of this article is structured as follows. In Section 2, we give an overview of multitask learning and describe several related works. Section 3 shows different MTL schemes used by Caruana in his thesis [4], and the new scheme is proposed. Section 4 describes the data problems used in this work to test the architecture we propose. The section concludes showing the simulation results. Finally, several comments and conclusions complete this paper.

## 2 Previous Works

In his work, Hinton analyzes the generalization capabilities in artificial neural networks [5]. He shows that these capabilities are improved if networks learn to represent underlying regularities of the domain. Following this idea, Baxter introduced the concept of *inductive bias* [6]. Inductive bias is anything that causes an inductive learner to prefer some hypotheses over other hypotheses. A learner that learns many related tasks at the same time can use these tasks as inductive bias for each other and thus better learn the regularities of the domain. In this sense, the union of the main task and all extra related tasks make up the domain. This can make learning more accurate and may allow hard tasks to be solved in a better way that when they are learned in isolation [7].

The MTL term was introduced in 1993 by Rich Caruana [2]. Multitask learning is an approach to inductive transfer (using what is learned for one problem to help another problem) that emphasizes learning multiple tasks in parallel while using a shared representation. The simplest way to implement this in neural network is to add extra outputs (extra tasks) to the net. In his Ph. D. Thesis [4], R. Caruana analyzed MTL using *MultiLayer Perceptrons* (MLP) trained with Backpropagation algorithm [8]. In a MLP network, MTL is implemented by having extra outputs share a common hidden layer in the network with the main task output(s). This sharing promotes inductive transfer, i. e., the hidden layer representations learned for the extra outputs are available to the main task output(s) and often improve performance on the main task.

## 3 Exploiting MTL Using Private Subnetworks

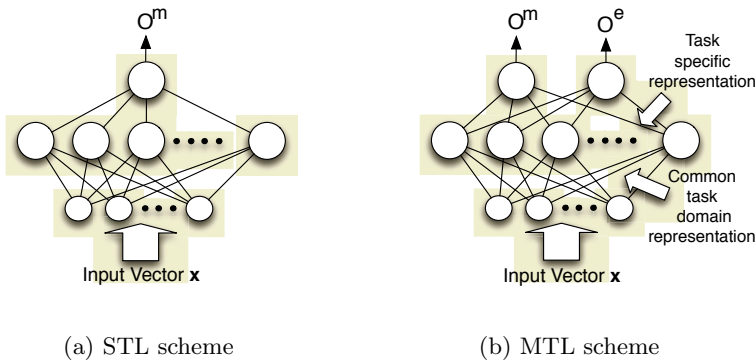
In order to make the content of the paper easier, the nomenclature used in the following sections is showed in Table 1.

In this section, we study the MTL schemes proposed by Caruana and introduce our MTL architecture. Before analyzing MTL, we describe STL approach. Consider a dataset  $\mathcal{M}$ , associated to a single task, with its respective input set  $\mathbf{X}^m$  and target set  $\mathbf{T}^m$ . Figure 1(a) shows STL scheme for solving this problem. This net can be trained to minimize an error function between network outputs and target values. Therefore, network learns only a single task, in other words, it learns only targets  $\mathbf{T}^m$  from  $\mathbf{X}^m$ .

**Table 1.** Nomenclature of mathematical symbol used in this paper

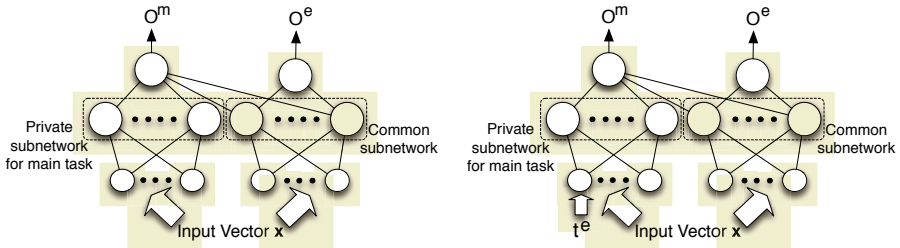
|                                                                                                         |                                 |
|---------------------------------------------------------------------------------------------------------|---------------------------------|
| $\mathbf{X} = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(i)}, \dots, \mathbf{x}^{(N)}\}$ | Input set                       |
| $\mathbf{T} = \{t^{(1)}, t^{(2)}, \dots, t^{(i)}, \dots, t^{(N)}\}$                                     | Target set                      |
| $\mathcal{M} = \{\mathbf{X}^m, \mathbf{T}^m\}$                                                          | Set to learning main task       |
| $\mathcal{E} = \{\mathbf{X}^e, \mathbf{T}^e\}$                                                          | Set to learning extra task      |
| $\mathbf{x}^{(i)} = [x_1^{(i)}, x_2^{(i)}, \dots, x_j^{(i)}, \dots, x_d^{(i)}]$                         | Input vector                    |
| $t^{(i)}$                                                                                               | Target or desired value         |
| $o^m, o^e$                                                                                              | Net outputs associated to tasks |

Thus, a main task and one or several tasks are learned all together in a MTL scheme. MTL uses the idea to create extra tasks that get trained on the same net with the main task. Consider a dataset  $\mathcal{M}$ , associated to a main task, with its respective input set  $\mathbf{X}^m$  and target set  $\mathbf{T}^m$  and a dataset  $\mathcal{E}$ , associated to a extra task, with its respective input set  $\mathbf{X}^e$  and target set  $\mathbf{T}^e$ . Now, the network can be trained to minimize an error function between network outputs and target values for each task. The aim is that network will accurately predict new values to main task for future inputs not used in the training process (or test set). Often, it is not cared how well extra tasks are learned because of their only purpose is to help the main task be learned better. A first approach is to use only one network with a hidden layer of neurons for learning all tasks. We call this architecture as *standard MTL* scheme. Figure 1(b) shows an MTL feedforward neural network with a hidden layer and an output associated with each task.

**Fig. 1.** STL vs. MTL in its standard version

The outputs associated with these tasks are fully connected to the hidden layer. The hidden layer of this net is shared by all tasks. This is the central idea in MTL: to share the learned information while the tasks are learned in parallel. Compared to this, when each task is learned in isolation, there is not sharing of information among the tasks. With respect to weights, those of the first layer are updated depending on the error of all tasks; while the weights

that connect each output unit to hidden neurons are only influenced by errors produced by the associated task. It is possible to improve MTL performance using network architectures more complicated than standard MTL scheme. One solution is proposed by Caruana [4], adding a *private* or *specific subnetwork* to learn only the main task. Figure 2(a) shows this scheme. Therefore, there are now two disjoint hidden layers or two disjoint subnetwork. One of them is a private subnetwork used only by the main task, while other is the common subnetwork shared by the main task and the extra task. This common subnetwork supports MTL transfer. This net architecture is asymmetric because the main task can see and affect the private subnetwork used by the extra task, but the extra task can not see or affect the subnetwork reserved for the main task. In other words, the private subnetwork learn only the regularities of the main task and the common subnetwork learn the regularities of the domain; therefore, use of private subnetworks helps to get better the generalization of the main task.



(a) A MTL scheme with a private subnetwork used by the main task

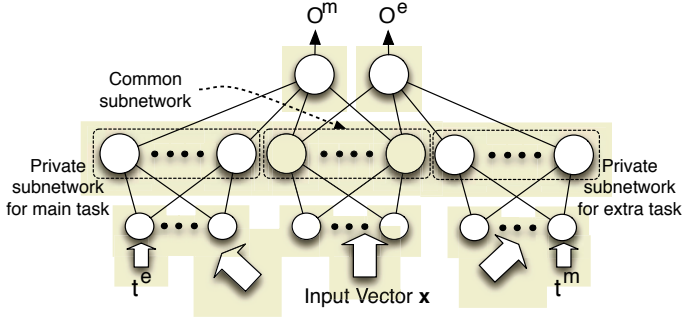
(b) A MTL scheme with a private subnetwork used by main task and an extra input

**Fig. 2.** MTL schemes with a private subnetwork

Up to now, we have supposed that inputs  $\mathbf{x}$  are equal to all hidden neurons; using our notation,  $\mathbf{X}^e$  are the same that  $\mathbf{X}^m$ . But performance of MTL can be improved if we introduce the desired values  $t^e$  together with inputs  $\mathbf{x}^m$  to learn the main task. Figure 2(b) shows this architecture. In other words, we are using the extra task learned in common subnetwork  $\mathbf{X}^m = [\mathbf{X}, \mathbf{T}^e]$ ; therefore, we are adding a priori information about domain in private subnetwork aiding to the main task learning in order to produce a better generalization capability.

Finally, we propose a different MTL scheme that uses the concepts of private subnetworks and extra inputs. Figure 3 shows this scheme. This architecture has a private subnetwork for each task, one for main task and other for extra task. Each private subnetwork is connected to output unit associated to each task, thus, learning the specific regularities of each one. The common subnetwork is full-connected with outputs units, then it will learn the complete regularities of the domain. In other words, private subnetworks play the roll of a STL scheme for each task, while common subnetwork works as a MTL scheme. Moreover,

this scheme uses as extra inputs in every private subnetwork the targets corresponding to the another task (case of just two tasks, main and extra). This new MTL network presents several advantages over the previous one, improving the performance and the generalization capabilities. Next sections show these aspects.



**Fig. 3.** A MTL scheme with a private subnetwork and extra input to each task. Common subnetwork learns the regularities of the domain and private subnetworks learn specific regularities of each task, using moreover the desired values of the other task as extra input

## 4 Experiments

After a short description of each problem, we present the results of our experiments. It will be proved how the use of additional private subnetwork and the extra inputs on the training process improves the generalization capability of the trained network.

### 4.1 Description of Data Sets

We evaluate performance of all MTL schemes presented in the previous section using two different problems. Both problems have two tasks, a main task and an extra task. Our aim is improving the learning of main task by means of MTL schemes.

**Binary Problem.** This problem is implemented by Caruana for testing his schemes [4]. Inputs are binary vectors of 8 bits,  $\mathbf{x} = [x_1, \dots, x_8]$ , so there are 256 samples. We consider two related tasks:

$$\text{Task A: } x_1 \vee (2 \cdot Nbits(x_2, \dots, x_4) < Nbits(x_5, \dots, x_8))$$

$$\text{Task B: } x_1 \wedge (2 \cdot Nbits(x_2, \dots, x_4) < Nbits(x_5, \dots, x_8))$$

where  $Nbits()$  returns the number of bits set to 1 in its argument, “ $\vee$ ” denotes the logical OR relation, and “ $\wedge$ ” the logical AND relation. From this definition of both tasks, it is obvious that they are clearly related. In the experiments, task A is used as main, and task B as extra. From the 256 cases, 128 cases have been randomly selected as training set and the remaining samples as test set.

**Monk’s Problem.** This problem describes an artificial robot domain in which appearance of robots is described by 6 different categorical attributes [9],  $x_1, \dots, x_6$ . The learning task is a binary classification task for three related problem. In our experiments, we use two problems to test MTL schemes. In particular:

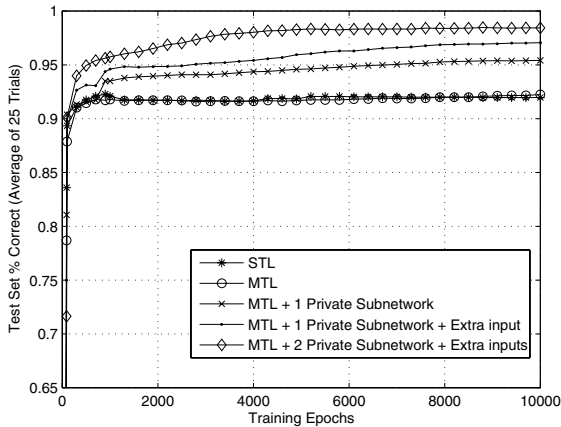
Task A:  $(x_1 = x_2) \vee (x_5 \text{ has the first categorical value})$

Task B: *exactly two of the six attributes have the first categorical value*

We have run experiments using task A as main and task B as extra. There are 432 cases, where 124 are used as training set and the 432 cases as test set.

### 4.2 Results

All the activation functions in the network are the hyperbolic tangents. The cost function to be minimized is the root-mean-squared-error (RMSE) of the output with respect to the desired values. The gradients are calculated using back propagation algorithm. Before learning, all weights are initialized randomly with values from the interval  $[-0.5, 0.5]$ . All weights are adapted through gradient descent method in batch mode with adaptive learning rate and momentum term (0.5). The test set is used to evaluate the percentage of correct classifications after training, thus measuring the generalization capability of the network. A validation set has not been used to stop training because of we have observed that MTL training is low sensitive to overfitting.

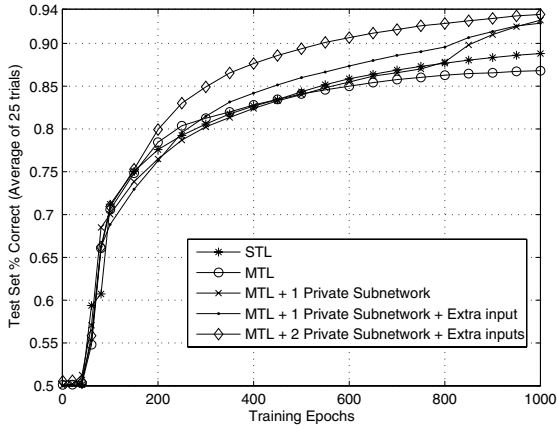


**Fig. 4.** Test-percentage of correct classifications on main task of the Binary Problem

In the binary problem, we use 100 hidden neurons both for STL as standard MTL; 40 neurons in each subnetwork for MTL scheme with one private subnetwork; and 20 neurons in each subnetwork for the MTL scheme proposed. Figure 4 shows the test-set percentage curves on task A for all schemes. Each curve in the graph are the average of 25 trials using 10000 training epochs. Table 2 summarizes the final results of binary problem. In the case of percentage correct on extra task, we get 74.3% for standard MTL and around of 97% for all MTL schemes with private subnetworks.

**Table 2.** Test-set performance on Binary Problem for each scheme

| Scheme    | STL   | MTL   | MTL + 1 private subnet | MTL + 1 private subnet + extra input | MTL + 2 private subnet + extra inputs |
|-----------|-------|-------|------------------------|--------------------------------------|---------------------------------------|
| % Correct | 91.97 | 92.25 | 95.42                  | 97.1                                 | 98.44                                 |
| Std. Dev. | 0.171 | 0.164 | 0.017                  | 0.015                                | 0.022                                 |

**Fig. 5.** Test-percentage of correct classifications on main task of the Monk’s Problem**Table 3.** Test-set performance on Monk’s Problem for each scheme

| Scheme    | STL   | MTL   | MTL + 1 private subnet | MTL + 1 private subnet + extra input | MTL + 2 private subnet + extra inputs |
|-----------|-------|-------|------------------------|--------------------------------------|---------------------------------------|
| % Correct | 88.81 | 86.81 | 92.71                  | 92.35                                | 93.37                                 |
| Std. Dev. | 0.151 | 0.113 | 0.075                  | 0.089                                | 0.069                                 |

In Monk’s problem, we use 6 hidden neurons both for STL as standard MTL; 3 neurons in each subnetwork for MTL scheme with one private subnetwork; and 2 neurons in each subnetwork for the MTL scheme proposed. Figure 5 shows the test-set percentage correct curves on task A for all schemes. It shows that adding private subnetworks speeds up learning. Each curve in the graph are the average of 25 trials using 1000 training epochs. Table 3 summarizes the final results on this problem. In the case of percentage correct on extra task, we get 69.9% for standard MTL; around of 71% for MTL schemes with one private and 78.4% for the MTL scheme proposed.

## 5 Discussion and Conclusion

Results shows that MTL can beneficiate learning of the main task, increasing its performance and generalization capabilities. By means of using private subnetworks for each task, the capabilities of the standard MTL are clearly improved. Private subnetworks learn specifically one task, while a common subnetwork learns the complete domain. Simulations over two problems show that the new scheme proposed here obtains better generalization and faster convergence than the rest of considered alternatives. Moreover, the new MTL architecture not only improves the learning performance of the main task, but also improves learning of the extra task. Therefore, proposed scheme is optimum in problems where it's necessary to have good learning in all tasks. It is important to note that these advantages are achieved without increasing the complexity of the network, i.e., without increasing the number of neurons used in standard MTL scheme.

This work will stimulate future work in several directions. Some of them are an extended study to more than two class problems and the use of a separate learning rate for each task, according to the recent works of Silver [10].

## References

- [1] Abu-Mostafa, Y. S. Learning from hints in neural networks. *Journal of Complexity*. **6(2)**. (1990), 192-198
- [2] Caruana, R. Multitask learning: a knowledge-based source of inductive bias. *Proceedings of the 10<sup>th</sup> International Conference of Cognitive Science*. (1993), 41-48
- [3] Thrun, S. Is learning the n-thing any easier than learning the first?. *Advances in Neural Information Processing Systems (NIPS)*. (1996), 640-646
- [4] Caruana, R. *Multitask learning*. Ph. D. Thesis. Carnegie Mellon University. (1997)
- [5] Hinton, G.E. Learning distributed representations of concepts. *Proceedings of the 8<sup>th</sup> International Conference of Cognitive Science*. (1986), 1-12
- [6] Baxter, J. *Learning Internal Representations*. Ph. D. Thesis. The Flinders University of South Australia. (1994)
- [7] Baxter. J. A model of inductive learning bias learning. *Journal of Artificial Intelligence Research*. (2000), 12:149-98
- [8] Rumelhart, D. E., Hinton, G. E., Williams, R. J. Learning Representations by Back-propagating Errors. *Nature*,323. (1986), 533-536
- [9] Wnek, J., Michalski, R. S. Comparing symbolic and subsymbolic learning: Three studies. In *R.S. Michalski and G. Tecuci, editors, Machine Learning: A Multi-strategy Approach*, volume 4, (1993), 318-362.
- [10] Silver, D. *Selective transfer of neural network task knowledge*. Ph. D. Thesis. University of Western Ontario. (2000)

# Co-evolutionary Learning in Liquid Architectures

Igal Raichelgauz<sup>1</sup>, Karina Odinaev<sup>2</sup>, and Yehoshua Y. Zeevi<sup>1</sup>

<sup>1</sup> Department of Electrical Engineering, Technion – Israel Institute of Technology,  
Haifa 32000, Israel

igal.raichelgauz@intel.com,  
zeevi@ee.technion.ac.il

<sup>2</sup> Department of Bio-Medical Engineering, Technion – Israel Institute of Technology,  
Haifa 32000, Israel

karinao@tx.technion.ac.il

**Abstract.** A large class of problems requires real-time processing of complex temporal inputs in real-time. These are difficult tasks for state-of-the-art techniques, since they require capturing complex structures and relationships in massive quantities of low precision, ambiguous noisy data. A recently-introduced Liquid-State-Machine (LSM) paradigm provides a computational framework for applying a model of cortical neural microcircuit as a core computational unit in classification and recognition tasks of real-time temporal data. We extend the computational power of this framework by closing the loop. This is accomplished by applying, in parallel to the supervised learning of the readouts, a biologically-realistic learning within the framework of the microcircuit. This approach is inspired by neurobiological findings from ex-vivo multi-cellular electrical recordings and injection of dopamine to the neural culture. We show that by closing the loop we obtain a much more effective performance with the new Co-Evolutionary Liquid Architecture. We illustrate the added value of the closed-loop approach to liquid architectures by executing a speech recognition task.

## 1 Introduction

Of the various alternatives, large, random, vastly connected cortical networks are the best candidates for a core of biologically-motivated computational architectures. Moreover, even a relatively simple model composed of ~100 leaky-integrate-and-fire neurons connected by dynamic synapses with stochastic heterogeneous parameters has an interesting computational power in a domain of parallel processing of temporal noisy data in real-time.

A new computational paradigm, called Liquid-State-Machine (LSM), recently presented by [1], provides a theoretical basis for applying a model of neural microcircuit to generic computational tasks. The LSM system is composed of two parts: (1) Liquid – a model of neural microcircuit is used as a "reservoir" of complex dynamics to transform the input time series  $u(.)$  into "liquid states"  $x(t)$ . (2) Readout – memory-less function which maps the liquid state  $x(t)$  at time  $t$  onto the output  $v(t)$ . Readout may be implemented by a simple one-layer network of perceptron, trained by



linear algorithm to build a function mapping liquid-states onto desired outputs. It was shown [2] by the means of simulations that such a system is computationally effective in executing parallel tasks of recognition and classification of temporal data. LSM is identified with properties of anytime computing, by processing spatio-temporal input in real time. The computational core of LSM is randomly structured generic filter with dynamical properties of fading memory and separation.

In the framework of computational LSM, a neural microcircuit is used as an efficient generic filter transforming different temporal inputs into significantly different liquid states. The task-dependent part is executed by the readout after being trained by supervised-learning algorithm to map these states onto predefined output. Turning back to neurobiological facts, the plasticity and learning ability of real cortical networks should not be neglected in the biologically-motivated computational framework. A feedback from the environment drives the learning process in neurobiological systems and allows the success in tasks varying in time rather than being predefined. In this study we extend LSM computational framework to a closed-loop setup wherein feedback from the environment drives the learning process of the computational core liquid unit – Neural Microcircuit (NM).

However, it is not a straightforward task to define a learning algorithm to such a large and randomly constructed network, therefore an inspiration from neurobiological findings is required again. We use two neurobiological paradigms for implementing the learning of the NM – Reward based learning [3] and Dopamine induced learning by dispersion mechanism [4]. The learning process of the NM is composed of two stages – the exploration of various states of NM and the recognition of the appropriate one. In the proposed closed-loop framework, this biologically-motivated learning of the NM is done in parallel to the supervised learning of the readout, i.e. there is a co-evolutionary learning process of NM and readout until the best performance of the overall system is reached.

## 2 Neural Microcircuit as a Generic Computational Unit

The neocortex is characterized by precise structure of columns and layers. Within neocortical layers neurons are mapped into each other, where anatomical and physiological properties are unique for each type of pre- and post-synaptic combination. However remarkable morphological, electrophysiological and spatial stereotypy exists in these networks, in addition to very stereotypical connectivity and patterning of synaptic connections between neighboring cells. This clear stereotypy exists across different regions of the brain, suggesting that there is a generic template of microcircuit and that all neocortical microcircuits are merely subtle variations of that common microcircuit template. Such templates could subserve the apparent omnipotent functional capacity of the neocortical microcircuitry [5]. A computational model of generic neural microcircuit is inherently endowed with powerful and versatile information processing capabilities. We used a similar model to [2], composed of a 3-dimensional recurrent network of 135 Leaky-Integrate-and-Fire (LIF) neurons with random connectivity, and similarity to generic cortical microcircuit, 20% of the neurons are randomly chosen to be inhibitory and,

accordingly, 80% excitatory. The probability of connection between two neurons depends on the distance between them according to,

$$C \cdot \exp(-D(i, j) / \lambda^2), \quad (1)$$

wherein  $\lambda$  and  $C$  are parameters that determine the average number of connections for a certain Euclidean distance  $D$  between the neuron  $i$  and neuron  $j$ . This connectivity characterization by primary local connections and a few longer connections is biologically realistic. Long range connections will be incorporated, and their functional effects on the computational properties of the network will be investigated within a context of a different study.

Random, heterogeneous parameters of NM model fit neurobiological data from rat somatosensory cortex [2]. Synaptic short-term plasticity of the NM is implemented by dynamic synapses in which the amplitude of each post-synaptic-current depends on the spike train that is impinging on the synapse [6], and causes facilitation and depression processes. The model was implemented using CSIM simulator [7].

### 3 Learning by Dispersion

Learning process drives a neural microcircuit to a desired state defined by configuration of sets of associations between stimuli and responses. This dynamical process begins with exploration of various network's states through modification of neuronal correlations. Two mechanisms which may be responsible for changing neuronal correlations are driving stimuli and neuromodulation by dopamine. Experiments on ex-vivo culture have shown [4], [8] that both mechanisms enhance changes in neuronal correlations by dispersing existing correlations, i.e. decorrelating previously acquired correlated activity. It is assumed that both mechanisms that cause decorrelation (dispersion) are mediated by a biophysical jittering of the synaptic strengths at polysynaptic level. This has led to the idea of modeling both mechanisms by what Eytan and Marom [10] coined as "Dispersing Mechanism".

The second phase of learning, the recognition, is responsible for "freezing" the NM state by stopping the exploration process. In recent years, a major effort was devoted to mapping of the behavioral concept of reward to neural mechanisms that change the functionality of a given NM based on its past performance [9]. The regulation of exploration process, driven by dopamine neuromodulation, is enabled by reward prediction error (RPE) signals. Dopamine neurons appear to emit RPE signal, as they are activated by rewards that are better than predicted, uninfluenced by rewards that occur exactly as predicted and depressed by rewards that are worse than predicted [9].

Learning by reward can occur by associating a stimulus or an action with a reward [3]. The learning is a function of RPE, defined by Schultz as a scalar difference in value (magnitude x probability) between a delivered (DR) and a predicted reward (PR):

$$RPE = DR - PR = f(\text{error in task execution}). \quad (2)$$

We apply a constant delivered reward, i.e.  $p(DR)=1$ , as long as there is any success in task execution. The predicted reward is a function of the system's previous success

in executing the task, i.e.  $PR=f(\text{success in task execution})$ . Since the performance of the system at the beginning of learning is lower than 100%; the predicted reward is lower than the delivered; dopamine neurons should be activated and emit dopamine to the system. We implement a feedback mechanism based on this reward mechanism in our Co-Evolutionary Learning of Liquid Architecture.

According to this "exploration and recognition" paradigm the dopamine jitters network's formed associations and thus enables state transition across the NM states space. In other words, the mechanism of jittering the synaptic efficacies, discovered by Eytan and Marom, is instrumental in avoiding trapping into a fixed point. When the best state dictated by the environment is found, the system reaches the recognition phase, and by stopping the dopamine emission, network's associations are "frozen".

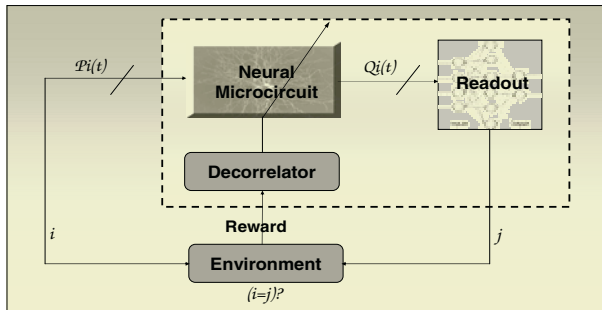
A mathematical model of this process, in which the synaptic efficacies are randomly jittered by regulation of RPE is formulated by:

$$\Delta W = \psi(W_0 \cdot K \cdot RPE), \tag{3}$$

wherein  $\psi$  is uniformly distributed between positive and the negative values of the argument,  $W_0$  is the previous value of the synaptic strength,  $K$  is a constant, and  $\Delta W$  is the change in the strength of the synapse. The model illustrates exploration and recognition processes, by dispersion of the NM synaptic strengths, regulated by the success in achieving the task of the overall system.

### 4 Co-evolutionary Learning in a Closed-Loop Framework

We propose a new closed-loop liquid architecture based on a NM as a core computational unit. The components of the system, illustrated in Fig.1, are NM, Readout function and a Decorrelator. In the open-loop setup the system is equivalent



**Fig. 1.** Closed-loop liquid architecture implemented in a classification task of time-varying inputs. NM is composed of 135 LIF neurons. Time-varying stimuli  $P_i(t)$  are transformed by NM onto liquid states,  $Q_i(t)$ , defined as firing patterns of NM at time  $t_0$ . Readout neurons are trained by supervised learning to identify the input applied to the system by transforming NM liquid states onto discrete value  $j$ . A feedback on system's performance is sent by the environment in form of reward signals to determine the RPE. Decorrelation, regulated by RPE, enables the co-evolution of the Readout and NM until a desired performance is obtained

to recently-proposed general theoretical model, called Liquid-State-Machine [1]. LSM presented a convenient framework for neural computations in real time for rapidly-time-varying continuous input functions. NM stores information about past inputs with high dimensional dynamics in its internal perturbations. Different input streams to the microcircuit cause different internal states (liquid states) of the system and enable the inputs to be separated. Liquid states of the NM are read by memoryless Readout. Readout is trained by supervised learning algorithms to transform high-dimensional transient liquid states of the NM onto desired outputs. After the Readout learns to define a needed class of equivalence, it can perform the learned task on novel inputs. The separation property (SP) requirement of the NM for functionality of LSM framework was illustrated in [2].

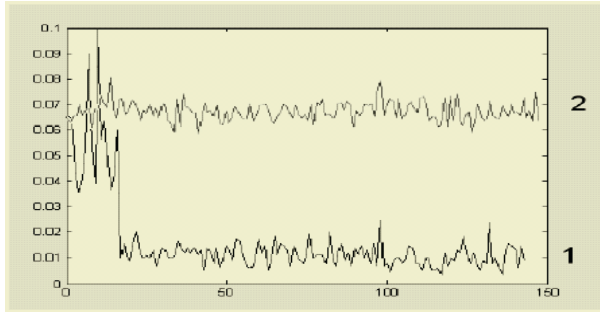
Within LSM framework the learning process is applied to the readout only, while the function of the NM as a generic filter is not changed. We propose an extended closed loop framework in which we apply to NM a previously-described learning-by-dispersion, driven by a feedback from the environment.

The overall framework is described in Fig. 1. Time-varying stimuli from the environment excite NM with a continuous input stream ( $P_i(t)$ ). At any time  $t_0$ , the internal liquid state of the microcircuit ( $Q_i(t_0)$ ) holds a substantial amount of information about recent inputs  $P_i(t < t_0)$ . Memoryless readout neurons are trained to map liquid states  $Q_i(t_0)$  onto discrete predefined values ( $j$ ). Discrete value  $j$  is a decision/action of the system in its environment. If the system succeeds in the task, i.e.  $i=j$  for classification task, reward signal is sent by the environment to the system. Reward signals, injected by the environment, are determined by system's performance and activate the Decorrelator by setting the value of RPE. Decorrelation mechanism modifies the NM synaptic strengths according to previously defined algorithm and drives the exploration phase of learning. During the exploration of NM states, the Readout is trained by supervised learning to transform the new formed liquid states onto system's output. When system's performance is sufficient, RPE is low, the recognition phase is reached and NM state is "frozen" by stopping the dispersion of the synaptic strengths.

We applied this co-evolutionary learning of the liquid architecture in general computational task of classification time varying stimuli. Randomly generated Poisson spike trains were injected to the system with a certain noise. Analysis of system's performance in a closed-loop versus an open-loop setup will be described in the next section.

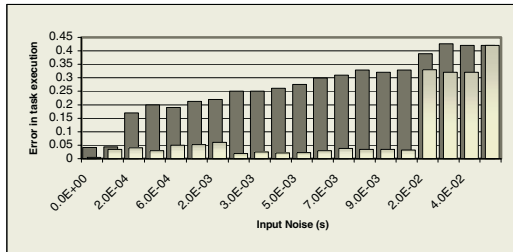
## 5 Computational Analysis of the Closed-Loop Framework

The added value of a closed-loop setup is examined in a general computational task of classification of a Poisson spike train. The error-in-task of the open-loop setup remains almost constant, since the optimal performance of the system is reached after the first supervised learning of the Readout is completed. In a closed-loop setup, in parallel to the supervised learning of the Readout, we apply a learning-by-dispersion of the NM. This co-evolutionary learning, of NM and Readout, generates an exploration process until the optimal performance of an overall system is obtained. The learning curve of a closed-loop versus an open-loop setup is illustrated in Fig. 2.



**Fig. 2.** Learning curve of a closed-loop (1) versus open-loop (2) setup implemented in classification task of time-varying stimuli

As the computational results depicted in Fig.2 indicate, the curve of co-evolutionary learning in a closed-loop setup does not converge gradually to the optimal point, since there is no *a-priori* knowledge of such a point. Various states of the NM are explored. This type of exploration is manifested by “jumps” characteristic of the learning curve. The exploration continues until a sufficient performance is obtained, at which time the NM state is “frozen”.



**Fig. 3.** Error-in-task for a closed-loop (*bright bars*) and open-loop (*dark bars*) setups versus noise in input

The closed-loop architecture exhibits superior performance, compared with the open-loop, insofar as the signal-to-noise ratio (SNR) is concerned. As the data depicted in Fig. 3 illustrates, the SNR of the closed-loop setup is by far lower than that characteristic of the open-loop setup. Whereas the error increases with noise level in the open-loop, as expected, in the closed-loop it even decreases, until at a certain noise level this advantage of the closed-loop breaks down. This abrupt shift in performance may be due to a network’s phase transition to a chaotic state. It is likewise possible that at a certain noise level network’s performance breaks down due to its saturation by the reward.

## 6 Voice Recognition Task

Co-evolutionary learning of liquid architecture was applied in a well-studied computational benchmark task for which data had been made publicly available – a speech recognition task [11]. The dataset consists of 500 input files: the words "zero", "one", "two", ..., "nine" are spoken by 5 different speakers, 10 times by each speaker. The task was to construct a network of I&F neurons that could recognize each of the spoken words.

The waveforms of the input sound were preprocessed by performing Fourier transform. Each of the frequency bands was composed of one or more of the following three events: onset (the start of the phase of significant energy), offset (the end of this phase), and peak (the first maximum of energy). The entire waveform is normalized to have maximum amplitude of 0.7, the sampling rate used in this case is 12000 samples/sec. The running average power and its second derivative are subsequently used in identification of events in the sound's spectrogram. This sound preprocessing converts the sound signal into a spatiotemporal sequence of events, suitable for recognition. Monosyllabic words are encoded into such sequences by retrieving features in different frequency bands in their spectrogram. Finally, sound waveform is converted into a list of 40 single events that are converted in turn into their respective times of occurrence.

Internet competition was publicized on this dataset for finding a network with the best classification performance. The best performance in this competition exhibited an error of 0.15, and was accomplished by a network with 800 pools of neurons [12]. The same task was solved by Maass, Natchlaeger and Markram in 2002 [2] using LSM framework with 145 I&F neurons. The average error in this classification task, achieved by this network, was 0.14.

We tested the co-evolutionary learning of the liquid architecture on the same task and the same dataset. A randomly chosen subset of 300 input files was used for training and the other 200 for testing. A previously described, randomly generated NM was implemented in a co-evolutionary learning of a closed-loop setup. The average error in this classification task, achieved by this closed-loop system, was 0.06.

## 7 Discussion

Liquid architectures embed interesting computational learning features in NM model. These emerging architectures are motivated by neurobiological findings obtained in experiments with neural culture. The common component of these liquid architectures is a core computational unit implemented by a generic heterogeneous model of NM. The proposed feedback mechanism adds a significant computational power to liquid architectures, illustrated for example in our simulations comparing the performance of the open and a closed-loop as a function of a noise level.

Liquid architecture exhibit a broad spectrum of solutions obtained under the condition of an identical task, manifested by its internal parameters. Co-evolutionary learning, illustrated in this study, provides a robust mechanism that exploits this computational feature, by randomly exploring the states space. Feedback mechanism regulates the exploration process until a sufficient solution is obtained. Converging

the ideas of liquid architecture, feedback mechanism and learning by exploration reveals a powerful paradigm for real-time, parallel computation in a rapidly varying environment.

## References

1. W. Maass, T. Natschläger, and H. Markram. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation*, 14(11):2531-2560 (2002).
2. W. Maass, T. Natschläger, and H. Markram. Computational models for generic cortical microcircuits. In J. Feng, editor, *Computational Neuroscience: A Comprehensive Approach*, chapter 18, pages 575-605. Chapman & Hall/CRC, Boca Raton, (2004)
3. Schultz W.: Neural coding of basic reward terms of animal learning theory, game theory, microeconomics and behavioural ecology. *Curr Opin Neurobiol.* Apr;14(2):139-47 (2004)
4. Danny Eytan, Amir Minerbi, Noam Ziv and Shimon Marom. Dopamine-induced Dispersion of Correlations Between Action Potentials in Networks of Cortical Neurons. *J. Neurophysiol.* 92:1817-1824 (2004)
5. G. Silberberg, A. Gupta and H. Markram: Stereotypy in neocortical microcircuits, *Trends Neurosci.* May;25(5):227-30 (2002)
6. M Tsodyks, K. Pawelzik, H. Markram , Neural networks with dynamic synapses. *Neural Computation* 10, 821-835 (1998)
7. T. Natschläger, H. Markram, and W. Maass. Computer models and analysis tools for neural microcircuits. In R. Kötter, editor, *A Practical Guide to Neuroscience Databases and Associated Tools*, chapter 9. Kluwer Academic Publishers (Boston), 2002. in press. (<http://www.lsm.tugraz.at>)
8. Goded Shahaf and Shimon Marom. Learning in networks of cortical neurons. *J. of Neuroscience* [volume 21(22):8782-8788, Nov. 15 (2001)
9. Schultz W. Predictive reward signal of dopamine neurons. *J Neurophysiol*, 80:1-27 (1998) Danny Eytan and Shimon Marom. Learning in Ex-Vivo
10. Developing Networks of Cortical Neurons *Progress in Brain Research*, Volume 147, "Development, Dynamics and Pathology of neural Networks", van Pelt et al, Editors, (2004)
11. J. Hopfield and C. Brody. The mus silicium (sonoran desert sand mouse) web page. Base: <http://moment.princeton.edu/~mus/Organism>
12. J. Hopfield and C. Brody. What is a moment? Transient synchrony as a collective mechanism for spatiotemporal integration. *Proc. Natl. Acad. Sci. USA*, 98(3):1282-1287, 2001

# Extended Sparse Nonnegative Matrix Factorization

Kurt Stadlthanner<sup>1</sup>, Fabian J. Theis<sup>1</sup>, Carlos G. Puntonet<sup>2</sup>,  
and Elmar W. Lang<sup>1</sup>

<sup>1</sup> Institute of Biophysics, University of Regensburg,  
93040 Regensburg, Germany

<sup>2</sup> Dept. Arq. y Téc. de Comp.,  
Universidad de Granada, 18071 Granada, Spain  
kusta@web.de

**Abstract.** In sparse nonnegative component analysis (sparse NMF) a given dataset is decomposed into a mixing matrix and a feature data set, which are both nonnegative and fulfill certain sparsity constraints. In this paper, we extend the sparse NMF algorithm to allow for varying sparsity in each feature and discuss the uniqueness of an involved projection step. Furthermore, the eligibility of the extended sparse NMF algorithm for blind source separation is investigated.

## 1 Matrix Factorization and Blind Source Separation

Often when it comes to analyze recorded observations, a suitable data representation is sought. One way of finding such a data representation is matrix factorization, where the  $m \times T$  observation matrix  $\mathbf{X}$  is decomposed into a  $m \times n$  matrix  $\mathbf{W}$  and a  $n \times T$  matrix  $\mathbf{H}$

$$\mathbf{X} = \mathbf{WH}. \quad (1)$$

Here, it is assumed that  $m$  observations, consisting of  $T$  samples, constitute the rows of  $\mathbf{X}$  and that  $m \leq n$ .

One application of matrix factorization is blind source separation (BSS), where the observations  $\mathbf{X}$  are known to be weighted sums of  $n$  underlying sources. If the sources form the rows of the  $n \times T$  matrix  $\mathbf{S}$ , and the element  $a_{ij}$  of the so-called mixing matrix  $\mathbf{A}$  is the weight with which the  $j$ -th source contributes to the  $i$ -th observation, then  $\mathbf{X}$  can be decomposed as

$$\mathbf{X} = \mathbf{AS}. \quad (2)$$

In BSS now, given only the matrix  $\mathbf{X}$ , a matrix factorization as in (1) is sought such that  $\mathbf{A}$  and  $\mathbf{S}$  are essentially equal to  $\mathbf{W}$  and  $\mathbf{H}$ , i.e. they are identical up to some scaling and permutation indeterminacies. Obviously, the BSS problem is highly underdetermined such that it can only be solved uniquely if additional assumptions on the sources or the mixing matrix are made.



### 1.1 (Sparse) Nonnegative Matrix Factorization

Unlike alternative matrix factorization techniques like principal component analysis (PCA), independent component analysis (ICA) or sparse component analysis (SCA), in nonnegative matrix factorization (NMF) it is strictly required that the entries of both matrices  $\mathbf{H}$  and  $\mathbf{W}$  are nonnegative, i.e. the data  $\mathbf{X}$  can be described using only additive components. Such a constraint has many physical realizations and applications, for instance in parts-based object decomposition [2].

Usually, the matrices  $\mathbf{W}$  and  $\mathbf{H}$  are determined in NMF by minimizing the least-squares (Euclidean) contrast

$$E(\mathbf{W}, \mathbf{H}) = \|\mathbf{X} - \mathbf{W}\mathbf{H}\|^2, \quad (3)$$

so (1) is required to hold true only approximately.

Apart from nonnegativity, a sparse representation of the data  $\mathbf{X}$  is often desired in matrix factorizations. Hence, Hoyer proposed in [1] a sparse NMF algorithm, in which the objective (3) is minimized under the constraints that all columns of  $\mathbf{W}$  and all rows of  $\mathbf{H}$  have common sparsenesses  $\sigma_W$  and  $\sigma_H$  respectively. Hereby, the sparseness  $\sigma(\mathbf{x})$  is quantized by the relation between the Euclidean norm  $\|\cdot\|_2$  and the 1-norm  $\|\mathbf{x}\|_1 := \sum_i |x_i|$  as follows

$$\sigma(\mathbf{x}) := \frac{\sqrt{n} - \|\mathbf{x}\|_1 / \|\mathbf{x}\|_2}{\sqrt{n} - 1} \quad (4)$$

if  $\mathbf{x} \in \mathbb{R}^n \setminus \{0\}$ . So  $\sigma(\mathbf{x}) = 1$  (maximal) if  $\mathbf{x}$  contains  $n - 1$  zeros, and it reaches zero if the absolute value of all coefficients of  $\mathbf{x}$  coincide.

### 1.2 Extended Sparse Non-negative Matrix Factorization

While constraining all rows of  $\mathbf{H}$  to have a common sparseness improves the results of NMF in image decomposition applications significantly [1], more general constraints are needed in BSS, where the sources to be recovered may have all different sparsenesses. To cope with this problem, we introduce an extended sparse NMF (esNMF) algorithm (cf. Alg. 1). It represents an extension of Hoyer's sparse NMF algorithm [1], in which for each of the  $n$  rows of  $\mathbf{H}$  an individual sparseness constraint  $\sigma_k$ ,  $k = 1 \dots n$ , may be provided while no assumptions concerning the sparseness of  $\mathbf{W}$  are made. Note, that the update steps of the matrices  $\mathbf{W}$  and  $\mathbf{H}$  in Alg. 1 (steps 3 and 6) are adopted from the sparse NMF algorithm such that we refer the reader to [1] and references therein for their derivations.

Note that the NMF algorithm becomes very dependent on the order in which differing sparsenesses are presented if the latter are chosen differently for the different sources. It turns out that the initialization of the matrices  $\mathbf{W}$  and  $\mathbf{H}$  determine the order in which the sources will be recovered. If this order differs from the order in which the sparsenesses are supplied within the algorithm, poor results can be expected only. This fact is accounted for in the algorithm by the special way matrix  $\mathbf{H}$  is initialized in step 2 as well as the adaptive assignment of the sparseness constraints of the columns of  $\mathbf{H}$  in steps 4 and 5.

**Input:** observation data matrix  $\mathbf{X}$ , sparseness constraints  $\sigma_k, k = 1, \dots, n$ ,  
 (assume  $\sigma_1 \leq \sigma_2 \dots \leq \sigma_n$  for simplicity)

**Output:** decomposition  $\mathbf{WH}$  of  $\mathbf{X}$  such that rows of  $\mathbf{H}$  fulfill given sparseness  
 constraints  $\sigma_k, k = 1 \dots n$

- 1 Initialize  $\mathbf{W}$  to a random non-negative matrix.
- 2 Initialize  $\mathbf{H}$  to a random non-negative matrix where all rows have sparseness 0.9.

**repeat**

- 3     Set  $\mathbf{H} \leftarrow \mathbf{H} - \mu_S \mathbf{W}^T (\mathbf{WH} - \mathbf{X})$ .
- 4     Determine the current sparseness  $\tilde{\sigma}_i, i = 1, \dots, n$ , of each row of  $\mathbf{H}$ . Find permutation  $\{j_1, \dots, j_n\}$  of  $\{1, \dots, n\}$  such that  $\tilde{\sigma}_{j_1} \leq \tilde{\sigma}_{j_2} \leq \dots \leq \tilde{\sigma}_{j_n}$ .
- 5     Project the  $j_k$ -th row of  $\mathbf{H}$  such that it fulfills the sparseness constraint  $\sigma_k, k = 1, \dots, n$  (cf. Alg. 2).
- 6     Set  $\mathbf{W} \leftarrow \mathbf{W} \otimes (\mathbf{XH}^T) \oslash (\mathbf{WHH}^T)$ .

**until** convergence;

Note: in the last step  $\otimes$  and  $\oslash$  symbolize elementwise multiplication and division respectively.

**Algorithm 1:** Extended Sparse NMF Algorithm

We try to circumvent this problem by initializing the rows of  $\mathbf{H}$  such that they are all of equal sparsity and unbiased towards the original sources. The latter condition can be met by demanding that the rows of  $\mathbf{H}$  have a much higher sparseness than the original sources. In our experiments, we always used an initial sparseness of 0.9 for the rows of  $\mathbf{H}$ .

During the first iteration  $\mathbf{H}$  is updated under the assumption that the randomly initialized matrix  $\mathbf{W}$  is the actual mixing matrix. This leads to a first estimate of the original sources of which the sparsenesses can be computed. Even if this estimate is still very rough a comparison between the original and the estimated sparsenesses (cf. step 4 in Alg. 1) can already reveal the order in which the sources will be recovered. Hence, we can now project the estimated sources such that they fulfill their corresponding sparseness constraints (step 5) using the projection operator as will be described in Sec. 2, Alg. 2. Finally, in step 6 of esNMF, the estimated mixing matrix  $\mathbf{W}$  is updated under the assumption that  $\mathbf{H}$  contains the actual sources.

## 2 Sparse Projection

The sparse NMF algorithm uses a projection step as follows: Given  $\mathbf{x} \in \mathbb{R}^n$  and fixed  $\lambda_1, \lambda_2 > 0$ , find  $\mathbf{s}$  such that

$$\mathbf{s} = \operatorname{argmin}_{\|\mathbf{s}\|_1 = \lambda_1, \|\mathbf{s}\|_2 = \lambda_2, \mathbf{s} \geq 0} \|\mathbf{x} - \mathbf{s}\|_2. \tag{5}$$

We want to solve problem (5) by projecting  $\mathbf{x}$  onto

$$M := \{\mathbf{s} \mid \|\mathbf{s}\|_1 = \lambda_1\} \cap \{\mathbf{s} \mid \|\mathbf{s}\|_2 = \lambda_2\} \cap \{\mathbf{s} \geq 0\}. \tag{6}$$

In order to solve equation (5),  $\mathbf{x}$  has to be projected onto a point adjacent to it in  $M$ :

**Definition 1.** A point  $\mathbf{p} \in M \subset \mathbb{R}^n$  is called adjacent to  $\mathbf{x} \in \mathbb{R}^n$  in  $M$ , in symbols  $\mathbf{p} \triangleleft_M \mathbf{x}$  or shorter  $\mathbf{p} \triangleleft \mathbf{x}$ , if  $\|\mathbf{x} - \mathbf{p}\|_2 \leq \|\mathbf{x} - \mathbf{q}\|_2$  for all  $\mathbf{q} \in M$ .

### 2.1 Indeterminacies

First we will discuss the question of existence of projection points  $\mathbf{p}$ . Obviously  $\mathbf{p}$  cannot exist if  $\mathbf{x}$  lies in the closure of  $M$  (i.e. ‘touches’ it) without being an element of  $M$ . Indeed this is the only obstruction to existence as the following remark shows. Proofs are not given due to space limitations.

*Remark 1 (Existence).* If  $M$  is closed and nonempty, then for every  $\mathbf{x} \in \mathbb{R}^n$  there exists  $\mathbf{p} \in M$  with  $\mathbf{p} \triangleleft \mathbf{x}$ .

In order to study uniqueness of the projection, we define an exception set as follows:

**Definition 2.** Let  $\mathcal{X}(M) := \{\mathbf{x} \in \mathbb{R}^n \mid \text{there exists more than one point adjacent to } \mathbf{x} \text{ w.r.t. } M\} = \{\mathbf{x} \in \mathbb{R}^n \mid \#\{\mathbf{p} \in M \mid \mathbf{p} \triangleleft \mathbf{x}\} > 1\}$  denote the exception set of  $M$ .

So the exception set contains the set of points from which no unique projection is possible. We want to show that this set vanishes or is at least very small.

Obviously the exception set of an affine linear hyperspace vanishes. Indeed, we can prove more generally that the exception set of a convex set is empty. In general however, we cannot expect  $\mathcal{X}(M)$  to vanish altogether. However we can show that in practical applications we can easily neglect it:

**Theorem 1 (Uniqueness).**  $\text{vol}(\mathcal{X}(M)) = 0$ .

This theorem states that the Lebesgue measure of the exception set is zero i.e. that it does not contain any open ball. In other words, if  $\mathbf{x}$  is drawn from a continuous probability distribution on  $\mathbb{R}^n$ , then  $\mathbf{x} \in \mathcal{X}(M)$  with probability 0. In practice, this proves uniqueness because samples are usually drawn from continuous distributions.

The theorem follows from the fact that if  $\mathbf{x}$  is some point of the exception set of  $M$ , any point lying on a line between  $\mathbf{x}$  and one of its projections does not again lie in the exception set.

### 2.2 Projection Algorithm

Now let  $M$  be defined by equation 6. Hoyer [1] essentially proposes algorithm 2 to project a given vector  $\mathbf{x}$  onto  $\mathbf{p} \in M$  such that  $\mathbf{p} \triangleleft \mathbf{x}$ . The algorithm iteratively detects  $\mathbf{p}$  by first satisfying the 1-norm condition and then the 2-norm condition.

```

Input: vector $\mathbf{x} \in \mathbb{R}^n$, norm conditions λ_1 and λ_2
Output: closest non-negative \mathbf{s} with $\|\mathbf{s}\|_i = \lambda_i$

1 Set $\mathbf{r} \leftarrow \mathbf{x} + (\|\mathbf{x}\|_1 - \lambda_1/n)\mathbf{e}$ with $\mathbf{e} = (1, \dots, 1)^\top \in \mathbb{R}^n$.
2 Set $\mathbf{m} \leftarrow (\lambda_1/n)\mathbf{e}$.
3 Set $\mathbf{s} \leftarrow \mathbf{m} + \alpha(\mathbf{r} - \mathbf{m})$ with $\alpha > 0$ such that $\|\mathbf{s}\|_2 = \lambda_2$.
 if exists j with $s_j < 0$ then
4 | Fix $s_j \leftarrow 0$.
5 | Remove j -th coordinate of \mathbf{x} .
6 | Decrease dimension $n \leftarrow n - 1$.
7 | goto 1.
end

```

**Algorithm 2:** Sparse projection

It terminates if the constructed vector is already positive; otherwise a negative coordinate is selected, set to zero and the search is continued in  $\mathbb{R}^{n-1}$ .

The algorithm terminates after maximally  $n - 1$  iterations. Indeed, it finds the correct projections if  $\mathbf{x} \notin \mathcal{X}(M)$  as the following theorem shows. It is proved by showing that in each step the new estimate has  $\mathbf{p}$  as closest point in  $M$ .

**Theorem 2 (Sparse projection).** *Given  $\mathbf{x} \geq 0$  such that  $\mathbf{x} \notin \mathcal{X}(M)$  and let  $\mathbf{p} \in M$ ,  $\mathbf{p} \triangleleft \mathbf{x}$  be its projection onto  $M$ . If  $\mathbf{s}$  is constructed by line 3 of algorithm 2, then  $\mathbf{p} \triangleleft \mathbf{s}$  and  $\mathbf{s} \notin \mathcal{X}(M)$ . Furthermore, the algorithm terminates at  $\mathbf{s} = \mathbf{p}$ .*

### 3 Simulations

#### 3.1 Adaptive Sparseness Assignment

In this section we show that the initialization of the matrix  $\mathbf{W}$  actually determines the order in which the sources are estimated, and demonstrate that the esNMF algorithm may automatically detect this order. Furthermore, it will be shown that this automatic detection is pivotal for the esNMF algorithm to find good solutions independent of the order in which the given sparsenesses are supplied to it.

For this purpose we compared our esNMF algorithm with a second, simplified version of it, called esNMF<sub>fix</sub>, in which the matrix  $\mathbf{H}$  is initialized by a random nonnegative matrix and in which the given sparseness constraints are applied to the rows of  $\mathbf{H}$  in the fixed order in which they were supplied to the algorithm.

We evaluated the performance of both algorithms by applying them to a toy BSS problem in which three nonnegative sources with different sparsenesses should be recovered from three observations. The three sources  $s_i$ ,  $i = 1, \dots, 3$ , consisted of 3000 data points and had sparsenesses of  $\sigma_1 = 0.61$ ,  $\sigma_2 = 0.47$  and  $\sigma_3 = 0.23$  respectively. The observation matrix  $\mathbf{X}$  was generated by multiplying the source matrix  $\mathbf{S}$  by a random nonnegative  $3 \times 3$  matrix  $\mathbf{A}$  as in (2).

**Table 1.** Comparison of the esNMF<sub>fix</sub> and the esNMF algorithm. The *Perm* column shows in which order the sparsenesses were provided to the algorithms. The *CTE<sub>fix</sub>* and *CTE* columns display the average of the 100 CTE's determined per experiment. The *CC<sub>fix</sub>* and *CC* columns depict the average of the 300 CCs determined in each experiment. The numbers in the *Fail* column show how often the esNMF algorithm could not recover the sources and the mixing matrix. The columns *CTE<sub>fix</sub>* and *CC<sub>fix</sub>* belong to the esNMF<sub>fix</sub> algorithm, the *CTE*, the *CC* and the *Fail* columns belong to the esNMF algorithm

| Perm | CTE <sub>fix</sub> | CC <sub>fix</sub> | CTE  | CC   | Fail |
|------|--------------------|-------------------|------|------|------|
| 123  | 0.00               | 1.00              | 0.05 | 0.99 | 4 %  |
| 132  | 7.33               | 0.74              | 0.12 | 0.99 | 10 % |
| 213  | 1.21               | 0.94              | 0.09 | 1.00 | 7 %  |
| 231  | 4.62               | 0.79              | 0.06 | 1.00 | 5 %  |
| 312  | 7.35               | 0.75              | 0.09 | 1.00 | 7 %  |
| 321  | 3.17               | 0.85              | 0.01 | 1.00 | 1 %  |

In this case, there are 6 possible orders in which the given sparsenesses  $\sigma_i$ ,  $i = 1, \dots, 3$ , can be provided to the algorithms. For each of these orders both algorithms were used to recover the original sources as well as the mixing matrix  $\mathbf{A}$ . As we wanted to show that the initialization of the matrix  $\mathbf{W}$  determines the order in which the sources are recovered, we initialized it in both algorithms by the original mixing matrix  $\mathbf{A}$ . Furthermore, as both algorithms are known to be prone to local minima, the algorithms were run 100 times for each sparseness order, each time with a different initialization of the matrix  $\mathbf{H}$ .

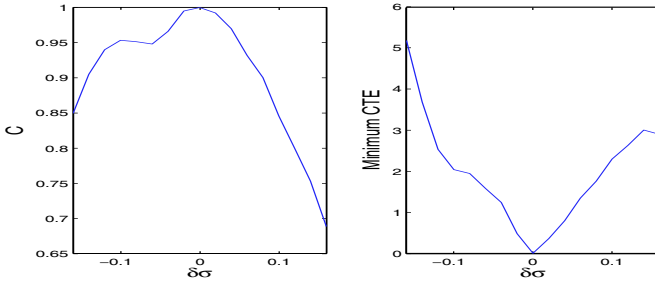
The results of both algorithms were compared using the crosstalking error (CTE) between the original and the estimated mixing matrix as well as by the correlation coefficients (CC) between the original and the recovered sources.

As can be seen in Tab. 1, the esNMF<sub>fix</sub> algorithm only succeeded in recovering the original sources and the mixing matrix when the sparsenesses were provided in the same order as the original sources. If we initialized the matrix  $\mathbf{W}$  in esNMF<sub>fix</sub> by a matrix  $\tilde{\mathbf{A}}$  which was equal to  $\mathbf{A}$  up to a permutation of two of its columns, then the order of the corresponding sparsenesses had to be permuted as well in order to obtain satisfying results. Hence, we conclude that the initialization of the matrix  $\mathbf{W}$  actually determines the order in which the sources are recovered.

On the other hand the esNMF algorithm only failed in maximally 10% of its runs regardless of the order in which the sparsenesses were provided. Apart from these failures, it always lead to CTEs smaller then  $10^{-7}$  as well as to CCs higher than 0.9999. This confirms the eligibility of the proposed adaptive assignment method.

### 3.2 Comparison of NMF and esNMF

As NMF has already been used very successfully in other matrix factorization tasks, we now apply it to BSS with sparseness constraints and compare the



**Fig. 1.** The dependence of  $C$  (see text) and the CTE on the sparseness presets

results with those obtained by our esNMF algorithm. For our simulations we have used the multiplicative NMF algorithm from [1].

Our simulations consisted of 100 single experiments. In each experiment three nonnegative random sources with sparsenesses of  $\sigma_1 = 0.23$ ,  $\sigma_2 = 0.47$  and  $\sigma_3 = 0.62$  were generated and then mixed by a nonnegative random  $3 \times 3$  mixing matrix  $\mathbf{A}$ . Then, an observation matrix  $\mathbf{X}$  was computed as in (2).

In each of the experiments the NMF as well as esNMF algorithm were used to recover the sources and the results were compared using the CTE between the estimated and the original mixing matrix as well as the CCs between the estimated and the original sources.

It turned out that the esNMF algorithm outperformed the NMF algorithm in most cases. To quantify the results, we defined the threshold  $\tau$ . If the lowest CC between a recovered and its corresponding original source was lower than  $\tau$  then we classified the experiment as failure, otherwise as success.

If  $\tau$  was set to 0.95, then NMF succeeded in 11% of the runs while the esNMF algorithm recovered the sources successfully in 20% of the runs. If  $\tau$  was increased to 0.98 then the esNMF algorithm still recovered the sources successfully in 19% of its runs, NMF, however, only in 6%. Finally, only the esNMF algorithm managed to recover sources with a CC larger than 0.9999 with the original sources (16% of all runs).

Similar results were obtained for the CTEs between the estimated and the original mixing matrices. The lowest value of  $10^{-7}$  was again achieved by the esNMF algorithm while the NMF algorithm did not lead to CTEs smaller than 0.5. Note, that again CTEs smaller than the latter value were achieved by esNMF in 18% percent of its runs.

Hence, the esNMF algorithm does not only lead to qualitatively better results than the NMF algorithm but is also much more robust against its initializations.

### 3.3 Sparseness Presets and Robustness

In real life experiments it is usually difficult to estimate the sparsenesses of the sources a priori. Accordingly, we have tested the esNMF algorithm for its robustness against sparseness presets  $\hat{\sigma}_k$  which deviated from the actual sparsenesses  $\sigma_k$  of the sources.

For the simulations we used three nonnegative sources with sparsenesses of  $\sigma_1 = 0.61$ ,  $\sigma_2 = 0.47$  and  $\sigma_3 = 0.23$  respectively and a nonnegative  $3 \times 3$  mixing matrix  $\mathbf{A}$ . Both the sources and the mixing matrix could be recovered perfectly ( $\text{CC} > 0.9999$ ,  $\text{CTE} < 1.8 \cdot 10^{-7}$ ) if the correct sparsenesses were provided to the esNMF algorithm.

Then, we shifted the sparseness presets  $\hat{\sigma}_k$  from their correct values  $\sigma_k$  by adding an error  $\delta\sigma$  ( $\hat{\sigma}_k = \sigma_k + \delta\sigma$ ,  $k = 1, \dots, n$ ), where  $\delta\sigma$  was varied from  $-0.16$  to  $0.16$  in steps of  $0.02$ . For each value of  $\delta\sigma$  the esNMF algorithm was run 20 times with different initializations for the matrices  $\mathbf{W}$  and  $\mathbf{H}$ . Now, for each of the 20 runs the minimal correlation coefficient  $c_j^{\min}(\delta\sigma)$ ,  $j = 1, \dots, 20$ , between the estimated sources and their original counterparts was determined. Furthermore, the CTEs between the estimated and the original mixing matrices were computed. The minimum of these 20 CTEs as well as the maximum value  $C$  of  $c_j^{\min}(\delta\sigma)$ ,  $j = 1, \dots, 20$ , are depicted in Fig. 1 for each  $\delta\sigma$ .

While the CTE seemed to be slightly more sensitive to too small presets, the sources could still be recovered well down to a  $\delta\sigma$  of about  $-0.1$ . Hence, we would suggest to use rather too low than too high presets for the sparsenesses in real life applications. Generally, a  $\delta\sigma$  between  $-0.5$  and  $0.5$  should still lead to acceptable results.

## 4 Conclusion

We have presented an extension of the sparse NMF algorithm [1] and have investigated its eligibility for BSS. Furthermore, the uniqueness of the projection operator was shown. One drawback of the presented method is that the sparsenesses of the sources must be known a priori. We have shown, however, that an exact preknowledge of these sparsenesses is not necessary to obtain good results.

## References

1. P.O. Hoyer. Non-negative matrix factorization with sparseness constraints. *Journal of Machine Learning Research*, 5:1457–1469, 2004.
2. D.D. Lee and H.S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 40:788–791, 1999.

# Using a Mahalanobis-Like Distance to Train Radial Basis Neural Networks

J.M. Valls, R. Aler, and O. Fernández

Carlos III University - Computer Science Department,  
Avenida de la Universidad, 30 - 28911 Leganés (Madrid), Spain  
jvalls@inf.uc3m.es

**Abstract.** Radial Basis Neural Networks (RBNN) can approximate any regular function and have a faster training phase than other similar neural networks. However, the activation of each neuron depends on the euclidean distance between a pattern and the neuron center. Therefore, the activation function is symmetrical and all attributes are considered equally relevant. This could be solved by altering the metric used in the activation function (i.e. using non-symmetrical metrics). The Mahalanobis distance is such a metric, that takes into account the variability of the attributes and their correlations. However, this distance is computed directly from the variance-covariance matrix and does not consider the accuracy of the learning algorithm. In this paper, we propose to use a generalized euclidean metric, following the Mahalanobis structure, but evolved by a Genetic Algorithm (GA). This GA searches for the distance matrix that minimizes the error produced by a fixed RBNN. Our approach has been tested on two domains and positive results have been observed in both cases.

## 1 Introduction

Radial Basis Neural Networks (RBNN) [1, 2] are originated from the use of radial basis functions, in the solution of the real multivariate interpolation problem [3, 4]. As the Multilayer perceptron (MLP) they can approximate any regular function [5]. Due to its local behavior and to the linear nature of its output layer, their training is faster than MLP training [5] and this fact makes them useful for a wide variety of applications. The most used radial basis functions are Gaussian functions, defined by equation 1.

$$\phi_m(x_k) = e^{-\frac{\|c_m - x_k\|^2}{2\sigma_m^2}} \quad (1)$$

Where  $\phi_m(x_k)$  represents the activation function for neuron  $m$  when an input pattern  $x_k$  is presented. The vector  $c_m$  is the center of the neuron  $m$ , and  $\sigma_m$  is its deviation or width.

One of the problems of RBNN is the symmetrical nature of their activation function, making that the activation of a neuron when a pattern is presented, only depends on the euclidean distance from this pattern to the neuron center



regardless of the importance of each attribute. This could be solved by altering the metric used in the activation function.

The Mahalanobis distance is a metric used in Statistics in order to normalize different attributes and take into account the correlation among them. This distance is computed according to expression 2.

$$d_{ij} = [(\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{S}^{-1} (\mathbf{x}_i - \mathbf{x}_j)]^{1/2} = [(\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{M}^T \mathbf{M} (\mathbf{x}_i - \mathbf{x}_j)]^{1/2} \quad (2)$$

Where  $d_{ij}$  is the Mahalanobis distance between vectors  $\mathbf{x}_i$  and  $\mathbf{x}_j$ ,  $S$  is the variance-covariance matrix of all vectors in the data set and  $M$  is the so-called Mahalanobis matrix[6,7,8]. This distance can be used to improve prediction accuracy in those learning systems that use distances [9]. However, the Mahalanobis distance is independent of the learning system used and of the error produced on the training data, because it is computed from the points in the data set only (more specifically, it is computed from the variance-covariance matrix).

In this paper, we propose to use a Mahalanobis-like distance in the activation function of the RBNN so that different attributes are treated differently according to their relevance. But instead of computing the distance using the variance-covariance matrix  $S$ , a matrix will be built in order to minimize the error of the network. This will be achieved by a genetic algorithm [10] whose individuals are generalized euclidean distance matrices and whose fitness function depends on the prediction accuracy attained by the network using the matrix.

## 2 Description of the Method

In this paper we use a standard Genetic Algorithm (GA) [10] to evolve distance matrices. A genetic algorithm is a kind of heuristic search. The algorithm maintains a set of candidate solutions (or population of individuals) and applies the search operators on them (also called genetic operators: mutation and crossover). The search is guided by a heuristic (or fitness) function. We have used a generational Genetic Algorithm with elitism and tournament selection. Matrices in the individuals are coded by representing each of their components in binary format. The fitness function is computed by training a RBNN on a set of training data and determining the training error. Thus, the GA tries to find the distance matrix that minimizes the RBNN training error. The number of hidden neurons is fixed from the start.

In order to determine the appropriate  $M$  matrix by using GA, individuals must be properly encoded. We have chosen matrix  $M$  to be symmetrical, to ensure that  $M^T M$  is invertible, although in the future this restriction could be removed in favor of less restrictive conditions. In that case, only the diagonal and the upper half of the matrix coefficients must be encoded to a binary representation in order to build the chromosome of the individual. Each matrix element is a real number that must be encoded to a binary representation with a fixed number of bits, following a fixed-point representation with a single bit

for the sign. Hence, the chromosome is a string of bits formed by the binary representation of each matrix element belonging to the diagonal or the upper half of the matrix. I.e. if the matrix is

$$\mathbf{M} = \begin{pmatrix} \mathbf{m}_{11} & \mathbf{m}_{12} & \dots & \mathbf{m}_{1d} \\ m_{21} & \mathbf{m}_{22} & \dots & \mathbf{m}_{2d} \\ \vdots & \vdots & \vdots & \vdots \\ m_{d1} & m_{d2} & \dots & \mathbf{m}_{dd} \end{pmatrix}$$

The corresponding string chromosome will be:

$$\{B(m_{11}), B(m_{12}), \dots, B(m_{1d}), B(m_{22}), B(m_{23}), \dots, B(m_{dd})\}$$

where  $B(m_{ij})$  is the binary representation of  $m_{ij}$ .

Each individual represents a matrix  $M$  that will determine the distance function to be used for the neurons activation (see Eq. 2). The goal of this work consists of improving the accuracy or the RBNN; hence, if the network error is small it means that the corresponding distance function is good; thus, the individual representing the  $M$  matrix must have a big fitness value. The fitness function chosen in this work is given in equation 3.

$$fitness = -\log_2 E \quad (3)$$

where  $E$  is the mean squared error committed by the network on the training data. It has been chosen so that fitness increases when error decreases. This function also manages to amplify differences between individuals whose error is close to zero. This is important to increase evolutionary pressure in the latest stages of GA-evolution, when all individuals are very good.

In the next the sequential structure of the proposed method is summarized.

- 
1. Create the initial population. A set of random chromosomes is generated. These chromosomes represent different distance functions to be used in the Radial Basis Functions of the networks.
  2. Evaluate the fitness of each element of the current population. In order to perform this point, RBNN with a fixed number of hidden neurons are trained using the distance function determined by each individual of the population. Training errors of these networks are used to calculate the fitness of each individual.
  3. Apply genetic operators to the population in order to create the next generation.
  4. If the number of generations is lower than the maximum, go to step 2
  5. Return the highest fitness matrix
- 

### 3 Empirical Evaluation

The purpose of this section is to validate empirically our approach. Two sets of experiments will be carried out. First, a synthetic domain, where the solution is known, will be posed to the system. Next, the well-known Mackey-Glass regression problem will be tested.

**Table 1.** Parameters of the Genetic Algorithm for the gaussian domain

|                       |      |
|-----------------------|------|
| Generations           | 30   |
| Tournament size       | 2    |
| Population size       | 20   |
| Elitism               | 1    |
| Crossover probability | 0.6  |
| Mutation probability  | 0.03 |

### 3.1 Synthetic Domain

This domain follows a bi-variate gaussian shape ( $\mu = (0.5, 0.5)$ ,  $\sigma^2 = 0.002$ ). However, instead of the euclidean distance, a generalized euclidean distance with matrix  $M$  will be used instead (see Eq. 2 and matrix 4). In euclidean space, the result is a rotated and stretched gaussian (i.e. non-symmetrical). In short, the goal is to approximate the function given by Eq. 5, where  $M$  is given by Eq. 4.

$$M = \begin{pmatrix} 0.2 & 0.75 \\ 0.75 & 1.0 \end{pmatrix} \tag{4}$$

$$e^{-\frac{(\mathbf{x}-0.5)^T M^T M (\mathbf{x}-0.5)}{2 \times 0.002}} \tag{5}$$

Obviously, a RBNN with a single neuron centered on (0.5, 0.5) will not be able to correctly learn this function, because the activation function uses a euclidean distance which is symmetric. But our GA should be able to learn the matrix  $M$  used to generate the domain. In order to get a proof-of-concept using this simple problem, we trained our system using a single neuron centered on (0.5, 0.5) with a  $\sigma^2 = 0.008$  (four times the  $\sigma^2$  used to generate the domain). The GA was run using the parameters shown in Table 1.

In addition, 3 bits were used for the integer part, and 5 bits for the fractionary part. Only symmetric matrices were allowed. After 30 generations, the following matrix was obtained (see Eq. 6), which approached very well the function (it achieved a  $5.867 \times 10^{-5}$  error).

$$M = \begin{pmatrix} 0.46875 & 1.50000 \\ 1.50000 & 2.00000 \end{pmatrix} \tag{6}$$

Matrix 6 does not match matrix 4 (the one used to generate the domain), although it can be seen that their components approximately double the ones in the domain matrix. In any case, it is the activation functions that must be the same, in order for the 1 neuron RBNN to approximate perfectly the function. That is, the following equality has to be satisfied (see Eq. 7):

$$(1/\sigma_1^2)[M_1^T M_1] = (1/\sigma_2^2)[M_2^T M_2] \tag{7}$$

where  $\sigma_1^2$  and  $M_1$  refer to the parameters used to generate the domain,  $\sigma_2^2$  is the variance of the neuron, and  $M_2$  is the matrix obtained by the genetic algorithm. This equality is almost satisfied, as Eq. 8 shows.

$$\begin{pmatrix} 301.25 & 450 \\ 450 & 781.25 \end{pmatrix} \sim \begin{pmatrix} 283.2 & 421.8 \\ 421.8 & 757.8 \end{pmatrix} \quad (8)$$

It is interesting to remark that even though the  $\sigma^2$  of the neuron (0.008) was not the same than the one used to generate the domain ( $\sigma^2 = 0.002$ ), the GA managed to fit the domain function by appropriately scalating the components of the evolved matrix.

### 3.2 The Mackey-Glass Domain

The Mackey-Glass time series is widely regarded as a benchmark for comparing the generalization ability of RBNN[11],[12],[13]. The task for the RBNN is to predict the value of the time series at point  $x[t + 50]$  from the earlier points ( $x[t], x[t - 6], x[t - 12], x[t - 18]$ ). It is a chaotic time series created by Eq. 9:

$$\frac{dx(t)}{dt} = -bx(t) + a \frac{x(t - \tau)}{1 + x(t - \tau)^{10}} \quad (9)$$

1474 patterns were generated for the Mackey-glass series, and values were normalized in (0,1). First, we ran some preliminary experiments in order to determine the number of neurons required. The minimum error was obtained with about 25 neurons. Also, these preliminary tests showed that 400 learning cycles and a 0.002 learning rate were reasonable values in this domain.

We tested two configurations of the system: allowing only diagonal matrices, and allowing general symmetrical matrices. The first case is equivalent to have a generalized euclidean distance, where every attribute is weighted by a factor. The most relevant attributes will be weighted by a larger number (see Eq. 10). The second case (the symmetrical matrix) can also consider correlations between attributes. Table 2 summarizes the parameters used. Two bits were used for the integer part, and three for the fractionary part.

$$d(A, B) = \sqrt{\sum_{i=0}^{i=N} C_i * (A_i - B_i)^2} \quad (10)$$

Table 3 displays the results comparing performance of a RBNN using an euclidean distance and evolved distances. 5-fold crossvalidation results are shown for both a diagonal matrix and a general symmetric matrix. Improvements of

**Table 2.** Parameters of the Genetic Algorithm for the Mackey-Glass problem

|                       |      |
|-----------------------|------|
| Generations           | 50   |
| Tournament size       | 2    |
| Population size       | 15   |
| Elitism               | 1    |
| Crossover probability | 0.7  |
| Mutation probability  | 0.01 |

**Table 3.** Comparison of results between euclidean and evolved distances (5-fold cross-validation)

| Distance used       | Error    | Improvement (%) |
|---------------------|----------|-----------------|
| RBNN euclidean      | 0.013981 |                 |
| RBNN GA diagonal    | 0.010337 | 26%             |
| RBNN euclidean      | 0.015789 |                 |
| RBNN GA symmetrical | 0.014835 | 6%              |

26% and 6% (respectively) can be observed. In this domain, using a diagonal matrix is better than using a symmetrical matrix.

In order to get a better understanding of results in this domain, we observed the values of the components of the matrices evolved by the GA. As we used a 5-fold crossvalidation procedure, 5 matrices were evolved. It can be observed that none of the components outside the diagonal are significantly different than 0 (taking into account the 5 folds, the median for these components is very close to 0). This means that for this domain, a symmetrical matrix does not give any advantage and that only a diagonal matrix is required. It can also be observed in both the diagonal and symmetrical matrices, that attributes 1 and 4 get larger components in the matrix than attributes 2 and 4. This means that in this domain, variables 1 and 4 are more relevant for the regression problem.

## 4 Conclusions

One of the problems of RBNN is the symmetrical nature of their activation function: the activation of a neuron only depends on the euclidean distance from the input pattern to the neuron center, without taking into account the importance of different attributes. This problem can be solved by altering the metric used in the activation function. The learning method presented in this work uses a Mahalanobis-like distance function: instead of computing the Mahalanobis matrix from the variance-covariance matrix of all vectors in the data set, it is determined in such a way that minimizes the error of the network. This is achieved by a genetic algorithm whose individuals are generalized distance matrices and whose fitness function depends on the prediction accuracy attained by the network.

Our GA approach has been tested on two domains: a simple synthetic one and the Mackey-Glass time series. It has been shown that using both diagonal and symmetrical evolved matrices improves prediction accuracy over a purely euclidean distance.

In the future, we would like to test our approach using problems with a larger dimensionality, to study the effect of evolving large matrices. We would also like to explore other evolutionary approaches, like evolution strategies, which are perhaps more suited to evolve structures with real numbers. Finally, it would be interesting to understand the characteristics of domains where a symmetrical matrix is better than a purely diagonal one.

## References

1. J.E. Moody and C. Darken. Fast learning in networks of locally tuned processing units. *Neural Computation*, 1:281–294, 1989.
2. Joydeep Ghosh and Arindam Nag. *An Overview of Radial Basis Function Networks*. R.J. Howlett and L.C. Jain (Eds). Physica Verlag, 2000.
3. D.S. Broomhead and D. Lowe. Multivariable functional interpolation and adaptive networks. *Complex Systems*, 2:321–355, 1988.
4. M. Powell. The theory of radial basis function approximation in 1990. *Advances in Numerical Analysis*, 3:105–210, 1992.
5. J. Park and I. W. Sandberg. Universal approximation and radial-basis-function networks. *Neural Computation*, 5:305–316, 1993.
6. C.G. Atkenson, A.W. Moore, and S. Schaal. Locally weighted learning. *Artificial Intelligence Review*, 11:11–73, 1997.
7. J. T. Tou and R. C. Gonzalez. *Pattern Recognition Principles*. Addison-Wesley, 1974.
8. S. Weisberg. *Applied Linear Regression*. New York: John Wiley and Sons, 1985.
9. F. Babiloni, L. Bianchi, F. Semeraro, J. del R-Millan, J. Mourino, A. Cattini, S. Salinari, M.G. Marciari, and F. Cincotti. Mahalanobis distance-based classifiers are able to recognize eeg patterns by using few eeg electrodes. In *Engineering in Medicine and Biology Society, 2001. Proceedings of the 23rd Annual International Conference of the IEEE*, volume 1, pages 651–654, 2001.
10. John H. Holland. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, 1975.
11. A. Leonardis and H. Bischof. An efficient mdl-based construction of rbf networks. *Neural Networks*, 11:963–973, 1998.
12. M. J. L. Orr. Introduction to radial basis neural networks. *Technical Report. Centre for Cognitive Science, University of Edinburgh*, 1996.
13. L. Yingwei, N. Sundararajan, and P. Saratchandran. A sequential learning scheme for function approximation using minimal radial basis function neural networks. *Neural Computation*, 9:461–478, 1997.

# Robustness of Radial Basis Functions<sup>\*</sup>

Ralf Eickhoff and Ulrich Rückert

Heinz Nixdorf Institute,  
System and Circuit Technology,  
University of Paderborn, Germany  
{eickhoff, rueckert}@hni.upb.de

**Abstract.** Neural networks are intended to be used in future nanoelectronics since these architectures seem to be robust against malfunctioning elements and noise. In this paper we analyze the robustness of radial basis function networks and determine upper bounds on the mean square error under noise contaminated weights and inputs.

## 1 Introduction

Neural networks are used as function approximators for continuous functions [1, 2]. Especially, radial basis function networks are utilized to perform a local approximation of an unknown function specified by a set of test data. The main reason why neural networks are used for this purpose is the adaptability of the network due to the learning process. Moreover, the networks seem to be fault tolerant [2] and robust against malfunctioning neurons [3] modeled as Stuck-At faults and noise corrupted weights and inputs.

Digital or analog implementations of neural networks have always to face malfunctioning elements [4] especially in future nanoelectronic realizations [5]. Moreover, when using analog hardware noise is always present due to thermal or flicker noise [6, 7, 8] and even if digital hardware is used quantization noise contaminates the weights and inputs [9]. Thus, the neural network structure should be resistant against these noises.

In this paper we analyze the radial basis function network with respect to equicontinuity and robustness. In [10] these properties are demonstrated for sigmoidal feedforward networks. First, a short overview about the analyzed network is given. Then the equicontinuous property is analyzed and necessary restrictions are introduced to get an equicontinuous network. Section 4 determines upper bounds on the mean square error for noise contaminated inputs and weights.

## 2 Radial Basis Functions

In this section a short overview about the architecture of a radial basis function network is given. The network consists of an input vector with a dimension

---

<sup>\*</sup> This work was supported by the Graduate College 776 - Automatic Configuration in Open Systems- funded by the Deutsche Forschungsgemeinschaft.

$\dim \mathbf{x} = n$ . At a second step  $m$  different basis functions which have different centers are superposed and denoted by a weight to produce the output.

The radial basis function network (RBF) can be used for local function approximation [11]. Basing on the regularization theory [12] the quadratic error is minimized with respect to a stabilizing term. Due to this stabilizer the interpolation and approximation quality is controlled in order to achieve a smooth approximation. Based on this stabilizer different basis functions can be performed for superposition. As a consequence, the network function can be expressed as

$$f_m(\mathbf{x}) = \sum_{i=1}^m \alpha_i h_i(\|\mathbf{x} - \mathbf{x}_i\|) \quad (1)$$

where  $m$  denotes the number of superposed basis functions.

The function  $h_i(z)$  can be any function related to a (radial) regularization stabilizer. Here, the stabilizer leading to a Gaussian function is considered, thus it follows

$$h_i(z) = \exp\left(\frac{-z^2}{2\sigma_i^2}\right) \quad (2)$$

and therefore

$$f_m(\mathbf{x}) = \sum_{i=1}^m \alpha_i \exp\left(\frac{-\|\mathbf{x} - \mathbf{x}_i\|^2}{2\sigma_i^2}\right) \quad (3)$$

Moreover, the parameters  $\mathbf{x}_i$  are the individual centers of each basis function,  $\sigma_i^2$  resembles the variance of each Gaussian function and  $\alpha_i$  denotes the weight from the  $i$ -th neuron to the output neuron, which performs a linear superposition of all basis functions. Thus, the neural network specifies a function set

$$\mathcal{S} = \left\{ f_m(\mathbf{x}) \mid f_m(\mathbf{x}) = \sum_{i=1}^m \alpha_i \exp\left(\frac{-\|\mathbf{x} - \mathbf{x}_i\|^2}{2\sigma_i^2}\right), m \in \mathbb{N}, \sigma_i, \alpha_i \in \mathbb{R}, \mathbf{x}_i \in \mathbb{R}^n \right\} \quad (4)$$

### 3 Equicontinuity of Radial Basis Functions

The equicontinuous property of functions is an important feature to produce a stable approximation. Stable approximation means that two slightly different inputs produce also slightly different output behavior meaning that small errors at the inputs only result in small errors at the output. Therefore, the nonequicontinuity produces an unstable approximation meaning that large discrepancies are possible while providing nearly identical values at the input.

Consequently, if the property of equicontinuity is fulfilled by a neural network function the whole network is expected to be fault tolerant and noise immune. If the inputs or weights are contaminated with noise or faults occur the network response differs only slightly in contrast to the ideal output behavior.

The equicontinuous property is defined as [10, 13].



**Definition 1.** Let  $\chi$  be a compact metric space with metric  $d$ , and let  $\mathcal{S}$  be a nonempty subset of  $\mathcal{C}(\chi)$ . By definition, if  $f$  is a member of  $\mathcal{S}$  then  $f$  is continuous, that is, for each  $\epsilon > 0$ , there exist  $\delta > 0$  such that  $d(x, y) < \delta \Rightarrow |f(x) - f(y)| < \epsilon$ .  $\mathcal{S}$  is said to be equicontinuous if for each  $\epsilon$  a  $\delta(\epsilon)$  can be found that serves at once for all functions  $f$  in  $\mathcal{S}$ .

### 3.1 Equicontinuity in the Input Space

For the equicontinuous property in the input space it holds

$$|f_m(\mathbf{x}) - f_m(\mathbf{y})| = \left| \sum_{i=1}^m \alpha_i \exp\left(\frac{-\|\mathbf{x} - \mathbf{x}_i\|^2}{2\sigma_i^2}\right) - \sum_{i=1}^m \alpha_i \exp\left(\frac{-\|\mathbf{y} - \mathbf{x}_i\|^2}{2\sigma_i^2}\right) \right| \quad (5)$$

By using the mean value theorem for some  $\xi \in \chi$  and the triangle inequality the following can be obtained

$$\begin{aligned} |f_m(\mathbf{x}) - f_m(\mathbf{y})| &= |\nabla_x f(\xi)| d(\mathbf{x}, \mathbf{y}) \quad (6) \\ &= \sum_{k=1}^n \left| \sum_{i=1}^m \alpha_i \exp\left(\frac{-\|\xi - \mathbf{x}_i\|^2}{2\sigma_i^2}\right) \cdot \frac{(-2)(\xi_k - x_{ik})}{2\sigma_i^2} \right| d(\mathbf{x}, \mathbf{y}) \\ &\leq \sum_{k=1}^n \sum_{i=1}^m \left| \alpha_i \exp\left(\frac{-\|\xi - \mathbf{x}_i\|^2}{2\sigma_i^2}\right) \cdot \frac{-(\xi_k - x_{ik})}{\sigma_i^2} \right| d(\mathbf{x}, \mathbf{y}) \\ &= \sum_{k=1}^n \sum_{i=1}^m |\alpha_i| \left| \exp\left(\frac{-\|\xi - \mathbf{x}_i\|^2}{2\sigma_i^2}\right) \cdot \frac{-(\xi_k - x_{ik})}{\sigma_i^2} \right| d(\mathbf{x}, \mathbf{y}) \end{aligned}$$

For a further analysis the derivation of the Gaussian function has to be evaluated. First, the values for  $|x_k| \rightarrow \infty$  are obtained by

$$\lim_{|x_k| \rightarrow \infty} \exp\left(\frac{-\|\mathbf{x} - \mathbf{x}_i\|^2}{2\sigma_i^2}\right) \cdot \frac{x_k - x_{ik}}{\sigma_i^2} = 0 \quad (7)$$

Consequently, the extrema of the derivation has been found. By setting the second derivation equal zero this leads to

$$-\exp(\cdot) \frac{(x_k - x_{ik})^2}{\sigma_i^4} + \exp(\cdot) \frac{1}{\sigma_i^2} = 0 \quad (8)$$

$$\Rightarrow (x_k - x_{ik})^2 = \sigma_i^2 \quad (9)$$

$$x_k = \pm \sigma_i + x_{ik} \quad (10)$$

Consequently, this leads to a maximum of the first derivation

$$\left| \exp(\cdot) \frac{\pm \sigma_i + x_{ik} - x_{ik}}{\sigma_i^2} \right| = \exp(\cdot) \frac{\pm \sigma_i}{\sigma_i^2} = \underbrace{|\exp(\cdot)|}_{\leq 1} \left| \frac{\pm \sigma_i}{\sigma_i^2} \right| \quad (11)$$

$$\leq \frac{1}{\sigma_i} \quad (12)$$

With the result of equation (11) the result of (6) can be further evaluated as

$$|\nabla_x f(\xi)| d(\mathbf{x}, \mathbf{y}) \leq \sum_{k=1}^n \sum_{i=1}^m |\alpha_i| \left| \frac{1}{\sigma_i} \right| d(\mathbf{x}, \mathbf{y}) \tag{13}$$

$$= n \cdot \sum_{i=1}^m \left| \frac{\alpha_i}{\sigma_i} \right| d(\mathbf{x}, \mathbf{y}) = \epsilon \tag{14}$$

With the result of (14) it follows

$$\delta \geq \frac{\epsilon}{n \cdot \sum_{i=1}^m \left| \frac{\alpha_i}{\sigma_i} \right|} \tag{15}$$

As a consequence of (15) the equicontinuous property is not valid for the radial basis function network in the input space, since relation (14) must be consistent to Definition 1 for all functions of set  $\mathcal{S}$  defined in (4). As a result of equation (15) the parameter  $\epsilon$  and  $\delta$  are depending on the weights  $\alpha_i$  and  $\sigma_i$ . Again, these weights are based on the function which should be approximated by the neural network and, consequently, both parameters ( $\epsilon, \delta$ ) are not independent of the function. Thus, an RBF has not the equicontinuous property in the input space like the MLP [10].

But the equicontinuous property can be adapted by the RBF network if the weights fulfill the following conditions. In order to make both parameters,  $\epsilon$  and  $\delta$ , independent of the network function, the weights have to be bounded. Thus, it follows

$$\begin{aligned} &\Rightarrow \sum_{i=1}^m \left| \frac{\alpha_i}{\sigma_i} \right| \text{ must be bounded} \\ &\Rightarrow |\alpha_i| \leq B \wedge \left| \frac{1}{\sigma_i} \right| \leq T \end{aligned} \tag{16}$$

$$\Rightarrow \sum_{i=1}^m \left| \frac{\alpha_i}{\sigma_i} \right| \leq \sum_{i=1}^m B \cdot T = m \cdot B \cdot T \tag{17}$$

Consequently, with the restriction of (16) the RBF network obtains the equicontinuous property in the input space.

### 3.2 Equicontinuity in the Weight Space

Let assume that the weight space is endowed by a metric and  $\mathcal{W}_1$  and  $\mathcal{W}_2$  are two distinct points in the weight space with  $d(\mathcal{W}_1, \mathcal{W}_2)$  denoting their distance. Then the equicontinuous property can be written as

$$\begin{aligned} |f_m(\mathcal{W}_1) - f_m(\mathcal{W}_2)| = & \left| \sum_{i=1}^m \alpha_i^1 \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}_i^1\|^2}{2\sigma_{i1}^2}\right) \right. \\ & \left. - \sum_{i=1}^m \alpha_i^2 \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}_i^2\|^2}{2\sigma_{i2}^2}\right) \right| \end{aligned} \tag{18}$$

Applying the mean value theorem to (18) this leads to

$$|f_m(\mathcal{W}_1) - f_m(\mathcal{W}_2)| \leq (|\nabla_\alpha f(\xi)| + |\nabla_{\mathbf{x}_i} f(\xi)| + |\nabla_\sigma f(\xi)|) d(\mathcal{W}_1, \mathcal{W}_2) \quad (19)$$

For the gradient operators in (19) follow

$$|\nabla_\alpha f(\xi)| = \sum_{i=1}^m \left| \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{2\sigma_i^2}\right) \right| \leq \sum_{i=1}^m |1| = m \quad (20)$$

$$\begin{aligned} |\nabla_{\mathbf{x}_i} f(\xi)| &= \sum_{k=1}^n \left| \sum_{i=1}^m \alpha_i \exp\left(-\frac{\|\mathbf{x} - \xi_i\|^2}{2\sigma_i^2}\right) \frac{x_k - \xi_{ik}}{\sigma_i^2} \right| \\ &\leq \sum_{k=1}^n \sum_{i=1}^m \left| \frac{\alpha_i}{\sigma_i} \right| = n \cdot \sum_{i=1}^m \left| \frac{\alpha_i}{\sigma_i} \right| \end{aligned} \quad (21)$$

$$\begin{aligned} |\nabla_\sigma f(\xi)| &= \sum_{i=1}^m \left| \alpha_i \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{2\xi^2}\right) \frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{\xi^3} \right| \\ &\leq \sum_{i=1}^m |\alpha_i| \left| \frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{\xi^3} \right| \end{aligned} \quad (22)$$

Equation (21) can be evaluated in the same manner as in (12). The estimation in (22) is determined by the maximum of each basis function

$$h(z) = \exp\left(\frac{-z^2}{2\sigma^2}\right) \leq 1 \quad (23)$$

Consequently, it follows for the equicontinuous property

$$\begin{aligned} |\nabla_{\mathcal{W}} f(\xi)| d(\mathcal{W}_1, \mathcal{W}_2) &\leq \left( m + n \cdot \sum_{i=1}^m \left| \frac{\alpha_i}{\sigma_i} \right| + \sum_{i=1}^m \left| \alpha_i \frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{\xi^3} \right| \right) d(\mathcal{W}_1, \mathcal{W}_2) \\ &= \epsilon \\ \Rightarrow \delta &\geq \frac{\epsilon}{m + n \cdot \sum_{i=1}^m \left| \frac{\alpha_i}{\sigma_i} \right| + \sum_{i=1}^m \left| \alpha_i \frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{\xi^3} \right|} \end{aligned} \quad (24)$$

As a consequence of (24) the RBF network also has not the equicontinuous property in the weight space since the parameters  $\epsilon$  and  $\delta$  depend on the weights  $\alpha_i$  and  $\sigma_i$  and the Euclidean distance of the input vector and the RBF centers. Since these weights are depending on the actual function which has to be approximated  $\epsilon$  and  $\delta$  also depend on the function. Thus, the RBF network is not equicontinuous in the weight space.

In order to attain equicontinuity in the weight space the same bounds for the weights as in section 3.1 can be set ( $|\alpha_i| \leq B$  and  $\left| \frac{1}{\sigma_i} \right| \leq T$ ). Moreover, the Euclidean distance  $\|\mathbf{x} - \mathbf{x}_i\|$  has to be bounded

$$\|\mathbf{x} - \mathbf{x}_i\| \leq C \quad (25)$$

and it follows

$$\sum_{i=1}^m \left| \frac{\alpha_i}{\sigma_i} \right| \leq \sum_{i=1}^m B \cdot T = mBT \tag{26}$$

$$\sum_{i=1}^m \left| \alpha_i \frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{\xi^3} \right| \leq \sum_{i=1}^m B \cdot C^2 \cdot T^3 = mBC^2T^3 \tag{27}$$

Since  $\sigma$  has a lower bound inequality (27) holds. Consequently, with the restrictions of equation (26) and (27) the RBF network has the equicontinuous property.

### 4 Robustness of Radial Basis Functions

In this section we analyze the tolerance of an RBF network to noise in the input and the weight space. For the analysis it is assumed that the inputs are restricted to the interval [-1:1] and are contaminated with Gaussian noise with zero mean and finite variance  $\sigma_n^2$ . Thus, noise contaminated inputs lead to a mean square error of

$$\text{mse} \leq n^2 \left( \sum_{i=1}^m \left| \frac{\alpha_i}{\sigma_i} \right| \right)^2 n \cdot \sigma_n^2 \tag{28}$$

Equation (28) implies that the mean square error is not bounded for an unrestricted RBF network in the case of Gaussian noise in the inputs. The restriction (cf. (16)) leads to

$$\text{mse} \leq n^3 m^2 B^2 T^2 \sigma_n^2 \tag{29}$$

Consequently, the mean square error has an upper bound with these restrictions.

For noise in the weight space the same assumption are made. All the weights are contaminated with Gaussian noise with zero mean and finite variance. For an RBF network with  $m$  basis function  $m + m \cdot n + m = m(n + 2)$  parameters are needed. This leads to an upper bound of the mean square error

$$\text{mse} \leq \left( m + n \cdot \sum_{i=1}^m \left| \frac{\alpha_i}{\sigma_i} \right| + \sum_{i=1}^m \left| \alpha_i \frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{\xi^3} \right| \right)^2 m(n + 2)\sigma_n^2 \tag{30}$$

If no restrictions are made, the mean square error has no upper bound. If the weights of the RBF network are bounded and the Euclidean distance between the input vector and the RBF centers is also bounded it follows for the upper bound

$$\text{mse} \leq (m + nmBT + mBC^2T^3)^2 m(n + 2)\sigma_n^2 \tag{31}$$

#### 4.1 Technical Realisation

Technical neural networks, especially technical RBF networks, are always restricted in their weights, thus the restrictions of (16) holds. Furthermore, due to the number representation in technical systems the Euclidean distance is also

bounded, and therefore restriction (25) holds. Consequently, practical realizations of an RBF network will have the equicontinuous property. In the case of an implementation of an RBF network in digital or analog hardware the mean square error is bounded due to a maximum resolution in both realizations.

As (29) and (31) imply the mean square error is depending on the size of the network (number of basis functions resp. neurons), the input dimension and the bounds of the weights and of the Euclidean distance. Furthermore, the variance of the noise is influencing the error. In the case of an analog realization the variance is closely related to the noise in this system. Thermal and flicker noise are Gaussian noise processes with zero mean and thus the noise power is identical to the variance  $\sigma_n^2$

$$\sigma_n^2 = E\left((X - E(X))^2\right) = E\left(X^2\right) = N \quad (32)$$

where  $E$  denotes the expected value,  $X$  the random variable and  $N$  the noise power.

As a consequence, the mean square error is closely related to the Signal-Noise-Ratio (SNR) of the system. Thus, this leads to

$$\text{SNR}_{\text{dB}} = 10 \log \left( \frac{P_s}{N} \right) \quad (33)$$

In an analog system this relation is bounded by the different architecture and by the number of transistors which are contributing to the noise power.

In a digital system the variance  $\sigma_n^2$  is related to the resolution  $k$  of the performed computation. From information theory [14] it is well known that the resolution is related to the noise power by

$$k = \frac{1}{2} \text{ld} \left( \frac{P_s}{N} \right) \quad (34)$$

As a result of (34) with an increasing resolution  $k$  of the computation the noise power is decreased. Thus, for a low mean square error a high resolution should be provided. On the contrary, if the resolution is related to the bounds of the weights the impact of the mean square error can be canceled.

## 5 Conclusion

We have shown the equicontinuous property of the radial basis function network under certain restrictions. Moreover, upper bounds on the mean square error are determined for noise contaminated inputs and weights.

Without these restrictions RBF networks are not equicontinuous and not noise immune. But restricting the trained weights by an upper resp. lower bound and the Euclidean distance between input vector and RBF centers this leads to a noise immune network. Fortunately, practical implementations providing these bounds naturally due to their limited resolution, memory and number

representation. But as a result of the limited resolution of digital systems noise contaminates the inputs and weights. On the contrary, analog systems have always to face noise in its signals.

The upper bound of the mean square error depends on the dimension of the input vector, the number of basis functions resp. the number of neurons, the upper and lower bounds and the noise power of the system. The important parameter is the noise power which should be decreased by a high resolution in a digital system and high SNR in an analog system. All the other parameters are closely correlated with each other while the input dimension is fixed for a certain problem. By restricting the weights by lower resp. upper bounds the number of neurons can be increased and thus counteracting the positive impact of the mean square error. But this has to be analyzed in detail.

## References

1. Geva, S., Sitte, J.: A constructive method for multivariate function approximation by multilayer perceptrons. *IEEE Transactions on Neural Networks* **3** (1992) 621–624
2. Haykin, S.: *Neural Networks. A Comprehensive Foundation*. Second edn. Prentice Hall, New Jersey, USA (1999)
3. Rückert, U., Surmann, H.: Tolerance of a binary associative memory towards stuck-at-faults. In Kohonen, T., ed.: *Artificial Neural Networks*. Volume 2., Amsterdam, North-Holland (1991) 1195–1198
4. Rückert, U., Kreuzer, I., Tryba, V.: Fault-tolerance of associative memories based on neural networks. In: *Proceedings of the International Conference on Computer Technology, Systems and Applications*, Hamburg, Germany (1989) 1.52–1.55
5. Beiu, V., Rückert, U., Roy, S., Nyathi, J.: On nanoelectronic architectural challenges and solutions. *Fourth IEEE Conference on Nanotechnology* (2004)
6. Razavi, B.: *Design of Analog CMOS Integrated Circuits*. McGraw-Hill (2000)
7. Sitte, J., Körner, T., Rückert, U.: Local cluster neural net: Analog vlsi design. *Neurocomputing* **19** (1998) 185 – 197
8. Körner, T., Rückert, U., Geva, S., Malmstrom, K., Sitte, J.: Vlsi friendly neural network with localied transfer functions. In: *Proceedings of the IEEE International Conference on Neural Networks*. Volume 1., Perth, Australia (1995) 169 – 174
9. Widrow, B., Kollár, J.: *Quantization Noise*. Prentice Hall PTR, New Jersey, USA (2002)
10. Chandra, P., Singh, Y.: Feedforward sigmoidal networks - equicontinuity and fault-tolerance properties. *IEEE Transactions on Neural Networks* **15** (2004) 1350–1366
11. Girosi, F., Poggio, T.: Networks and the best approximation property. *Biological Cybernetics* **63** (1990) 169–176
12. Girosi, F., Jones, M., Poggio, T.: Regularization theory and neural networks architectures. *Neural Computation* **7** (1995) 219–269
13. Simmons, G.F.: *Topology and Modern Analysis*. McGray-Hill, Tokyo, Japan (1964)
14. Shannon, C.E.: A mathematical theory of communication. *Bell System Technical Journal* **27** (1948) 379–423 and 623–656

# Improving Clustering Technique for Functional Approximation Problem Using Fuzzy Logic: ICFA Algorithm

A. Guillén, I. Rojas, J. González, H. Pomares, L.J. Herrera,  
O. Valenzuela, and A. Prieto

Department of Computer Architecture and Computer Technology,  
Universidad de Granada, Spain

**Abstract.** Clustering algorithms have been applied in several disciplines successfully. One of those applications is the initialization of Radial Basis Functions (RBF) centers composing a Neural Network, designed to solve functional approximation problems. The Clustering for Function Approximation (CFA) algorithm was presented as a new clustering technique that provides better results than other clustering algorithms that were traditionally used to initialize RBF centers. Even though CFA improves performance against other clustering algorithms, it has some flaws that can be improved. Within those flaws, it can be mentioned the way the partition of the input data is done, the complex migration process, the algorithm's speed, the existence of some parameters that have to be set in order to obtain good solutions, and the convergence is not guaranteed. In this paper, it is proposed an improved version of this algorithm that solves the problems that its predecessor has using fuzzy logic successfully. In the experiments section, it will be shown how the new algorithm performs better than its predecessor and how important is to make a correct initialization of the RBF centers to obtain small approximation errors.

## 1 Introduction

Designing an RBF Neural Network (RBFNN) to approximate a function from a set of input-output data pairs, is a common solution since this kind of networks are able to approximate any function [4, 9]. Formally, a function approximation problem can be formulated as, given a set of observations  $\{(\mathbf{x}_k; y_k); k = 1, \dots, n\}$  with  $y_k = F(\mathbf{x}_k) \in \mathbb{R}$  and  $\mathbf{x}_k \in \mathbb{R}^d$ , it is desired to obtain a function  $\mathcal{G}$  so  $y_k = \mathcal{G}(\mathbf{x}_k) \in \mathbb{R}$  with  $\mathbf{x}_k \in \mathbb{R}^d$ . Once this function is learned, it will be possible to generate new outputs from input data that were not specified in the original data set.

The initialization of the centers of RBFs is the first step to design an RBFNN. This task has been solved traditionally using clustering algorithms [8] [10]. Clustering techniques have been applied to classification problems [6], where the task to solve is how to organize observed data into meaningful structures. In classification problems, the input data has to be assigned to a pre-defined set of labels,

thus, if a label is not assigned correctly, the error will be greatly increased. In the functional approximation problem, a continuous interval of real numbers is defined to be the output of the input data. Thus, if the generated output value is near the real output, the error does not increase too much.

In this context, a new clustering algorithm for functional approximation problems was designed in our research group: Clustering for Functional Approximation (CFA)[5]. The CFA algorithm uses the information provided by the function output in order to make a better placement of the centers of the RBFs. This algorithm provides better results in comparison with traditional clustering algorithms but it has several elements that can be improved.

In this paper, a new algorithm is proposed, solving all the problems presented in the CFA algorithm using fuzzy logic techniques, and improving results, as it will be shown in the experiments section.

## 2 RBFNN Description

A RBFNN  $\mathcal{F}$  with fixed structure to approximate an unknown function  $F$  with  $n$  entries and one output starting from a set of values  $\{(\mathbf{x}_k; y_k); k = 1, \dots, n\}$  with  $y_k = F(\mathbf{x}_k) \in \mathbb{R}$  and  $\mathbf{x}_k \in \mathbb{R}^d$ , has a set of parameters that have to be optimized:

$$\mathcal{F}(\mathbf{x}_k; C, R, \Omega) = \sum_{j=1}^m \phi(\mathbf{x}_k; \mathbf{c}_j, r_j) \cdot \Omega_j \tag{1}$$

where  $C = \{\mathbf{c}_1, \dots, \mathbf{c}_m\}$  is the set of RBF centers,  $R = \{r_1, \dots, r_m\}$  is the set of values for each RBF radius,  $\Omega = \{\Omega_1, \dots, \Omega_m\}$  is the set of weights and  $\phi(\mathbf{x}_k; \mathbf{c}_j, r_j)$  represents an RBF. The activation function most commonly used for classification and regression problems is the Gaussian function because it is continuous, differentiable, it provides a softer output and improves the interpolation capabilities [2, 7]. The procedure to design an RBFNN for functional approximation problem is shown below:

1. Initialize RBF centers  $\mathbf{c}_j$
2. Initialize the radius  $r_j$  for each RBF
3. Calculate the optimum value for the weights  $\Omega_j$

The first step is accomplished by applying clustering algorithms, the new algorithm proposed in this paper will initialize the centers, providing better results than other clustering algorithms used for this task.

## 3 Clustering for Function Approximation Algorithm: CFA

This algorithm uses the information provided by the objective function output in such a way that the algorithm will place more centers where the variability of the output is higher instead of where there are more input vectors.



To fulfill this task, the CFA algorithm defines a set  $O = \{o_1, \dots, o_m\}$  that represents a hypothetic output for each center. This value will be obtained as a weighted mean of the output of the input vectors belonging to a center.

CFA defines an objective function that has to be minimized in order to converge to a solution:

$$\frac{\sum_{j=1}^m \sum_{\mathbf{x}_k \in C_j} \|\mathbf{x}_k - \mathbf{c}_j\|^2 \omega_{kj}}{\sum_{j=1}^m \sum_{\mathbf{x}_k \in C_j} \omega_{kj}} \tag{2}$$

where  $\omega_{kj}$  weights the influence of each input vector in the final position a center. The bigger the distance between the expected output of a center and the real output of an input vector is, the bigger the influence in the final result will be. The calculation of  $w$  is obtained by:

$$\omega_{kj} = \frac{|F(\mathbf{x}_k) - o_j|}{\max_{i=1}^n \{F(\mathbf{x}_i)\} - \min_{i=1}^n \{F(\mathbf{x}_i)\}} + \vartheta_{\min}, \quad \vartheta_{\min} > 0. \tag{3}$$

The first addend in this expression calculates a normalized distance (in the interval  $[0,1]$ ) between  $F(\mathbf{x}_k)$  and  $o_j$ , the second addend is a minimum contribution threshold. The smaller  $\vartheta_{\min}$  becomes, the more the centers are forced to be in areas where the output is more variable.

The CFA algorithm is structured in three basic steps: Partition of the data, centers and estimated output updating and a migration step.

The partition is performed as it is done in Hard C-means [3], thus, a Voronoi partition of the data is obtained. Once the input vectors are partitionated, the centers and their estimated outputs have to be updated, this process is done iteratively using the equations shown below:

$$\mathbf{c}_j = \frac{\sum_{\mathbf{x}_k \in C_j} \mathbf{x}_k \omega_{kj}}{\sum_{\mathbf{x}_k \in C_j} \omega_{kj}} \quad o_j = \frac{\sum_{\mathbf{x}_k \in C_j} F(\mathbf{x}_k) \omega_{kj}}{\sum_{\mathbf{x}_k \in C_j} \omega_{kj}} . \tag{4}$$

The algorithm, to update centers and estimated outputs, has an internal loop that iterates until the total distortion of the partition is not decreased significantly.

The algorithm has a migration step that moves centers allocated in input zones where the target function is stable, to zones where the output variability is higher. The idea of a migration step was introduced in [11] as an extension of Hard C-means.

CFA tries to find an optimal vector quantization where each center makes an equal contribution to the total distortion [4]. This means that the migration step will iterate, moving centers that make a small contribution to the error to the areas where centers make a bigger contribution.

### 3.1 Flaws in CFA

CFA has some flaws that can be improved, making the algorithm more robust and efficient and providing better results.

The first disadvantage of CFA is the way the partition of the data is made. CFA makes a hard partition of the data where an input vector can belong uniquely to a center, this is because it is based on the Hard C-means algorithm. When Fuzzy C-means [1] was developed, it demonstrated how a fuzzy partition of the data could perform better than a hard partition. For the functional approximation problem, it is more logical to apply a fuzzy partition of the data because an input vector can activate several neurons with a certain degree of activation, in the same way an input vector can belong to several centers in a fuzzy partition.

The second problem is the setting of a parameter which influences critically the results that can be obtained. The parameter is  $\vartheta_{\min}$ , the minimum contribution threshold. The smaller this parameter becomes, the slower the algorithm becomes and the convergence becomes less warranted. The need of a human expert to set this parameter with a right value is crucial when it is desired to apply the algorithm to different functions, because a wrong value, will make the algorithm provide bad results.

The third problem of CFA is the iterative process to converge to the solution. The convergence is not demonstrated because it is presented as a weighted version of Hard C-means, but the equations proposed do not warrant the convergence of the algorithm. The iterative method is quite inefficient because it has to iterate many times on each iteration of the main body of the algorithm.

The last problem CFA presents is the migration process. This migration step is quite complex and makes the algorithm run very slow. It is based on a distortion function that require as many iterations as centers, and adds randomness to the algorithm making it not too robust.

## 4 Improved CFA Algorithm: ICFA

Let's introduce the new elements in comparison with CFA, and let's see the reasons why this new elements are introduced.

### 4.1 Input Data Partition

As it was commented before, for the functional approximation problem, is better to use a fuzzy partition, but CFA uses a hard partition of the data. In ICFA, in the same way as it is done in Fuzzy C-means, a fuzzy partition of the data is used, thus, an input vector belongs to several centers at a time with a certain membership degree.

### 4.2 Parameter $w$

In CFA, the estimated output of a center is calculated using a parameter  $w$  (3). The calculation of  $w$  implies the election of a minimum contribution value ( $\vartheta_{\min}$ )

that will affect in a serious way the performance and the computing time of the algorithm.

In order to avoid the establishment of a parameter, ICFA removes this threshold, and the difference between the expected output of a center and the real output of the input data is not normalized. Thus, the calculation of  $w$  is done by:

$$w_{kj} = |F(\mathbf{x}_k) - o_j| \tag{5}$$

where  $F(\mathbf{x})$  is the function output and  $o_j$  is the estimated output of  $\mathbf{c}_j$ .

### 4.3 Objective Function and Iterative Process

In order to make the centers closer to the areas where the target function is more variable, a change in the similarity criteria used in the clustering process is needed. In Fuzzy C-means, the similarity criteria is the euclidean distance. Proceeding this way, only the coordinates of the input vectors are used, thus, the membership values  $u_{ik}$  for the matrix  $U = [u_{ik}]$  for a given center will be small for the input vectors that are far from that center, and the values will be big if the input vector is close to that center. For the functional approximation problem, this is not always true because, given a center, its associated cluster can own many input vectors even if they are far from this center but they have the same output values.

To consider these situations, the parameter  $w$  is introduced (5) to modify the values of the distance between a center and an input vector.  $w$  will measure the difference between the estimated output of a center and the output value of an input vector. The smaller  $w$  is, the more the distance between the center and the vector will be reduced. This distance is calculated now by modifying the norm in the euclidean distance:

$$D_{kj}W = \|\mathbf{x}_k - \mathbf{c}_j\|^2 \cdot w_{kj}^2. \tag{6}$$

where  $w_{kj} = |Y_k - o_j|$ . The objective function to be minimize is redefined as:

$$J_h(U, C, W) = \sum_{k=1}^n \sum_{i=1}^m u_{ik}^h D_{kj}W. \tag{7}$$

This function is minimized applying the LS method, obtaining the following equations that will converge to the solution:

$$u_{ik} = \left( \sum_{j=1}^m \left( \frac{D_{ik}W}{D_{jk}W} \right)^{\frac{2}{h-1}} \right)^{-1} \quad \mathbf{c}_i = \frac{\sum_{k=1}^n u_{ik}^h \mathbf{x}_k w_{ik}^2}{\sum_{k=1}^n u_{ik}^h w_{ik}^2}$$

$$\mathbf{o}_i = \frac{\sum_{k=1}^n u_{ik}^h Y_k d_{ik}^2}{\sum_{k=1}^n u_{ik}^h d_{ik}^2} \tag{8}$$

where  $d_{ij}$  is the euclidean distance between  $c_i$  and  $x_j$ , and  $h > 1$  is a parameter that allow us to control how fuzzy will be the partition and usually is equal to 2.

These equations are the equivalence of the ones defined for CFA (4) where the centers and their expected outputs are updated. These equations are obtained applying Lagrange multipliers and calculating the respect derivatives of the function, so convergence is warranted, unlike in CFA. ICFA, requires only one step of updating, being much more efficient than CFA where an internal loop is required on each iteration of the algorithm to update the centers and the outputs.

#### 4.4 Migration Step

As in CFA, a migration step is incorporated to the algorithm. CFA's migration iterates many times until each center contributes equally to the error of the function defined to be minimized. On each iteration, all centers are considered to be migrated, making the algorithm inefficient and, since it adds random decisions, the migration will affect directly to the robustness of the final results.

ICFA only makes one iteration and instead of considering all centers to be migrated, it performs a pre-selection of the centers to be migrated. The distortion of a center is the contribution to the error of the function to be minimized. To decide what centers will be migrated, it is used a fuzzy rule that selects centers that have a distortion value above the average. By doing this, centers that do not add a significant error to the objective function are excluded because their placement is correct and they do not need help from other center.

There is a fixed criteria to choose the centers to be migrated, in opposite to CFA where a random component was introduced at this point. The center to be migrated will be the one that has assigned the smallest value of distortion. The destination of the migration will be the center that has the biggest value of distortion. The repartition of the input vectors between those two it is like in CFA. If the error is smaller than the one before the migration step, the migration is accepted, otherwise is rejected.

#### 4.5 ICFA General Scheme

Once all the elements that compose the algorithm have been described, the general scheme that ICFA follows is:

**Do**

Calculate the weighted distance between  $C_i$  and  $X$  using  $w$

Calculate the new  $U_i$ ,  $C_i$  using  $U_i$  and  $O_i$  using  $C_i$

Migrate

**While**( $\text{abs}(C_{i-1}-C_i < \text{threshold})$ )

In ICFA, the start point is not a random initialization of matrix  $U$  as in Fuzzy C-means. In the new algorithm, centers will be distributed uniformly through the input data space and their estimated outputs will be equal to the difference between the maximum and the minimum value of the output function. Proceeding like this, all random elements of the previous algorithm are excluded, obtaining the maximum robustness.

## 5 Experimental Results

To compare the results provided by the different algorithms, it will be used the normalized root mean squared error (NRMSE)

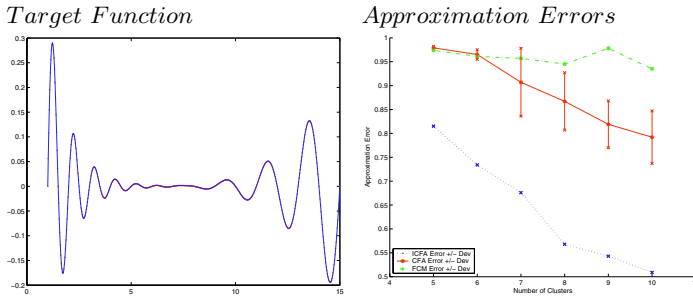
The radii of the RBFs were calculated using the k-neighbors algorithm with  $k=1$ . The weights were calculated optimally by solving a linear equation system.

Table 1 shows the errors when approximating the function  $f_1$  (Fig. 1) using the ICFA, Fuzzy C-means and CFA algorithms. In Fig. 1 are represented graphically the results shown in Table 1. Function  $f_1$  is defined as:

$$Y = \sin(2\pi X)e^{-X} - 34\sin(\pi X)e^{-75/X}. \tag{9}$$

**Table 1.** Mean and Standard Deviation of the approximation error (NRMSE) for function  $f_1$

| Clusters | FCM           | CFA          | FCFA     |
|----------|---------------|--------------|----------|
| 6        | 0.961(0.3E-4) | 0.965(0.010) | 0.734(0) |
| 7        | 0.957(0.1E-4) | 0.907(0.071) | 0.676(0) |
| 8        | 0.945(0.4E-4) | 0.867(0.060) | 0.568(0) |
| 9        | 0.978(0.001)  | 0.819(0.049) | 0.543(0) |
| 10       | 0.935(0.001)  | 0.792(0.055) | 0.509(0) |



**Fig. 1.** Target Function (blue line) and training set (red dots). Mean and Standard Deviation of the approximation error (NRMSE)

The results clearly show the improvement in performance of ICFA in comparison with CFA and its predecessors, not improving only the results, but the robustness.

## 6 Conclusions

RBFNNs provides good results when they are used for functional approximation problems. The CFA algorithm was designed in order to make the right initialization of the centers for the RBFs improving the results provided by the clustering

algorithms that were used traditionally for this task. CFA had some mistakes and disadvantages that could be improved. In this paper, a new algorithm which fix all the problems in CFA is proposed. This new algorithm performs much better than its predecessor.

From the analysis of the results, the following conclusions are obtained:

- It has been demonstrated how important an initialization step is when designing RBFNN for functional approximation problems.
- All problems in CFA were solved so the new algorithm obtains better results.

## Acknowledgments

This work has been partially supported by the Spanish CICYT Project TIN2004-01419.

## References

1. J. C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum, Nueva York, 1981.
2. Adrian G. Bors. Introduction of the Radial Basis Function (RBF) networks. *On-Line Symposium for Electronics Engineers*.
3. R.O. Duda and P.E. Hart. *Pattern classification and scene analysis*. New York:wiley, 1973.
4. A. Gersho. Asymptotically Optimal Block Quantization. *IEEE Transactions on Information Theory*, 25(4):373–380, July 1979.
5. J. González, I. Rojas, H. Pomares, J. Ortega, and A. Prieto. A new Clustering Technique for Function Aproximation. *IEEE Transactions on Neural Networks*, 13(1):132–142, January 2002.
6. J. A. Hartigan. *Clustering Algorithms*. New York:wiley, 1975.
7. A. Prieto I. Rojas, M. Anguita and O. Valenzuela. Analysis of the operators involved in the definition of the implication functions and in the fuzzy inference process. *Int. J. Approximate Reasoning*, 19:367–389, 1998.
8. N. B. Karayannis and G.W. Mi. Growing radial basis neural networks: Merging supervised and unsupervised learning with network growth techniques.
9. J. Park and J. W. Sandberg. Universal approximation using radial basis functions network. *Neural Computation*, 3:246–257, 1991.
10. Y. Cai Q. Zhu and L. Liu. A global learning algorithm for a RBF network.
11. M. Russo and G. Patan. Improving the LBG Algorithm. *Lecture Notes in Computer Science*, 1606:621–630, 1999.

# Input Variable Selection in Hierarchical RBF Networks

Mohammed Awad, Héctor Pomares, Ignacio Rojas,  
Luis J. Herrera, and Alberto Prieto

Dept. of Computer Architecture and Computer Technology,  
University of Granada, Granada, Spain  
awad@atc.ugr.es

**Abstract.** In this paper we propose a new technique focused on the search of new architectures for modelling complex systems in function approximation problems, in order to avoid the exponential increase in the complexity of the system that is usual when dealing with many input variables. The new hierarchical network proposed, is composed of complete Radial Basis Function Networks (RBFNs) that are in charge of a reduced set of input variables. For the optimization of the whole net, we propose a new method to select the more important input variables, thus reducing the dimension of the input variable space for each RBFN. We also provide an algorithm which automatically finds the most suitable topology of the proposed hierarchical structure and selects the more important input variables for it. Therefore, our goal is to find the most suitable of the proposed families of hierarchical architectures in order to approximate a system from which a set of input/output (I/O) data has been extracted.

## 1 Introduction

In many real world practical modelling problems, it is often possible to measure the value of many physical signals (variables), but it is not necessarily known which of them are relevant and required to solve the problem [1]. An excessively high computational complexity can occur when developing multivariate models for industrial or medical applications when the best set of inputs to use is not known. The main problems to face here are that when the input dimensionality increases, the computational complexity and memory requirements of the model increase (in some cases even exponentially); learning is more difficult with unnecessary inputs; and understanding complex models are much more difficult than simple models.

Neural networks can be defined as an architecture comprising massively parallel adaptive processing elements interconnected via structured networks. There are many network structures. The main weakness of a neural network lies in its totally flat structure. A direct consequence of such structural simplicity is often a huge network, with an excessively large number of hidden units. One effective solution is to incorporate proper hierarchical structure into the network. Hierarchical structures have a very rich variety of applications in computing since they provide representations that can be composed, modified, and manipulated in a very flexible way [2].

For example, in [8], de Souza et al. proposed a new hybrid neuro-fuzzy model, named hierarchical neuro-fuzzy quadtree (HNFQ), which was based on a recursive partitioning method of the input space named quadtree. Fukumizu, and Amari in [7] investigate the hierarchical geometric structure of the parameter space of three-layer perceptrons in order to show the existence of local minima and plateaus. In the paper, the authors proved that a critical point of the model with  $h$  hidden units always gives many critical points of the model with  $h+1$  hidden unit. Also, in [9] a hierarchical radial basis function network is presented. The main characteristic is that it is constituted of hierarchical layers, each containing a Gaussian grid at a decreasing scale, which are inserted only where the local error is over a certain threshold guaranteeing a uniform residual error and the allocation of more units with smaller scales where the data contain higher frequencies.

The main problem to solve is that when the number of input variables increases, the number of parameters usually increases in a very rapid way, even exponentially. This phenomenon named the curse of dimensionality [5] prevents the use of the majority of conventional modelling techniques and forces us to look for more specific solutions. To deal with this problem, input variable selection (IVS) procedures try to reduce the dimension of the input variable space, identifying and removing as much irrelevant and redundant data as possible, thus reducing the dimensionality of the data and allowing learning algorithms to operate faster and more effectively.

Input variable selection (IVS) has been researched intensively and has been applied to various problems such as data mining, knowledge discovery, pattern recognition, etc. One of the most popular methods used to select input variables is principal component analysis (PCA) which essentially transforms linearly a number of possibly correlated variables into a smaller number of uncorrelated variables which account for as much of the variability in the data as possible. The main drawback of PCA is that it considers only linear relations between variables. Several authors have also worked to select the most important input variables in function approximation problems. Pomares et al in [3] presented a method to obtain the structure of a complete rule-based fuzzy system for a specific approximation accuracy of the training data, deciding which input variables should be taken into account how many membership functions were needed in every selected input variable in order to reach the approximation target. The main drawback of that method is that it only could be applied to grid-based fuzzy systems with a limited number of input variables. Vehtari and Lampinen in [1] proposed to use posterior and marginal posterior probabilities obtained via variable dimension Markov chain Monte Carlo methods to find out potentially useful input combinations and to do the final model choice and assessment using the expected utilities computed by using the cross-validation predictive densities. Also noteworthy is the work made by Chen and Wang in [4], who proposed for a given a set of input and output variables, a fuzzy partition associating fuzzy sets with each input variable.

In our particular case, hierarchical structures will be used to provide a suitable construction of Multi-Radial Basis Function Networks (Multi-RBFNs) which improve significantly the performance of complex function approximation problems. In this paper we will see how our Multi-RBFN is capable of modelling complex systems without the above-mentioned problems inherent to the increase of the number of input variables. For that purpose, we provide a new method to select the more important input variables, thus reducing the dimension of the input variable space for each



RBFN, in order to find a model with the smallest number of input variables having statistically or practically at least the same expected utility as the full model with all the available inputs. We also propose an algorithm which automatically finds the most suitable topology of the proposed hierarchical structure and selects the more important input variables for it. Therefore, our goal is to find the most suitable of the proposed families of hierarchical architectures in order to approximate a system from which a set of input/output (I/O) data has been extracted.

The paper is organized as follows. *Section 2* describes the basic building modules and the hierarchical structures of the Multi-RBFN. The differences between several Multi-RBFN architectures are also discussed. *Section 3* presents the new proposed method for input variable selection, for our hierarchical Multi-RBFNs. Finally *Section 4* provides an example of how the proposed methodology is capable of finding the most suitable Multi-RBFN architecture.

## 2 Architecture of the Multi-RBFN

Fig. 1 presents the proposed hierarchical Multi-RBFN system. Each one of the nodes of the figure is a Radial Basis Functions Network (see Fig. 2). RBFNs can be seen as a particular class of Artificial Neural Networks ANNs. They are characterized by a transfer function in the hidden unit layer having radial symmetry with respect to a centre. The basic architecture of an RBFN is a 3-layer network. The output of the net is given by the following expression:

$$F(\vec{x}, \Phi, w) = \sum_{i=1}^m \phi_i(\vec{x}) \cdot w_i \tag{1}$$

where  $\Phi = \{\phi_i : i = 1, \dots, m\}$  are the basis functions set and  $w_i$  the associate weights for every RBF. The basis function  $\phi$  can be calculated as a gaussian function using the following expression:

$$\phi(\vec{x}, \vec{c}, r) = \exp\left(-\frac{\|\vec{x} - \vec{c}\|^2}{r}\right) \tag{2}$$

where  $\vec{c}$  is the central point of the function  $\phi$  and  $r$  is its radius.

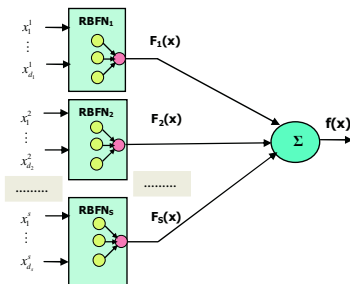


Fig. 1. hierarchical structures Multi-RBFN

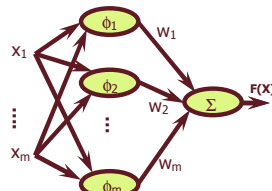


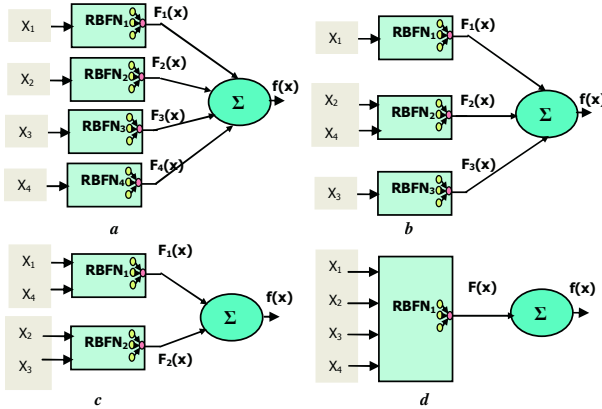
Fig. 2. Radial Basis Function Network

Each subset of the input variables  $\{x_1, \dots, x_d\}$  can be used as the inputs of each Sub-RBFN. Each Sub-RBFN receives their specific inputs variables values, and realizes the computation given in Eq. (1) to provide the output to the final node which simply adds-up the output of every Sub-RBFN. Therefore, the final output can be expressed as follows:

$$f(\bar{x}, \Phi, w) = \sum_{s=1}^S \sum_{i=1}^{m_s} \phi_i^s(\bar{x}) \cdot w_i^s \tag{3}$$

where  $\phi_i^s$  are the  $i$ -th basis functions of the  $s$ -th RBFN, and  $w_i^s$  is its weight.

Several hierarchical structures Multi-RBFN can be obtained for any given problem from a set of input variables. For example, for a 4-input problem  $\{x_1, \dots, x_4\}$ , many possible different architectures can be obtained, the simplest when each input variable forms a single set (See Fig. 3a), and the one that implies a bigger number of parameters which corresponds to the case where all input variables are used in the only Sub-RBFN (See Fig. 3d). Fig. 3b and c show other possible configurations in between.



**Fig. 3.** Different topologies of hierarchical Multi-RBFNs. a) 4 Sub-RBFNs with one input variable for each one b) 3 Sub-RBFNs with one and two input variables for each one. c) 2 Sub-RBFNs with two input variables for each one. d) 1 sub-RBFN with all the input variable set

To gain an insight of how the Multi-RBFN configuration affects the number of actual parameters of the system, let us recall that the total number of parameters in every Sub-RBFN is equal to  $m_s \cdot (d_s + 2)$ , where  $m_s$  is the number of RBFs in the  $s$ -th Sub-RBFN, and  $d_s$  is the number of input variables actually used by this  $s$ -th Sub-RBFN. Table 1 shows the number of parameters used in each one of the architectures of Fig. 3 using (for fair comparison) a total number of 24 RBFs for each one. We can see how even for this simple example with only 4 input variables to share, the differences can be notable (the number of parameters can be doubled). The hierarchical structure Multi-RBFN is thus capable of decreasing the number of parameters to optimize, provided that the selected structure is the most suitable one for the given set of I/O data examples.

In this paper we are concerned exclusively with the selection of the most suitable Multi-RBFN structure. However, some remarks could be made about the optimization of the rest of the parameters of the net, i.e. RBF centres, RBF radii and RBF weights. To optimize the centres of each RBF of each Sub-RBFN, it is common to use clustering algorithms such as the one presented in [6]. For the radii, several heuristics have been proposed in the literature such as the k-nearest neighbour technique [10]. Once the parameters of centres and radii of each Sub-RBFN has been initialized we can use a linear optimization method for optimizing the values of the weights that minimize the least square errors, since the Multi-RBFN topology preserves the linearity of the output functions w.r.t the RBF weights, as in the simpler RBFN. In the next section, we will concentrate on the problem of estimating the suitable hierarchical Multi-RBFN topology from a given I/O data set, i.e. which input variables should we use and which of them go together in every Sub-RBFN.

**Table 1.** Number of parameters between different architectures Multi-RBFN

| Fig # | #Sub-RBFNs | #RBF in each Sub-RBFN | #var. in each Sub-RBFN | #parm in each Sub-RBFN | #parm in Multi-RBFN |
|-------|------------|-----------------------|------------------------|------------------------|---------------------|
| 2a    | 4          | 6                     | 1                      | 18                     | 72                  |
|       |            | 6                     | 1                      | 18                     |                     |
|       |            | 6                     | 1                      | 18                     |                     |
|       |            | 6                     | 1                      | 18                     |                     |
| 2b    | 3          | 6                     | 1                      | 18                     | 84                  |
|       |            | 12                    | 2                      | 48                     |                     |
|       |            | 6                     | 1                      | 18                     |                     |
| 2c    | 2          | 12                    | 2                      | 48                     | 96                  |
|       |            | 12                    | 2                      | 48                     |                     |
| 2d    | 1          | 24                    | 4                      | 144                    | 144                 |

### 3 Input Variable Selection for the Multi-RBFNs

An input variable selection method tries to reduce the dimension of the input variable space and creates a new input variable set, thus identifying and removing as much irrelevant and redundant data as possible, which reduces the dimensionality of the data and allows learning algorithms to operate faster and more effectively.

In this section we propose a new method for input variables selection for the problem of function approximation, and more specifically for our Multi-RBFN system. This method considers a simple calculation to select the input variables, trying to relate each possible input dimension of the data  $\{x_1, \dots, x_d\}$  with the dependent variable  $y$ , dividing the data in every dimension  $x_i$  in  $p$  parts

$$\{P_i^{j-1} \leq (\bar{x}^k)_i < P_i^j\} \quad k = 1, \dots, K; i = 1, \dots, d; j = 1, \dots, p \tag{4}$$

where  $K$  is the number of I/O data and  $(\bar{x}^k)_i$  is the  $i$ -th component of the  $k$ -th input vector; and associating the data of each part to his corresponding output data

$$\{(\bar{x}^k)_i, y^k\} / P_i^{j-1} \leq (\bar{x}^k)_i < P_i^j \tag{5}$$

The next step is the calculation of the distance  $D_i^j$  between the maximum and the minimum values of the output in each partition of the input variable  $x_i$ . Finally, for each input variable  $x_i$  we calculate the mean of those distances  $\bar{D}_i$ . The smaller  $\bar{D}_i$ , the more important the input variable is for the problem, since this implies that the other variables affect very little to the output variable for every fixed value (partition interval) of  $x_i$ . Fig. 4 shows, in a schematic way, the general description of the proposed IVS method.

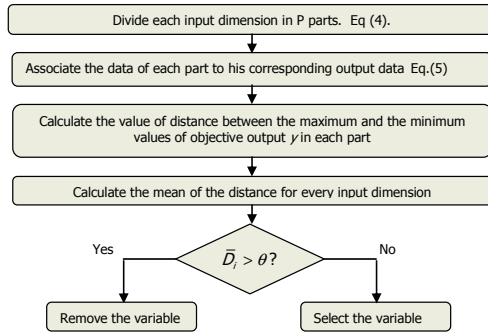


Fig. 4. General description of the IVS method

Finally, to select which input variables should be together in each Sub-RBFN of the hierarchical Multi-RBFN structure, we can use the mean distances obtained for each input dimension, so when two or more input variables have a similar mean distance values, these variables will be assigned the same Sub-RBFN.

### 4 Simulation Examples

In order to facilitate the understanding of the proposed IVS method, we will take an example with 6 possible input variables to choose from. Let us consider a set of 25000 I/O data pairs randomly taken from the function

$$f(x_1, x_2, x_3, x_4, x_5, x_6) = \sin(2\pi x_1 x_2) + 2x_3 \cdot \exp(-5 x_4) + 0x_5 + 0.0001x_6 \tag{6}$$

where each input variable is defined in the interval [0,1].

Fig. 5 shows the graphical representation of the output variable w.r.t. every input variable, which we have partitioned into 20 intervals. After the calculation of the mean distance for each one of the 6 input variables, we obtain the results given in Table 2.

Table 2. Mean distance results obtained for the example

| Input Variable | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ |
|----------------|-------|-------|-------|-------|-------|-------|
| Mean Distance  | 0.68  | 0.68  | 0.79  | 0.80  | 0.90  | 0.91  |

From Fig. 5 we can see that the “spectrum” for  $x_5$  and  $x_6$  is practically “flat” meaning that they can be considered as noisy variables. This is corroborated by the big mean distance value obtained for these two variables by the algorithm. Therefore, the algorithm will automatically discard those variables and will choose  $x_1, x_2, x_3, x_4$  as the possible input variables for this problem, which is the correct decision, as can be easily verified from Eq. (6).

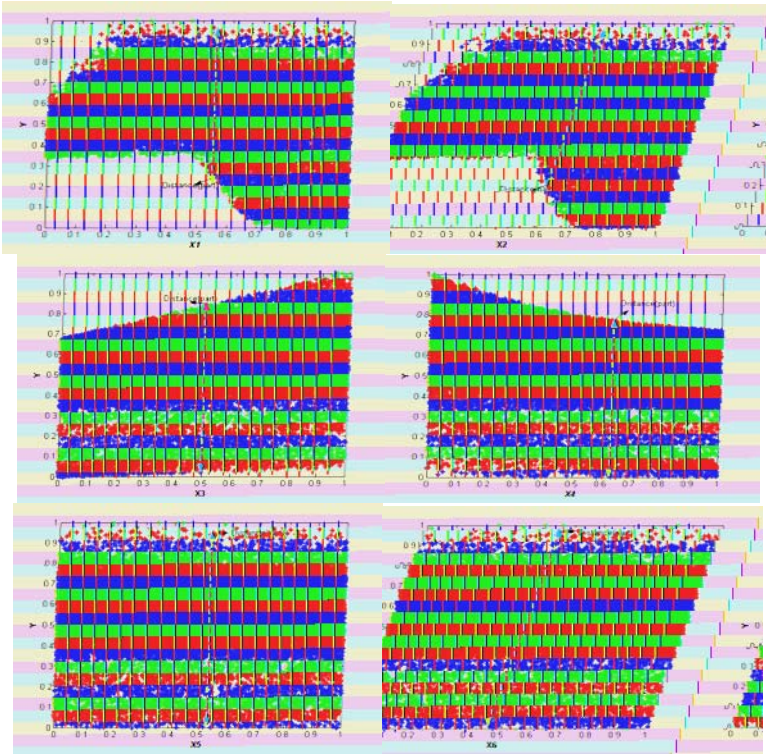


Fig. 5. Function output with respect to every input dimension

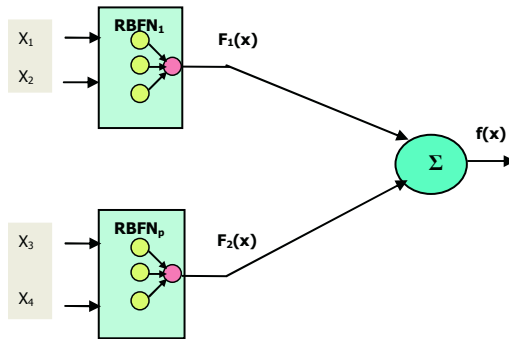


Fig. 6. Obtained hierarchical Multi-RBFN structure for the example

Now the algorithm evaluates the similarity in the mean distance values, and concludes that the variables  $\{x_3, x_4\}$  should be put together in one Sub-RBFN in the hierarchical Multi-RBFN structure, and that the variables  $\{x_1, x_2\}$  should also be the inputs to a new Sub-RBFN. Thus, the algorithm will select as the most suitable architecture the one depicted in Fig. 6. Once again, from Eq. (6), it is clear that this is the best architecture that we could have chosen for this problem.

## 5 Conclusions

A fundamental limitation in the problem of approximation systems is that when the number of input variables increases, the number of parameters usually increases in a very rapid way, even exponentially. This phenomenon prevents the use of the majority of conventional modelling techniques and forces us to look for more specific solutions. To deal with this problem, we have searched for new architectures for modelling complex systems in function approximation problems. The new hierarchical network proposed is composed of complete Radial Basis Function Networks that are in charge of a reduced set of input variables. For this architecture, we have proposed a new method to select the more important input variables, thus reducing the dimension of the input variable space for each RBFN. We have also provided a method which automatically finds the most suitable topology of the proposed hierarchical structure and selects the more important input variables for it.

## Acknowledgements

This work has been partially supported by the Spanish CICYT Project TIN2004-01419.

## References

1. Vehtari, A., Lampinen, J.: Bayesian Input Variable Selection Using Posterior Probabilities and Expected Utilities. Helsinki University of Technology, Laboratory of Computational Engineering publications. Report B. ISSN 1457-1404, 2002.
2. Kwasny, S. C., Kalman, B. L., Chang, N.: Distributed Patterns as Hierarchical Structures, World Congress on Neural Networks-Portland, OR. 1993.
3. Pomares, H., Rojas, I., González, J., Prieto, A.: Structure Identification in Complete Rule-Based Fuzzy Systems. IEEE Trans. On fuzzy systems, vol. 10, No. 3, June 2002
4. Chen, Y., Wang, J. Z.: Kernel machines and additive fuzzy systems: classification and function approximation. 789-795 vol.2. 2003
5. Bengio, S., Bengio, Y.: Taking on the curse of dimensionality in joint distributions using neural networks. IEEE Trans. Neur. Net., special issue on data mining and knowledge discovery, 11(3):550-557, 2000.
6. Gonzalez, J., Rojas, H., Ortega, J., Prieto, A.: A new clustering technique for function approximation. Neural Networks, IEEE Trans. on, Volume: 13 Issue: 1, Jan. 2002. Page(s): 132 -142.

7. Fukumizu, K., Amari, S-I.: Local Minima and Plateaus in Hierarchical Structures of Multilayer Perceptrons. Brain Science Institute. The Institute of Physical and Chemical Research (RIKEN) October 22, 1999.
8. de Souza, F.J., Vellasco, M.M.R., Pacheco, M.A.C.: Hierarchical neuro-fuzzy quadtree models. *Fuzzy Sets and Systems* 130 (2002) 189–205.
9. Ferrari, S., Maggioni, M., Borghese, N.A.: Multiscale approximation with hierarchical radial basis functions networks. *IEEE Trans. Neur. Net.*, vol.15, no.1, pp.178-188, 2004.
10. Moody J. and Darken C. Fast learning in networks of locally tuned units. *Neural Computations*.1989. 1(2):281-294.

# Approximating I/O Data Using Radial Basis Functions: A New Clustering-Based Approach

Mohammed Awad, Héctor Pomares, Luis Javier Herrera, Jesús González,  
Alberto Guillén, and Fernando Rojas

Dept. of Computer Architecture and Computer Technology,  
University of Granada, Granada, Spain  
awad@atc.ugr.es

**Abstract.** In this paper, we deal with the problem of function approximation from a given set of input/output data. This problem consists of analyzing these training examples so that we can predict the output of the model given new inputs. We present a new method for function approximation of the I/O data using radial basis functions (RBFs). This approach is based on a new efficient method of clustering of the centres of the RBF Network (RBFN); it uses the objective output of the RBFN to move the clusters instead of just the input values of the I/O data. This method of clustering, especially designed for function approximation problems, improves the performance of the approximator system obtained, compared with other models derived from traditional algorithms.

## 1 Introduction

Function approximation is the name given to a computational task that is of interest to many sciences and engineering communities [1]. Function Approximation consists of synthesizing a complete model from samples of the function and its independent variables [2]. In supervised learning, the task is that of learning a mapping from one vector space to another with the learning based on a set of instances of such mappings. We assume that a function  $F$  does exist and we endeavour to synthesize a computational model of that function. As a general mathematical problem, function approximation has been studied for centuries. However, some knowledge of the function to be approximated is usually assumed, depending on the specific problem. For example, in pattern recognition, a function mapping is made whose objective is to assign each pattern in a feature space to a specific label in a class space.

When one makes no assumptions about a model of the function to be approximated, mathematical theory can only provide interpolation techniques such as Splines, Taylor expansions, Fourier series, etc. Under this assumption, we can also make use of the so called model-free systems. These systems include neural networks and fuzzy systems, among others.

Radial Basis Function Networks (RBFNs) can be seen as a particular class of Artificial Neural Networks ANNs. They are characterized by a transfer function in the hidden unit layer having radial symmetry with respect to a centre. The basic architecture of an RBFN is a 3-layer network. The output of the net is given by the following expression:



$$F(\bar{x}, \Phi, w) = \sum_{i=1}^m \phi_i(\bar{x}) \cdot w_i \quad (1)$$

where  $\Phi = \{\phi_i : i = 1, \dots, m\}$  are the basis functions set and  $w_i$  the associate weights for every RBF. The basis function  $\phi$  can be calculated as a gaussian function using the following expression:

$$\phi(\bar{x}, \bar{c}, r) = \exp\left(\frac{\|\bar{x} - \bar{c}\|}{r}\right) \quad (2)$$

where  $\bar{c}$  is the central point of the function  $\phi$  and  $r$  is its radius.

RBFNs are universal approximators and thus best suited for function approximation problems. In general an approximator is said to be universal if it can approximate any continuous function on a compact set to a desired degree of precision.

The technique of finding the suitable number of radial functions is very complex since we must be careful of not producing excessively large networks which are inefficient and sensitive to over-fitting and exhibit poor performances.

When the values of  $\bar{c}$  and  $r$  of the basis functions are known it is possible to use a linear optimization method for finding the values of  $w_i$  that minimize the cost function computed on the sample set. This method relies on the computation of the pseudo-inverse matrix.

Other methods proposed in the literature try to optimize also the centre values of the RBFs. For instance, in [3] Chen et al. propose an alternative learning procedure based on the orthogonal least-squares method. The procedure chooses radial basis function centres one by one in a rational way until an adequate network has been constructed; each selected centre maximizes the increment to the explained variance or energy of the desired output and does not suffer numerical ill-conditioning problems. Orr in [4] selects the centres among the samples of the basis functions that most contribute to the output variance. Another solution to this problem is to cluster similar samples of the input data together. Every cluster has a centroid, which can then be chosen as the centre of a new RBF. We can find in the literature some unsupervised clustering algorithms such as k-means [5], fuzzy c-means [6], enhanced LBG [7], and also some supervised clustering algorithms such as the Clustering for Function Approximation method (CFA) [8], the Conditional Fuzzy Clustering algorithm (CFC) [9] and the Alternating Cluster Estimation method (ACE) [10].

In this paper we present a new method for function approximation from a set of I/O data using radial basis functions (RBFs). This approach is based on a new efficient method of clustering of the centres of the RBF Network; it uses the target output of the RBFN to migrate and fine-tune the clusters instead of just the input values of the I/O data. This method of clustering, especially designed for function approximation problems, calculates the error committed in every cluster using the real output of the RBFN trying to concentrate more clusters in those input regions where the approximation error is bigger, thus attempting to homogenize the contribution to the error of every cluster.

After this introduction, the organization of the rest of this paper is as follows. Section 2 presents an overview of the proposed algorithm. In Section 3, we present in detail the proposed algorithm for the determination of the pseudo-optimal RBF parameters. Then, in Section 4 we show some results that confirm the goodness of the proposed methodology. Some final conclusions are drawn in Section 5.

## 2 Overview of the Proposed Algorithm

As mentioned before, the problem of function approximation consists of synthesizing a complete model from samples of the function and its independent variables. Consider a function  $y = f(\vec{x})$  where  $\vec{x}$  is a vector  $(x_1, \dots, x_n)$  in  $n$ -dimensional space from which there are available a set of input/output data pairs. The idea is to approximate these data with another function  $F(\vec{x})$ . The accuracy of the approximation is generally measured by a cost function which takes into account the error between the output of the RBFN and the target output. In this paper, the cost function we are going to use is the so-called Normalized Root Mean Squared Error (NRMSE). This performance index is defined as:

$$NRMSE = \sqrt{\sum_{i=1}^p (y_i - f(\vec{x}_i))^2 / \sum_{i=1}^p (y_i - \bar{y})^2} \tag{3}$$

where  $\bar{y}$  is the mean of the target output and  $p$  is the data number.

The objective of our algorithm, which is inspired in the CFA algorithm, is to increase the density of clusters in the input domain areas where the target function is less accurately approximated, rather than just in the zones where there are more input examples, as most unsupervised clustering algorithms would do, or in zones where more variability of the output is found, as it is the case of CFA.

The RBFN universal approximation property states that an optimal solution to the approximation problem can be found, which minimizes the NRMSE. In order to find the minimum of the error function, the RBFN is completely specified by choosing the following parameters: the number  $m$  of radial basis functions, the centres  $\vec{c}$  of every RBF, the radius  $r$ , and the weights  $w$ .

The number of RBFs is a critical choice. In our algorithm we have used a simple incremental method to determine the number of RBFs. We will stop adding new RBFs when the approximation error falls below a certain target error, in our case  $NRMSE_{TARGET}=0.1$ . As to the rest of the parameters of the RBFN, in Section 3 we present a new clustering technique especially suited for function approximation problems.

The basic idea we have developed is to calculate the error committed in every cluster using the real output of the RBFN to compute the error for each training data belonging to the cluster, and concentrating more clusters in those input regions where the cluster error is bigger. Fig. 1 presents a flow chart with the general description of the complete incremental algorithm.

## 3 Parameter Adjustment of the RBF Network

The locality property inherent to the Radial Basis Functions allows us to use a clustering algorithm to obtain the RBF centres. Clustering algorithms may get stuck in a local minimum ignoring a better placement of some of the clusters, i.e., the algorithm is trapped in a local minimum which is not the global one. For this reason we need a clustering algorithm capable to solve this local minima problem. To avoid this problem we endow our supervised algorithm with a migration technique. This modifica-

tion allows the algorithm to escape from local minima and to obtain a prototype allocation independent of the initial configuration.

To optimize the other parameters of the RBFN (the radius  $r$  and the weights  $w$ ) we used well-known heuristics such as the  $k$ -nearest neighbour technique (knn) for the initialization of the radius of each RBF, and some conventional techniques such as singular value decomposition (SVD) to directly optimize the weights. Finally, local minimization routines such as the Levenberg-Marquardt algorithm are finally used to fine-tune the obtained RBFN.

Therefore, in this section we will concentrate on the proposed clustering algorithm. In Fig. 2, we show a flowchart representing the general description of our clustering algorithm.

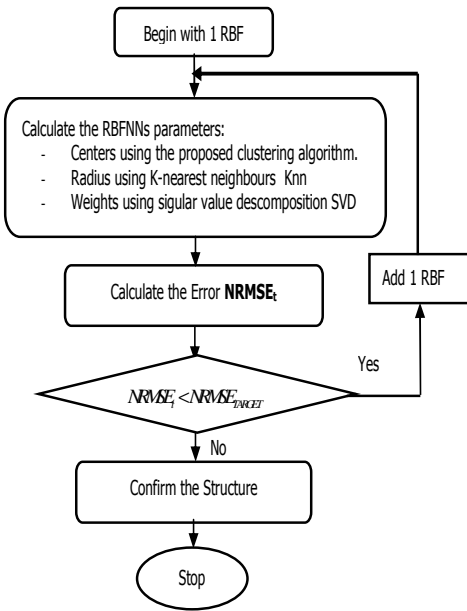


Fig. 1. General description of the algorithm

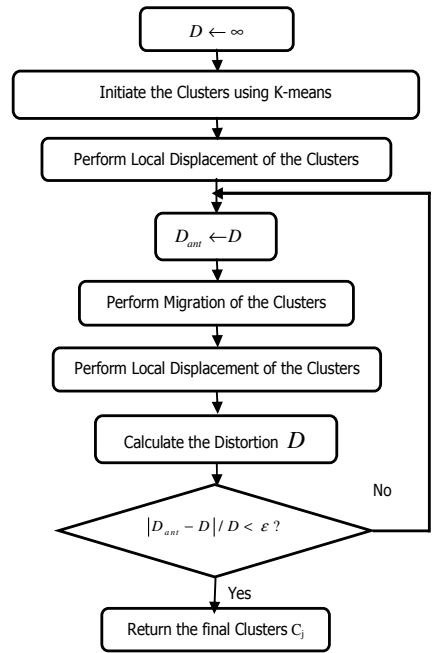


Fig. 2. General description of the proposed clustering algorithm

As can be seen from this figure, the initial values of the clusters are calculated using the  $k$ -means clustering algorithm followed by a local displacement process which locally minimizes the distortion ( $D$ ) within each cluster (see Fig. 3), which is defined as:

$$D = \frac{\sum_{j=1}^m \sum_{\bar{x}_i \in C_j} \|\bar{x}_i - \bar{c}_j\|^2 E_{ij}}{\sum_{j=1}^m \sum_{\bar{x}_i \in C_j} E_{ij}} \tag{4}$$

where  $m$  is the number of RBFs (clusters),  $\vec{c}_j$  is the centre of cluster  $C_j$  and  $E_{ij}$  is the error committed by the net when the input vector  $\vec{x}_i$  belongs to cluster  $C_j$ .

$$E = |y - f(\vec{x}, \Phi, w)| \tag{5}$$

In the Local Displacement of the Cluster Centres, we start by making a hard partition of the training set, just as in the k-means algorithm. This partition produces a Voronoi partition of the training data set. The second step of the process of local displacement is the calculation of the error of the RBFN using the the K-nearest neighbours algorithm to initiate the radii and the singular value decomposition to calculate the weights of the RBFs.

After this process we must update the cluster centres in order to minimize the total distortion (4). The algorithm stopped when the value of the distortion is less than the value of an threshold  $\epsilon$ .

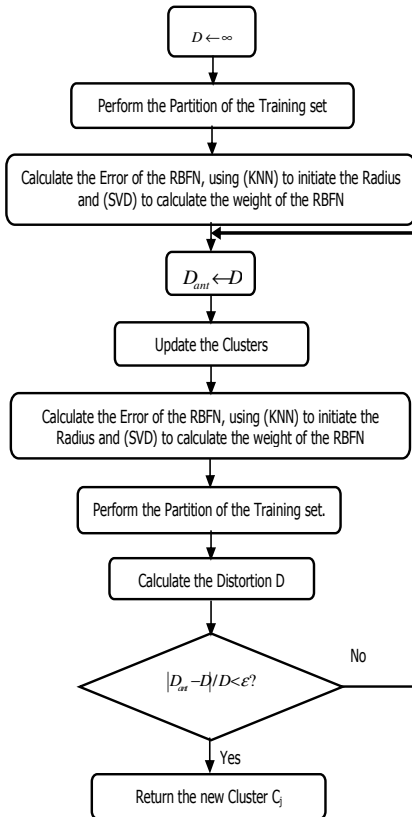


Fig. 3. Local Displacement of the Clusters

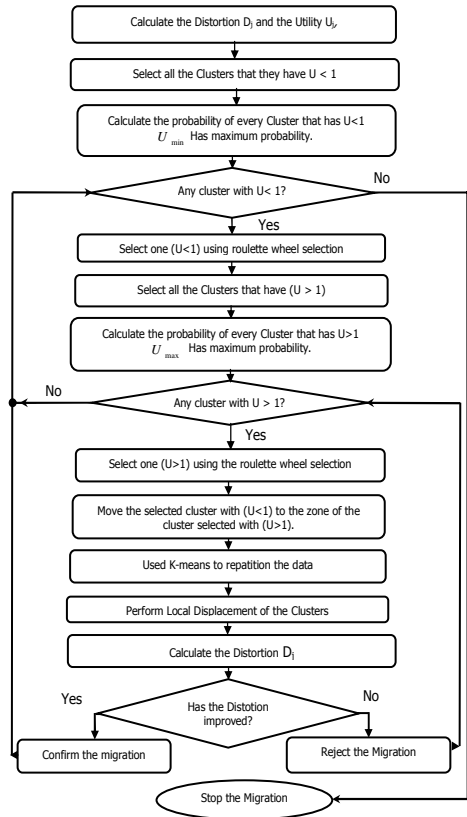


Fig. 4. The Migration Process

This is carried out by an iterative process that updates each cluster centres as the weighted mean of the training data belonging to that cluster and we repeat this process until the total distortion of the net reaches a minimum.

$$\bar{c}_m = \sum_{\bar{x}_i \in C_m} E_{im} \cdot \bar{x}_i / \sum_{\bar{x}_i \in C_m} E_{im} \tag{6}$$

The migration process migrates clusters from the better approximated zones toward those zones where the approximation error is worse, thus attempting to make equal their contribution to the total distortion. Our main hypothesis is that the best initial cluster configuration will be the one that equalizes the approximation error committed by every cluster. To avoid local minimums, the migration process uses a pseudo-random selection of the cluster to migrate, being the probability of choosing a given cluster inversely proportional to what we call “the utility” of that cluster, which is defined as

$$U_j = D_j / \bar{D} \quad j = 1, \dots, m \tag{7}$$

In this way, the proposed algorithm selects one cluster that has utility less than one and moves this cluster to the zone nearby a new selected cluster having utility more than one (see Fig. 4). This migration step is necessary because the local displacement of clusters only moves clusters in a local manner. It should be noted that using the k-means algorithm to divide the data that belonged to the zone that receives the new cluster is less complex and need less execution time than others migration algorithms such as ELBG and CFA.

### 4 Example of the Proposed Procedure

Let us consider the function [8]

$$f_1(x) = \sin(2\pi x) / e^x \quad x \in [0,10] \tag{8}$$

This has been chosen to demonstrate the importance of the equidistribution of the approximation errors throughout the clusters. This function (see Fig. 5a), has a very variable output when the input  $x$  is near to the value zero.

To test the effects caused by the proposed algorithm on the initialization and avoid local minima on the placement of the clusters, a training set of 2000 samples of the function was generated by evaluating inputs taken uniformly from the interval [0,10], from which we have removed 1000 points for validation.

The results of the proposed algorithm compared with the CFA algorithm, which resulted to be the best algorithm for this example in [8], with  $\mu_{\min} = 0.001$ , are represented in Table 1. In the table,  $NRMSE_C$  is the NRMSE of the training data after the clustering process is concluded.  $NRMSE_T$  is the final error index (for 10000 test data) obtained after the application of the Levenberg–Marquardt method. It must be noted, that both clustering algorithms were designed to provide an initial RBF configuration to be subsequently optimized using a local optimization method in order to find the global minimum.  $Std$  are the standard deviation of the error indices using 5 executions of both algorithms. Finally,  $Time_C$  is the clustering process execution time (in sec-

onds). As can be seen from the table, the proposed algorithm reaches better approximations using less time than the CFA algorithm, in all cases.

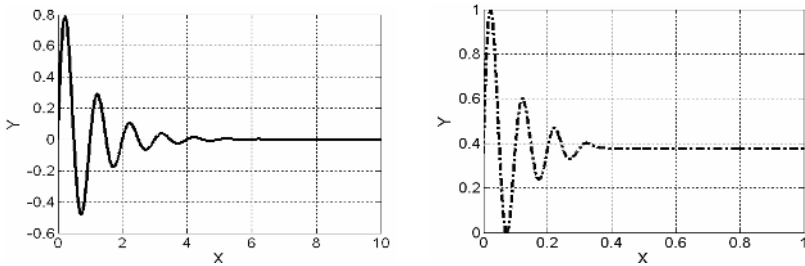


Fig. 5. a) Objective function. b) Approximation with 6 RBFs

Table 1. Comparison between CFA and the proposed approach

| $m$ | NRMSE <sub>C</sub> | Std  | NRMSE <sub>T</sub> | Std  | TIME <sub>C</sub> | NRMSE <sub>C</sub> | Std  | NRMSE <sub>T</sub> | Std  | TIME <sub>C</sub> |
|-----|--------------------|------|--------------------|------|-------------------|--------------------|------|--------------------|------|-------------------|
| 2   | 1.0                | 0.0  | 0.77               | 1E-1 | 0.27              | 1.01               | 2E-3 | 1.45               | 2E-8 | 8.1               |
| 3   | 0.99               | 1E-6 | 0.67               | 1E-1 | 0.38              | 1.00               | 6E-4 | 0.70               | 3E-2 | 8.2               |
| 4   | 1.00               | 1E-4 | 0.17               | 6E-2 | 1.11              | 1.00               | 3E-3 | 0.18               | 6E-2 | 13.5              |
| 5   | 0.90               | 7E-5 | 0.15               | 8E-2 | 1.17              | 0.98               | 1E-2 | 0.23               | 5E-3 | 30.4              |
| 6   | 0.88               | 7E-3 | 0.07               | 9E-3 | 1.39              | 0.96               | 8E-3 | 0.09               | 1E-3 | 50.2              |

**Proposed Approach**

**CFA**

As an example of the learning process, in Fig. 6a we can see the initial distortion distribution for the case of 6 equally distributed RBFs, which is the first configuration whose approximation error falls under the target error. Fig. 6b represents the same information when the clustering process has ended. We can now see the advantage that we can expect from the fact of making each cluster to have an equal contribution to the total distortion, which is the objective of the proposed clustering algorithm. Finally, 5b represents the approximation of the net using 6 RBFs. We can see how the net is capable of making practically a perfect approximation.

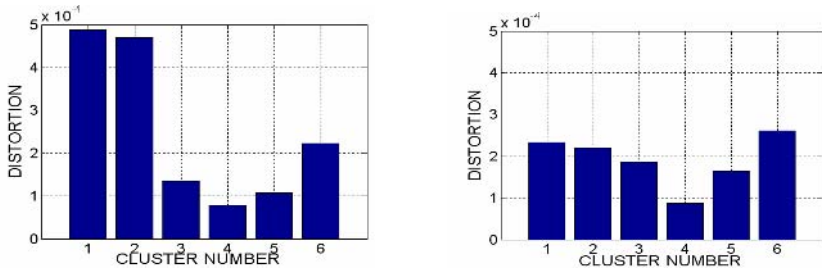


Fig. 6. a) The distortion before the migration

b) The distortion after the migration

## 5 Conclusions

In this paper we have proposed an algorithm of clustering especially suited for function approximation problems. This method calculates the error committed in every cluster using the real output of the RBFN, and not just an approximate value of that output, trying to concentrate more clusters in those input regions where the approximation error is bigger, thus attempting to homogenize the contribution to the error of every cluster. This algorithm is easy to implement and is superior in both performance and computation time to other algorithms such as the CFA method. We have also shown how it is possible to use this algorithm to find the minimal number of RBFs that satisfy a certain error target for a given function approximation problem.

## Acknowledgements

This work has been partially supported by the Spanish CICYT Project TIN2004-01419.

## References

1. Pomares, H., Rojas, I., Ortega, J., González, J., Prieto, A.: "A Systematic Approach to Self-Generating Fuzzy Rule-Table for Function Approximation", *IEEE Trans. Syst., Man, and Cybern.* (2000), Part-B, 30(3) 431-447.
2. Higgins, Ch.: *Classifications and approximation with Rule-Based Networks*. Phd. Thesis (1993), 33-36
3. Chen S., Cowan C. F. N, Grant P. M.: "Orthogonal least squares learning algorithm for radial basis functions networks", *IEEE Trans. Neural Networks* (1991), vol. 2, no. 2, 302-309.
4. Orr M. J. L.: "Regularization in the selection of radial basis function centers", *Neural Computation* (1995), 7(3), 606-623
5. Duda, R. O., Hart, P. E.: *Pattern Classification and Scene Analysis*. New York: Wiley, 1973.
6. Bezdek, J. C.: *Pattern Recognition with Fuzzy Objective Function Algorithms* Plenum, New York, 1981.
7. Russo, M., Patanè, G.: "Improving the LBG Algorithm," in *Lecture Notes in Computer Science*. New York: Springer-Verlag (1999), vol. 1606, 621-630.
8. Gonzalez, J., Rojas, I., Pomares, H., Ortega, J., Prieto, A.: "A new clustering technique for function approximation", *IEEE Trans. Neural Networks* (2002), vol.: 13, no. 1, 132 -142.
9. Pedrycz, W.: "Conditional fuzzy C-means," *Pattern Recognition Lett.*, vol. 17, pp. 625-632, 1996.
10. Runkler, T. A., Bezdek, J. C.: "Alternating cluster estimation: A new tool for clustering and function approximation", *IEEE Trans. Fuzz Syst.* (1999), vol.7, 377- 393.

# Application of ANOVA to a Cooperative-Coevolutionary Optimization of RBFNs

Antonio J. Rivera<sup>1</sup>, Ignacio Rojas<sup>2</sup>, and Julio Ortega<sup>2</sup>

<sup>1</sup>Departamento de Informática, Universidad de Jaén  
arivera@ujaen.es

<sup>2</sup>Departamento de Arquitectura y Tecnología de Computadores,  
Universidad de Granada  
{ignacio, julio}@atc.ugr.es

**Abstract.** In this paper the behaviour of a multiobjective cooperative-coevolutive hybrid algorithm for the optimization of the parameters defining a Radial Basis Function Network developed by our group, is analyzed. In order to demonstrate the robustness of the behaviour of the presented methodology when the parameters of the algorithm are modified, a statistical analysis has been carried out. In the present contribution, the relevance and relative importance of the parameters involved in the design of the multiobjective cooperative-coevolutive hybrid algorithm presented are investigated by using a powerful statistical tool, the ANalysis Of the VAriance (ANOVA). To demonstrate the robustness of our algorithm, a functional approximation problem is investigated.

## 1 Introduction

Evolutionary Computation (EC) [1] is one of the most important paradigms for the design of neural networks. Although this paradigm can be improved for certain types of problems, especially when different specialized subcomponents are involved in the problem solving strategy, as is the case with neural networks.

The Cooperative Coevolutionary paradigm [2] uses individuals that represent only a part of the solution and evolve in parallel, competing to survive but at the same time cooperating to find a common solution. We have proposed [3],[4] a new cooperative coevolutionary approach to find the optimal parameters of a Radial Basis Function Networks (RBFN).

A RBFN [5],[6], implements the function  $f: R^n \rightarrow R$ , that can be described, as it is shown in Figure 1, by the affined mapping:

$$f(\vec{x}) = \sum_{i=1}^m w_i \phi_i(\vec{x}) \quad (1)$$

where the  $m$  radially-symmetric basis functions  $\phi_i$ , are often taken to be translated dilations of a prototype radial basis function  $\phi_i: R^n \rightarrow R$  i.e.  $\phi_i(\vec{x}) = \phi_i(\|\vec{x} - \vec{c}_i\|/d_i)$ ,



where  $\vec{c}_i \in R^n$  is the center of basis function  $\phi_i$ ,  $d_i \in R$  is a dilation of scaling factor for the radius  $\|\vec{x} - \vec{c}_i\|$ , and  $\|\cdot\|$  is typically the Euclidean norm on  $R^n$ . From all the possible choices for  $\phi_i$  [7], the Gaussian function  $\phi(r) = \exp(-r^2)$ , that was first used during the 80s [5], is the one used in this paper as Radial Basis Function (RBF).

Our coevolutionary procedure [3][4] follows the same strategy described in [8] but improves the approach with some new contributions like the definition of credit apportionment and the use of a rule based system (FRBS) to determine the possibility of applying a given operator over a certain RBF.

In order to validate the experimental results of the presented algorithm when the parameters within its design are modified, a precise understanding of the significance of the different alternatives has been carried out by using a statistical tool. The ANalysis Of VAriance (commonly referred to as ANOVA) is one of the most widely used statistical techniques. The theory and methodology of ANOVA was developed mainly by R.A. Fisher during the 1920s [9]. ANOVA belies its name in that it is not concerned with analyzing variances but rather with analyzing the variation in means. ANOVA examines the effects of one, two or more quantitative or qualitative variables (termed factors) on one quantitative response. ANOVA is useful in a range of disciplines when it is suspected that one or more factors affect a response. The ANOVA is essentially a method of analysing the variance to which a response is subject into its various components corresponding to the sources of variation which can be identified.

In this paper the results of our coevolutionary algorithm are tested with ANOVA to demonstrate its robustness.

1. Initialize RBFN
2. Train weights of the RBFN with LMS
3. Evaluate RBFs
4. Select the worst RBFs
5. Apply operators to the worst RBFs
6. Substitute the RBFs that were eliminated
7. Train weights of the RBFN with LMS
8. If the stop-condition is not verified go to step 3, else go to 9
9. Apply Levenberg-Marquardt

**Fig. 1.** Detailed algorithm for the design of the RBFNs

The organization of this paper is as follows. Section II is a brief summary of the proposed algorithm In Section III a statistical analysis ANOVA of the main parameters that define the algorithm is shown. Finally, the conclusions of the paper are discussed in Section IV.

## 2 Proposed Algorithm

The major step of our algorithm, already presented in [3],[4], are shown in Figure 1. In what follows, we provide the main details of these steps.

### 2.1 Evaluate RBFs

Like credit allocation mechanism (value allocated to each individual according to its contribution to the final solution), we define three parameters for each RBF  $\phi_i$ .

- $a_i$ : measures the contribution of the base function to the output of the network.
- $e_i$ : gives the error in the basis function radius.
- $o_i$ : evaluates the overlapping among RBFs.

The contribution,  $a_i$ , of the RBF  $\phi_i, i=1, \dots, m$ , is determined by considering its weight  $w_i$  and the number of points of the training set inside its radius,  $pr_i$ . Thus, a parameter  $ra_i$  can be defined as:

$$ra_i = |w_i| \cdot (pr_i / \overline{pr}) \tag{2}$$

where  $\overline{pr}$  is the average of the values  $pr_i, i=1, \dots, m$ ; and  $a_i$  can be expressed as

$$a_i = ra_i / da \tag{3}$$

where the parameter  $da$  allows the normalization of  $a_i$  inside the interval [0,1]. This normalization is required to assure an adequate level of performance for a FRBS (Fuzzy Rule Based System) used in our procedure (as described bellow). To determine  $da$ , three factors will be taken into account: the average of the  $ra_i$  ( $\overline{ra}$ ), the standard deviation of the current values  $ra_i$  ( $\overline{S}(ra)$ ), and the zone of the interval [0, 1] where we want to move the  $a_i$  values. The aim is to place this new average in the upper zone of the interval [0, 1] so that any individual of the population having a factor  $a_i$  near to this average does not suffer much pressure and can maintain the evolutionary line.

Therefore  $da$  can be expressed as follows:

$$da = \frac{\overline{ra}}{A \cdot (\overline{fra} / B)} \tag{4}$$

where  $\overline{fra} = \overline{S}(ra) / \overline{ra}$ . A and B set the zone of the interval to move the  $a_i$  values.

The error measure,  $e_i$ , for each RBF  $\phi_i$ , is obtained from the normalized root mean square error  $nrmse_i$  of the data points inside the radius of the RBF, as well as the standard deviation error,  $\overline{S}(e_i)$ , also inside the radius of the RBF. Thus, the error  $re_i$  is defined as:

$$re_i = nrmse_i \cdot (\overline{S}(e_i) / \overline{\overline{S}(er)}) \tag{5}$$

where  $\overline{S(er)}$  is the average of the values  $\overline{S(er_i)}$ . In the same way:

$$e_i = re_i / de \quad (6)$$

and the parameter  $de$  can be expressed as:

$$de = \frac{\overline{re}}{C / (fre / B)} \quad (7)$$

where  $\overline{re}$  is the average of the parameters  $re_i$ ,  $\overline{S(re_i)}$  is their standard deviation and  $fre = \overline{S(re_i)} / \overline{re}$ . B and C set the zone of the interval to move the  $c_i$  values.

Any overlapping among the RBF  $\phi_i$  and other RBFs is quantified by using the parameter  $o_i$ . The parameter  $o_i$  is expressed as:

$$o_i = \sum_j^m o_{ij} \quad (8)$$

where  $m$  is the number of RBFs, and  $s_{ij}$  measures the overlapping among the RBFs  $\phi_j$ ,  $j=1, \dots, m$ , and  $\phi_i$ :

$$o_{ij} = \begin{cases} (1 - \|\phi_i - \phi_j\| / d_i) & \text{if } \|\phi_i - \phi_j\| < d_i \\ 0 & \text{other wise} \end{cases} \quad (9)$$

where  $\|\cdot\|$  is the Euclidean distance between the two basis functions, and  $d_i$  is the radius of the basis function  $\phi_i$ .

## 2.2 Applying the Operators to the Set of Worst Basis Functions

In the present implementation of our procedure only two operators are defined to be applied:

- Operator REMOVE: It simply eliminates the RBF used as operand.
- Operator ADAPT: This operator adapts the radius and center of the RBF. These modifications are based on a simplified version of the LMS algorithm [10].

As it has been said, to decide about the application of an operator to a basis function, a FRBS [11] is used. This way, the parameters  $a_i$ ,  $e_i$  and  $o_i$  are the inputs to the fuzzy system. These inputs respectively define the linguistic variables  $va_i$ ,  $ve_i$  and  $vo_i$ . Moreover, two outputs are defined. The output  $vremove$  gives the probability of applying the operator REMOVE, while the output  $vadapt$  gives the probability of applying the operator ADAPT.

The set of linguistic labels {Low, Medium, High} is used for each input. These three linguistic labels define their own fuzzy sets. With respect to the outputs, the set of linguistic labels {Low, Medium-Low, Medium-High, High} is considered. The number of linguistic labels has been empirically determined, with centres and bases directly related to their meaning.

With regards to the inference engine, we have used a Mandani Type one [12]. This reasoning mechanism is configured considering the Max function as the T-conorm and the Min function as the T-norm.

### 3 Analysis of the Experimental Results

To prove the robustness of our algorithm the bidimensional function approximation is considered. Concretely function  $f_4$ , reported in [13] and defined by the following equation, had been chosen:

$$f_4(x_1, x_2) = \frac{1 + \sin(2x_1 + 3x_2)}{3.5 + \sin(x_1 - x_2)} \quad x_1, x_2 \in [-2, 2] \tag{10}$$

For the ANOVA statistical study, the factors considered are:

- *nrbfs*: number of radial basis units in the hidden layer.
- *fda*: this factor is taking into account to measure the influence of *da*, expression (4), that allows the normalization of  $a_i$  inside the interval [0, 1]. The usual value of *fda*, 1.5, is obtained by operating with the constants A and B of the expression (4).
- *fde*: in the same way, this factor measures the influence of *de*, expression (7), that normalizes  $e_i$  inside the interval [0, 1]. The usual value of *fde* is 0,125 and is obtained by operating with the constants B and C of the expression (7).
- *sld*: is a non numeric factor, that contains the linguistic labels and their membership functions of the inputs variables of the FRBS. Usually *sld1* is used.

The response variable used to perform the statistical analysis is the mean square error in the output of the neural network system, when some of the levels of the factor considered vary with respect to a reference design. The changes in the response variable are produced when a new combination of the parameters *fda*, *fde*, *sld* or *nrbfs* are used.

To carry out the statistical study, a selection is made from a set of representative alternatives for each factor to be considered. By analyzing the different levels of each of these factors it is possible to determine their influence on the characteristics of the RBFN output. Table 1 and Figure 2 gives the different levels considered for each factor when carrying out multifactorial ANOVA.

**Table 1.** Levels of each factor considered in the statistical analysis

| Factor     | Level 0     | Level 1     | Level 2     | Level 3 | Level 4 |
|------------|-------------|-------------|-------------|---------|---------|
| <i>Fda</i> | 1.05        | 1.275       | 1.5         | 1.725   | 1.95    |
| <i>Fde</i> | 0.0875      | 0.10625     | 0.125       | 0.14375 | 0.1625  |
| <i>Sld</i> | <i>sld1</i> | <i>sld2</i> | <i>sld3</i> |         |         |

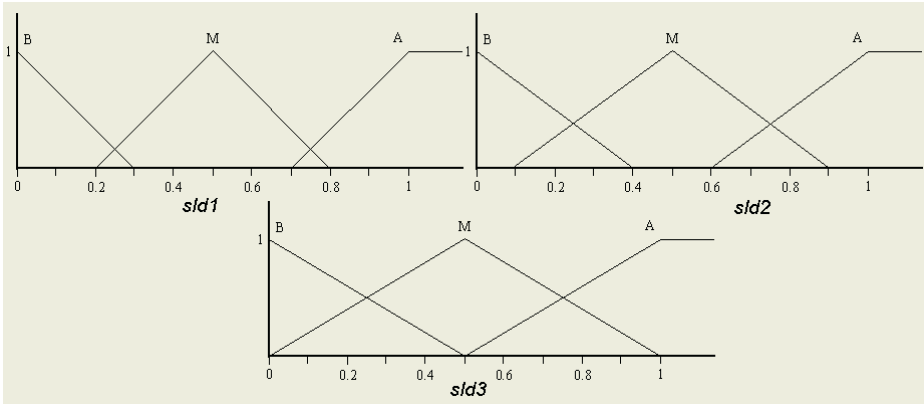


Fig. 2. Levels of factor *sld*

Table 2. ANOVA table of the main factors and iterations

| Source                          | Sum of Squares | DF  | Mean Square | F-Ratio | Sig. Level |
|---------------------------------|----------------|-----|-------------|---------|------------|
| <b>Main Factors</b>             |                |     |             |         |            |
| A: <i>prac</i>                  | 0,00031447     | 4   | 0,00007861  | 0,80    | 0,5259     |
| B: <i>prerr</i>                 | 0,00002238     | 4   | 0,00000559  | 0,06    | 0,9939     |
| C: <i>sld</i>                   | 0,00007844     | 2   | 0,00003922  | 0,40    | 0,6712     |
| D: <i>nrbfs</i>                 | 18,8405        | 3   | 6,28016     | 63913   | 0,0000     |
| <b>Significant Interactions</b> |                |     |             |         |            |
| AB                              | 0,00120626     | 16  | 0,00007539  | 0,77    | 0,7221     |
| AC                              | 0,00031363     | 8   | 0,00003920  | 0,40    | 0,9205     |
| AD                              | 0,00109188     | 12  | 0,00009098  | 0,93    | 0,5222     |
| BC                              | 0,00184766     | 8   | 0,00023095  | 2,35    | 0,0192     |
| BD                              | 0,00133659     | 12  | 0,00011138  | 1,13    | 0,3345     |
| CD                              | 0,00028219     | 6   | 0,00004703  | 0,48    | 0,8241     |
| Residual                        | 0,0220244      | 224 | 0,00009832  |         |            |
| Total                           | 18,869         | 299 |             |         |            |

Table 2 gives the four-way variance analysis for the whole set of combinations for the cooperative-coevolutionary optimization algorithm of RBFN presented. The analysis of the variance table containing the sum of squares, degrees of freedom, mean square, test statistics, etc., represents the initial analysis in a compact form. This kind of tabular representation is customarily used to set out the results of ANOVA calculations. As can be seen from Table 2, only the number of hidden units present statistical relevance, being the remaining non statistically significant factor(e.g, the selection between the different alternatives for the parameter *fda* has no influence on the behaviour of the neural network). These conclusions are also confirmed by the multiple range tables for the different factors (Table 3, Table 4 and Table 5). By

analyzing the different levels of each of the main factors, it is possible to determine their influence on the response of the network, enabling levels with the same response repercussion to be grouped homogeneously. Within the homogeneous groups column, each sub-column represents a different group.

**Table 3.** Multiple Range Test analysis for the variable *fda*

| Level of variable <i>fda</i>                       | LS Media | Homogeneous Groups |
|----------------------------------------------------|----------|--------------------|
| 1,725                                              | 0,283842 | X                  |
| 1,95                                               | 0,284155 | X                  |
| 1,5                                                | 0,28541  | X                  |
| 1,05                                               | 0,286191 | X                  |
| 1,275                                              | 0,286331 | X                  |
| Limit to establish significant differences: 0.0036 |          |                    |

**Table 4.** Multiple Range Test analysis for the variable *fde*

| Labels of variable <i>fde</i>                     | LS Media | Homogeneous Groups |
|---------------------------------------------------|----------|--------------------|
| 0,0875                                            | 0,284689 | X                  |
| 0,14375                                           | 0,285133 | X                  |
| 0,125                                             | 0,285302 | X                  |
| 0,10625                                           | 0,285312 | X                  |
| 0,1625                                            | 0,285493 | X                  |
| Limit to establish significant differences:0.0036 |          |                    |

**Table 5.** Multiple Range Test analysis for the variable *sld*

| Levels of variable <i>sld</i>                      | LS Media | Homogeneous Groups |
|----------------------------------------------------|----------|--------------------|
| <i>sld1</i>                                        | 0,284482 | X                  |
| <i>sld3</i>                                        | 0,285392 | X                  |
| <i>sld2</i>                                        | 0,285683 | X                  |
| Limit to establish significant differences: 0.0028 |          |                    |

**Table 6.** Multiple Range Test analysis for the variable *nrbfs*

| Levels of variable <i>nrbfs</i>                    | LS Media  | Homogeneous Groups |   |   |   |
|----------------------------------------------------|-----------|--------------------|---|---|---|
| 20                                                 | 0,0488078 | X                  |   |   |   |
| 15                                                 | 0,133404  |                    | X |   |   |
| 10                                                 | 0,259084  |                    |   | X |   |
| 5                                                  | 0,699447  |                    |   |   | X |
| Limit to establish significant differences: 0.0032 |           |                    |   |   |   |

Table 6 presents the multiple range test for the several levels of factor corresponding to the number of radial basis functions (*nrbfs*). This table applies a multiple comparison procedure to determine which of the means are significantly different from the others. Four homogeneous groups are identified using columns of X. Within each column, the levels containing X form a group of means within which there are no statistically significant differences. This indicates that this parameter has a knock-on effect on the behaviour of the neural systems, as expected *a priori*, because by increasing the number of hidden neurons the approximation error is decreased.

From Table 3, Table 4 and Table 5, it is clear that there is only one homogeneous group. Therefore, the parameters that are involved in the definition of the presented algorithm are not statistically significant, which demonstrates that the methodology is robust.

## 4 Conclusions

This paper analyzes a cooperative-coevolutionary algorithm in which each individual represents only a part of the solution, indeed just only one hidden neuron for the design of a Radial Basis Function Network. The robustness of our developed procedure is demonstrated with the inclusion of an ANOVA statistical test. Throughout this analysis, the parameters within the proposed cooperative coevolutionary algorithm have been analyzed, demonstrating that even though the nominal values of these parameters are varied, the system is robust and the obtained optimization results remain practically unaltered.

**Acknowledgements.** This paper has been supported by project TIN2004-01419 of the Spanish Ministerio de Ciencia y Tecnología.

## References

1. Bäck, T.; Hammel, U.; Schwefel, H. (1997). "Evolutionary computation: comments on the history and current state". IEEE Transactions on Evolutionary Computation, vol. 1 n. 1 April. Pp. 3-17
2. Potter, M.; De Jong, K. (2000) "Cooperative Coevolution: an architecture for evolving coadapted subcomponents". Evolutionary Computation, 8(1), 1-29.
3. Rivera, A.; Ortega, Julio.; Rojas, I.; del Jesús, M. (2003) "Co-evolutionary Algorithm for RBF by Self-Organizing Population of Neurons". LNCS 2686. pp. 470-477. IWANN 2003.
4. Rivera, A. (2003). "Diseño y Optimización de Redes de Funciones de Base Radial Mediante Técnicas Bioinspiradas". Ph.D. dissertation, Univ. Granada, Spain, Sept.
5. Broomhead, D. and Lowe, D. (1988). "Multivariable functional interpolation and adaptive networks. Complex System. 2. pp 321-355.
6. Orr, M. (1996). "Introduction to radial basis function networks". Technical report. Center for cognitive science. University of Edinburgh.

7. Rojas,I.; Valenzuela,O.; Prieto,A. (1997): "Statistical Analysis of the Main Parameters in the Definition of Radial Basic Function Networks", Lecture Notes in Computer Science, Vol.1240, pp. 882-891, Springer-Verlag, June.
8. Whitehead, B; Choate, T. (1996) "Cooperative-competitive genetic evolution of Radial Basis Function centers and widths for time series prediction". IEEE Trans. on Neural Networks, Vol.7, No.4, pp.869-880. July.
9. Fisher, R.A. (1936) "The Comparison of Samples with Possibly Unequal Variances", Annals of Eugenics 9 174-180 (Also in R.A.Fisher, *Contribution to Mathematical Statistics*, New York: Wiley, 1950).
10. Widrow, B.; Lehr, M.A.(1990). "30 Years of adaptive neural networks: perceptron, madaline and backpropagation". Proc. of the IEEE, v. 78 n. 9, sept.
11. Mendel, J. (1995) "Fuzzy logic system for engineering: a tutorial" Proceedings of the IEEE, vol. 83, n. 3. march
12. Mandani, E.; Assilian, S. (1975) "An experiment in linguistic synthesis with a fuzzy logic controller". Int. J. Man-Machine Stud., v. 7, n. 1, pp. 1-13, 1975
13. Cherkassky, V.; Gehring, D.; Mulier, F. (1996). "Comparison of adaptive methods for function estimation from samples", IEEE Trans. Neural Networks, vol.7, no.4, pp.969-984, July.



# Characterizing Self-developing Biological Neural Networks: A First Step Towards Their Application to Computing Systems

Hugues Berry and Olivier Temam

Alchemy-INRIA Futurs, Parc Club Orsay Université,  
ZAC des vignes, 4 rue Jacques Monod, 91893 Orsay Cedex, France  
{hugues.berry, olivier.temam}@inria.fr

**Abstract.** Carbon nanotubes are often seen as the only alternative technology to silicon transistors. While they are the most likely short-term alternative, other longer-term alternatives should be studied as well, even if their properties are less familiar to chip designers. While contemplating biological neurons as an alternative component may seem preposterous at first sight, significant recent progress in CMOS-neuron interface suggests this direction may not be unrealistic; moreover, biological neurons are known to self-assemble into very large networks capable of complex information processing tasks, something that has yet to be achieved with other emerging technologies.

The first step to designing computing systems on top of biological neurons is to build an abstract model of self-assembled biological neural networks, much like computer architects manipulate abstract models of transistors and circuits. In this article, we propose a first model of the *structure* of biological neural networks. We provide empirical evidence that this model matches the biological neural networks found in living organisms, and exhibits the *small-world* graph structure properties commonly found in many large and self-organized systems, including biological neural networks. More importantly, we extract the simple local rules and characteristics governing the growth of such networks, enabling the development of potentially large but realistic biological neural networks, as would be needed for complex information processing/computing tasks. Based on this model, future work will be targeted to understanding the evolution and learning properties of such networks, and how they can be used to build computing systems.

## 1 Introduction

Carbon nanotubes look like a promising alternative technology to silicon chips because the manufacturing process, possibly based upon self-assembly, will be much cheaper than current CMOS processes. On the other hand, these individual components may turn out to be much slower than current transistors, exhibit lots of manufacturing defects, and may be difficult to assemble into complex and irregular structures like today's custom processors. Current research are focused on building increasingly large structures of carbon nanotubes and understanding how they can be transformed into computing devices.

However, carbon nanotubes, though the most promising and short-term, is not the only possible alternative to silicon chips. Other emerging technologies, even if they are less familiar to chip designers, should be explored as well. In this article, we focus our attention on *biological neurons*. They share some properties with carbon nanotubes: they have a low design cost, but they will provide even slower components, a significant percentage of these components will be similarly faulty, and it will be hard to assemble them into complex, irregular *pre-determined* structures. On the other hand, they have a significant asset over carbon nanotubes: we already know it is possible to self-assemble them into very large structures capable of complex information processing tasks.

While proposing computing structures based on biological neurons may seem preposterous at first sight, G. Zeck and P. Fromherz [12, 30] at the Max Planck Institute for Biochemistry in Martinsried, Germany, have recently demonstrated they can interface standard silicon chips with biological neurons, pass electrical signals back and forth through one or several biological neurons, much like we intend to do with carbon nanotubes, i.e., hybrid carbon nanotubes/standard CMOS chips [13]. Moreover, based on this research work, Infineon (one of the main European chip manufacturers) has recently announced it is investigating a prototype of a chip (called "NeuroChip") that can interconnect a grid of transistors with a network of biological neurons [16], based on Fromherz's research work. So, while we will not claim this research direction should be mainstream, it is certainly worth exploring.

Now, computing machines, such as current processor architectures, are designed using a very abstract model of the physical properties of transistors and circuits. Typically, what processor architects really use (e.g., at Intel or other chip manufacturers) is how many logic gates can be traversed in a single clock cycle, and how many logic gates can be laid out on a single chip. They do not deal with the complex physics occurring at the transistor level, they rely upon a very abstract and simplified model of the undergoing physical phenomena. Similarly, if we want to start thinking about computing systems built upon biological neurons, we must come up with sufficiently abstract models of biological networks of neurons that will enable the design of large systems without dealing with the individual behavior of biological neurons.

The vast literature on artificial neural networks provides little indications on the *structures* of biological neural networks [14]. To understand what kind of computing systems can be built upon biological neurons, we must first understand the kind of *structures* into which biological neurons can self-assemble. Consequently, we have turned to biology for that issue, and the current article is a joint work between computer science and biology research groups. We start with the biological neural network of a small living organism, a worm named *Caenorhabditis elegans*, which has been described in great details in [1, 29].<sup>1</sup> Based on this work, Oshio *et al.* [23] have recently built a database which describes this biological neural network and facilitates its manipulation. Using this map as an oracle, we define a model of network growth in real space and provide

---

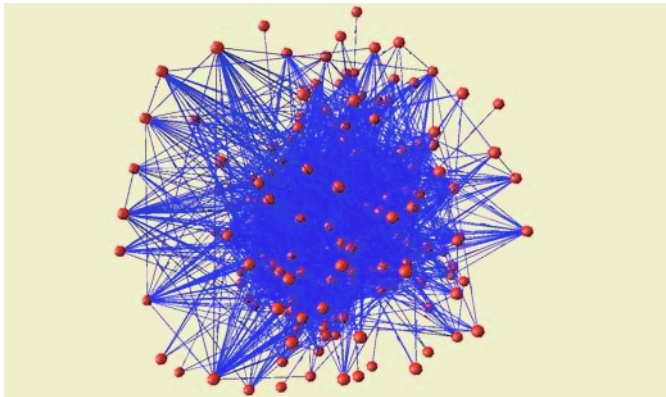
<sup>1</sup> This worm has been subject to intense scientific study as one of the most simple living organism that retains many of the characteristics of complex organisms, such as a brain, learning capabilities, and other physiological similarities. As a side note, it is the first organism which genome has been entirely sequenced.

empirical evidence that the characteristics of networks built upon this model and the above mentioned biological network closely match. Since this model describes the network *growth* using simple local rules, it can be used to represent much larger networks, as would be needed for computing systems. Note that such models had not yet been derived by and are not directly useful to biologists: there are many studies on biological neural networks, but they focus on the identification of *regular* biological networks with clear structures, such as the basic circuit of the visual cortex [10], and they do not account for the seemingly irregular structure of the vast majority of biological neural networks. We provide a *statistical* description of these apparently unstructured biological networks, that can be used as a building block for computing systems studies. Future work will focus on analyzing the evolution and learning properties of neural networks with such structures.

In Section 2, we present the biological neural network of *C. elegans* and study its properties. In Section 3, we build a network model with similar properties, provide empirical evidence that it closely emulates the neural network of *C. elegans*, and provide a detailed comparison of the model and its biological counterpart.

## 2 A Biological Neural Network

*C. elegans* is a small (millimetric) worm with a simple network of 302 neurons. All the connections between its neurons have been mapped [1, 29] and are believed to be relatively well conserved between individual worms. To construct a graph model of this system, we used the electronic database recently published by Oshio *et al.* [23]. A part of this system, comprising 20 neurons and referred to as the "pharyngeal system" is dedicated to control rhythmic contractions of a muscular pump that sucks food into the worm body [1]. This system is almost totally disconnected from the rest of the network. Following Morita *et al.* [21], we neglected here the pharyngeal system and only deal with the remaining 282 neurons. We then further neglected those neurons for which



**Fig. 1.** Visual illustration of the neural network of *C. elegans*. Neurons are displayed as nodes and connections between them are symbolized as links. Spatial positions are arbitrary

no connection had been described, as well as the connections to non specified cells. At the end, the network thus consisted of 265 neurons. Unlike Morita *et al.* [21], we treated each link as directed, i.e., we differentiated links from neuron  $i$  to neuron  $j$  and links from  $j$  to  $i$ ; however, we collapsed multiple identical links into a single one. In opposition to chemical synapses, which are unidirectional connections, some of the neuron connections, called electrical synapses or *gap junctions*, are bidirectional. Here we treated gap junctions as pairs of links with opposite directions. Overall, we obtain 2335 unique links (or 10234 connections if we allow redundant links with the same orientation between two neurons).

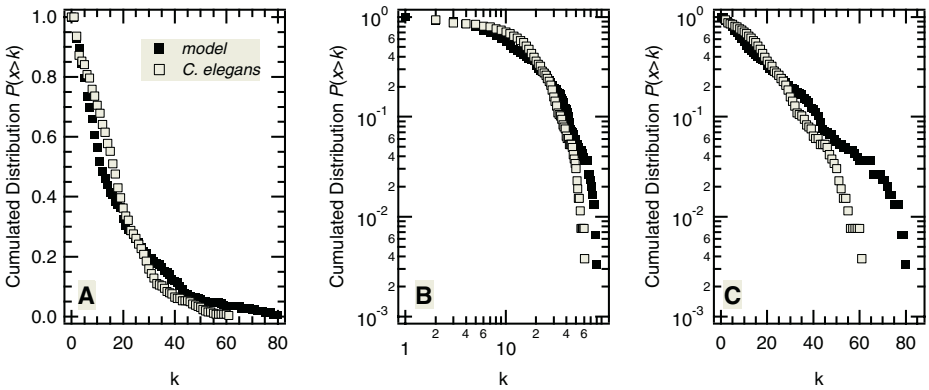
Figure 1 shows a visual illustration of the corresponding neural network. A visual inspection of this figure, especially the peripheral nodes,<sup>2</sup> indicates that the network is rather heterogeneous: strongly connected nodes coexist with sparsely connected ones. We further tried to estimate the nature of the probability distribution of the connectivity (or graph degree), as it plays a fundamental role in characterizing the network type. The probability distribution of the connectivity in *C. elegans* neural network has been controversial. A first study claimed the distribution was compatible with a power-law (graphs determined by power-law distributions are also called “scale-free” graphs) [3]. Not long after, this result was contradicted by an article from H.E. Stanley’s team that studied outgoing and incoming connectivity separately (and ignored gap junctions) and showed that both distributions were exponential, thus excluding scale-free properties [2]. Finally, Morita *et al.* put forward correlations among incoming, outgoing and gap junctions to explain that the total degree (incoming + outgoing + gap junctions) was neither exponential nor displayed a clear power law decrease [21]. Figure 2 presents the distribution of the total connectivity for *C. elegans* neural network (white squares). The center panel is a replot of the left one, in log-log coordinates. A power law decrease would yield a straight line in this representation, which is clearly not the case. Further, the right panel is another replot of the same data, in log-linear coordinates. Here, a straight line would indicate an exponential decrease. A clear exponential decrease is not apparent from this panel. Thus, our results confirm that connectivity distribution for *C. elegans* neural network is neither scale-free nor clearly exponential.

We will see in the next section that additional network characteristics are necessary to emulate this network structure; more importantly, we will extract the simple local rules governing the network growth, enabling the development of potentially large but realistic biological networks using the same rules.

### 3 A Model of Biological Neural Networks

**Small-World graphs and neural networks.** The global behavior of most large systems emerges from local interactions between their numerous components. At an abstract level, these systems can often be viewed as graphs, with each link representing the interaction between two components. Such graph theory approaches have proven success-

<sup>2</sup> On paper, the core of the network structure is barely visible, but on a screen, it can be inspected through zooming and 3D manipulations; however the peripheral structure is the same as the core structure.



**Fig. 2.** Cumulated distributions of the connectivity,  $P(x > k)$ , where  $k$  is the connectivity (i.e., the number of links to and from each node), for the *C. elegans* neural network (white squares); the black squares curve corresponds to the network model later introduced and commented in Section 3. The same data are presented as a linear-linear (A), a log-log (B) and log-linear plot (C)

ful in understanding the global properties of several complex systems originating from highly disparate fields, from the biological to social and technological domain. Hence the same (or similar) reasonings can be applied to understand cell metabolism [17], the citation of scientific articles [22], software architecture [27], the Internet [3] or electronic circuits [5]. A common feature of all these networks is that their physical structure reflects their assembly and evolution, so that their global features can be understood on the basis of a small set of simple local rules that control their growth. The most common statistical structures resulting from these local rules are the so-called *small-world* and *scale-free* networks. Small-world properties characterize networks with both small average shortest path and a large degree of clustering, while scale-free networks are defined by a connectivity probability distribution that decreases as a power law (more formal definitions will be given in the following). At a much coarser grain, graph theory methods have recently been applied to networks of cortical areas [26, 11],<sup>3</sup> i.e., not networks of neurons but networks of neuron *areas*, with the prospect of understanding the network functions. Since we target the characterization of networks of biological *neurons*, we study the neural network of the millimetric worm *C. elegans* at the level of individual *neurons*, and attempt to derive a network growth model that closely emulates it.

Most complex networks can be categorized into four families [4]. In *random networks* (also known as Erdős-Rényi graphs), two nodes  $i$  and  $j$  are connected with a predefined probability independently of all others. These graphs are characterized by short paths between two nodes (denoted  $\lambda$ ) and a low clustering (denoted  $\langle C \rangle$ ). On the opposite, *regular graphs* (where each node has the same connectivity) are characterized by a high clustering and a large average shortest path. Between these two extremes,

<sup>3</sup> Cortical areas are functionally related zones of the cortex that contain approximately  $10^8$  neurons.

*small-world* graphs display both small average shortest paths and a high degree of clustering. For most small-world networks,  $P(k)$ , i.e., the probability distribution of the connectivity  $k$ , decreases very quickly (exponentially) beyond the most probable value of  $k$ , which thus sets the connectivity scale. However, in some graphs (such as the Internet),  $P(k)$  decreases as a power-law of  $k$  ( $P(k) \propto k^{-\gamma}$ ), i.e., in a much slower way. In this case, nodes with a very high connectivity (hubs) can also be present with a significant probability so that the connectivity does not display a clear scale, hence the term “scale-free” networks.

We now formally introduce the parameters of a network model. Besides the number of nodes  $N$  and number of links  $K$ , the structural characteristics of complex networks are mainly quantified by their link density  $\rho$ , average connectivity  $\langle k \rangle$ , connectivity distribution  $P(k)$ , average shortest path  $\lambda$  and average clustering coefficient  $\langle C \rangle$  [25]. The network density  $\rho$  is the density of links out of the  $N(N - 1)$  possible directed links<sup>4</sup> (recall multiple links between two nodes are considered a unique link and self-connections are forbidden)

$$\rho = K / (N^2 - N) \quad (1)$$

The connectivity (or *degree*)  $k_i$  of node number  $i$  is the number of links coming from or directed to node  $i$ .  $P(k)$  is the probability distribution of the  $k_i$ 's and  $\langle k \rangle$  their average over all the nodes in the network. Let  $d(i, j)$  be the shortest path (in number of neurons) between neuron  $i$  and  $j$ , then  $\lambda$  is its average over the network

$$\lambda = 1 / (N^2 - N) \sum_{i,j} d(i, j) \quad (2)$$

The clustering coefficient of a node  $i$  with  $k_i$  (incoming plus outgoing) connected neighbors is defined by

$$C_i = E_i / (k_i^2 - k_i) \quad (3)$$

where  $E_i$  is the number of connections among the  $k_i$  neighbors of node  $i$ , excluding the connections between a neighbor and node  $i$  itself. The average clustering coefficient  $\langle C \rangle$  is the average of the  $C_i$ 's over all nodes and expresses the probability that two nodes connected to a third one are also connected together (degree of cliquishness).

The main structural characteristics of the *C. elegans* neural network are indicated in Table 1. Compared to a random network with the same density, this neural network has a similar average shortest path but the clustering has increased almost fivefold. This means that, in the *C. elegans* neural network, one neuron can reach any other neuron in only three connections on average. This is a clear sign of small-world properties. Considering the network is treated here as a directed graph, these results are coherent with previously published estimates [28, 9].

**In biological neural networks, distance matters.** In order to design a network growth model of the *C. elegans* neural network that achieves small-world properties, we have found that taking into account the physical *distance* between two nodes (neurons) is

<sup>4</sup> Each node can have at most  $N - 1$  outgoing links, so the maximum number of links is  $N(N - 1)$ .

**Table 1.** Structural characteristics of the neural network of *C. elegans* shown in Figure 1, a comparable random (Erdős-Rényi) network and the network obtained with the proposed growth model. The data for the random network are averages over 20 network realizations. See text for definition of the listed properties

| Network           | $\rho$ | $\langle k \rangle$ | $\lambda$ | $\langle C \rangle$ |
|-------------------|--------|---------------------|-----------|---------------------|
| <i>C. elegans</i> | 0.033  | 17.62               | 3.19      | 0.173               |
| random            | 0.033  | 17.62               | 2.75      | 0.0352              |
| model             | 0.030  | 17.73               | 3.37      | 0.175               |

critical. Most network growth models do not consider this parameter [19]. For instance, most scale-free networks are obtained through a preferential attachment rule which postulates that new nodes are linked to the already most connected nodes [3]. Not only this development rule implies some global control mechanism (i.e., a node must somehow know which are the most connected nodes) which seems unlikely in the case of a neuronal system, but it also implies that long connections are just as likely as shorter ones. Similar arguments can be opposed to the Watts-Strogatz rewiring algorithm that generates small-world networks through addition of long-range connections to a pre-existing regular circular network [28]. An improved variation of the Watts-Strogatz algorithm restricts rewiring to a local spatial neighborhood around each node [9] thus implicitly introducing the distance factor. However, these last two models are not and cannot evolve into network growth models and thus do not provide a biologic realistic metaphor. In opposition to these models, we address in the present work the specific case of biological neuron network *growth* in real three dimensional space.

Several observations in biology further support the key notion of physical distance. Long distance connections are expensive in biological neural networks because they imply large volumes of metabolically active tissue to be maintained and long transmission delays [8], just like links between two nodes in Internet or in airport transportation systems are more costly. Moreover, wiring length optimization seems to be a crucial factor of cortical circuit development [6, 7]. The network structure itself depends on the wiring length. For instance, small-world properties (as well as, under some circumstances, scale-free connectivity [15]) have been shown to emerge naturally upon minimization of the euclidian distance between nodes [20]. Furthermore, Kaiser et al. [18] have recently shown that network structure during growth in a metric space is influenced by neuron density (number of neurons per unit volume) when growth occurs in a spatially constrained domain.

**A network growth model for *C. elegans*.** We now propose a network growth model in a three-dimensional space. Neurons are abstracted as cubical volumes of unit size. The position of each neuron on the cubic lattice is defined by the integer coordinates  $(i, j, k)$  of its center of mass and spans over the volume comprised between  $(i - 1/2, j - 1/2, k - 1/2)$  and  $(i + 1/2, j + 1/2, k + 1/2)$ . The lattice dimensions are  $L_x, L_y$  and  $L_z$ , defining a volume of  $L_x \times L_y \times L_z$  unit sizes. Each step of the growth algorithm consists of six elementary substeps:

1. choose a neuron  $n$  at random among the neurons already connected in the network (*origin* neuron). Let  $(i, j, k)$  be the spatial coordinates of  $n$  on the lattice.

2. Then choose a *destination* site  $(i', j', k')$  at distance  $d$  with probability

$$P(d) = 1/\xi \exp(-d/\xi) \quad (4)$$

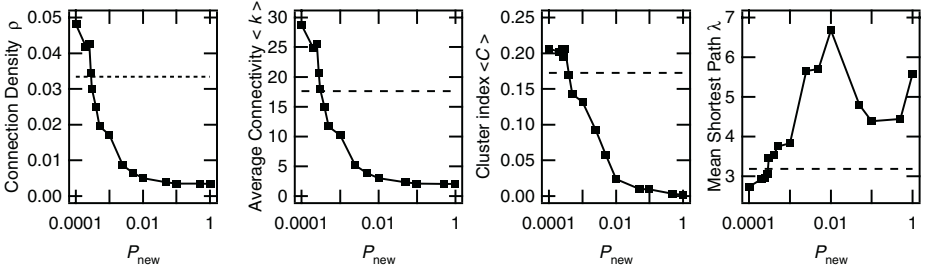
where  $d$  is the euclidian distance between  $(i, j, k)$  and  $(i', j', k')$ , and  $\xi$  is a parameter that sets the average connection distance. If the chosen destination site is located outside the lattice borders, go back to substep 1. Thanks to the exponential distance distribution, the probability to create a connection of a given length (at a certain distance) decreases rapidly with the length, which accounts for wiring minimization. Note that, in biological neural networks, new connections are established through cell outgrowths (neurites) from existing neurons; these outgrowths are guided by gradients of chemical concentration which similarly decay rapidly with distance. We also tested other probability distributions, such as power-laws, and obtained similar results.

3. If a neuron  $n'$  of the network already exists at the destination site  $(i', j', k')$ , a connection is created between  $n$  and  $n'$ .
4. If there is no neuron at the destination site, a new neuron  $n'$  is placed at the destination site (and a connection is created between  $n$  and  $n'$ ) with probability  $P_{new}$ ; the value of  $P_{new}$  is discussed below.
5. If a connection has been created during one of the two preceding steps, its direction ( $n \rightarrow n'$  or  $n' \rightarrow n$ ) is chosen with probability  $P_{n \rightarrow n'} = k_n^{out} / (k_n^{out} + k_n^{in})$  where  $k_n^{out}$  and  $k_n^{in}$  are respectively the outdegree and indegree of neuron  $n$ , i.e., the number of connections with  $n$  as starting node and  $n$  as destination node. This probability distribution reflects the property that, in *C. elegans*, strongly connected neurons (hubs) exhibit asymmetric fractions of incoming/outgoing connections: either much more incoming or much more outgoing connections. The *joint degree distribution matrix* of the *C. elegans* network shown in Figure 4A highlights this characteristic: each matrix element  $m_{ij}$  corresponds to the number (coded using a gray scale) of neurons with  $i$  incoming and  $j$  outgoing connections, so that a scattered plot indicates an asymmetrical repartition of incoming and outgoing connections for many neurons; hubs are the dots located farthest from the origin (lots of connections). We can observe that several of the hubs are located far from the diagonal, hence the asymmetry.
6. go back to substep 1.

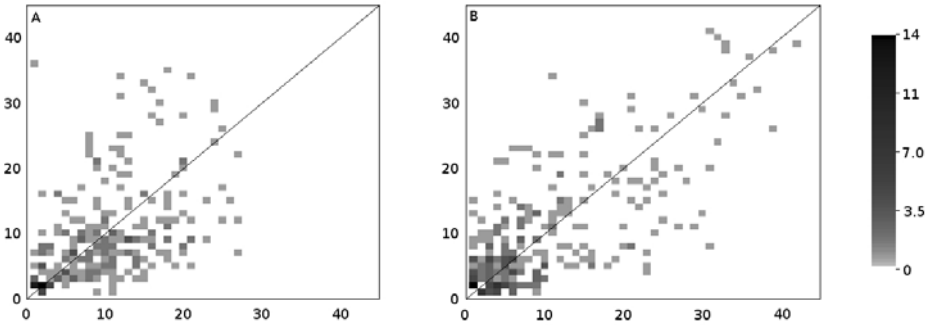
This algorithm iterates until the network contains a prescribed number of neurons  $N$ . In this study, we set  $N = 300$ .

**New neurons are unlikely to be created in already cluttered areas.** If one sets  $P_{new} = 1$ , i.e., a new neuron is certain to be created in a currently empty location, our algorithm is essentially a three-dimensional extension of the model recently proposed by Kaiser and Hilgetag [18]. As demonstrated by these authors, networks obtained with  $P_{new} = 1$  progressively acquire small-world properties when neuron density approaches 1 (i.e. when  $N \rightarrow L_x \times L_y \times L_z$ ). In this case however, the average connectivity and the connection density (as well as the joint degree distribution matrix, see below) remains severely smaller than observed in *C. elegans* (e.g., we could not achieve an average connectivity higher than 6-7).





**Fig. 3.** Structural characteristics of the networks obtained with the proposed model as a function of the probability  $P_{new}$  that a new neuron connects to the network. The dashed line indicates the corresponding value found for the network of *C. elegans*. Other parameters were  $L_x \times L_y \times L_z = 51 \times 51 \times 51$  (neuron size units),  $\xi = 10$ ,  $N = 300$



**Fig. 4.** Joint degree distribution matrices for the neural network of *C. elegans* (A) and a model neural network obtained with  $P_{new} = 0.00028$  (B). Each element  $\{m_{ij}\}$  of the matrices is color-coded with a grey level proportional to the number of neurons displaying  $i$  incoming and  $j$  outgoing links, as shown in the colorbar on the right. The straight line simply aims at highlighting the symmetry axis denoted by the diagonal. All network parameters are the same as in Figure 3

Decreasing  $P_{new}$  breeds significantly more realistic results. As seen in Figure 3, connection density, average connectivity and clustering index increase as  $P_{new}$  decreases while the average shortest path  $\lambda$  decreases. Thus, decreasing  $P_{new}$  yields networks with increasingly strong small-world properties together with increasingly high average connectivity. Furthermore, Figure 4B indicates that our model indeed yields hubs with asymmetric connectivity, as observed for *C. elegans* in Figure 4B. Taken together, the results of Figure 3 and 4 show that all the studied structural properties of the networks obtained with our algorithm match that of *C. elegans* neural network for  $P_{new} \approx 3 \times 10^{-4}$ . A biological interpretation of this value is that natural neural networks would be very reluctant to admit new neurons in the network (as only 1 contact out of  $\approx 3000$  would be statistically successful). Interestingly, recent results in neurobiology suggest that the lack of neural turnover and/or replacement of injured neurons in the adult brain is not due to the absence of potentially competent cell, but, more probably, to a strong reluctance of the neurons to accept newcomers into an already

established neural network [24]. In light of these findings, our results suggest that this strong reluctance could be one of the factors inducing the high average connectivity observed in biological neural networks, i.e., if new neurons can hardly emerge in already cluttered areas, connections are mostly drawn among existing neurons, hence the high connectivity. Since we made the observation that  $P_{new}$  must be very small while fine-tuning the model and before being aware of these recent neurobiological findings, we interpreted this correlation as additional evidence of the validity of our model.

Finally, we can note that the model network characteristics match the *C. elegans* network characteristics, as shown in Table 1. And, in spite of some discrepancies due to the relatively small number of neurons for high connectivity values, the connectivity distribution of the model network is also fairly close to that of the *C. elegans* network, as shown in Figure 2, see the black squares curve. In particular, the model network distribution is neither a power-law, nor clearly exponential (though it is closer to an exponential decrease than that of *C. elegans*). We can thus conclude that our model closely emulates the structure of the biological network of *C. elegans*.

## 4 Conclusions and Future Work

Based on empirical data of a tiny organism, we have elaborated a model for biological neural networks. In agreement with previous works [28], we have found that the neural network of *C. elegans* has a graph structure with *small-world* properties, like many complex systems found in nature. Because the model defines the network *growth* properties, we can now use it to characterize the large neural networks needed for achieving computing tasks. The next step will consist in studying how this network structure affects the learning capabilities and characteristics of neural networks. We then intend to pursue two research directions.

A further step will consist in improving the model accuracy/realism by integrating known but abstract characteristics of the behavior of individual neurons. Finally, through this combined model, we will investigate the application of such biological neural networks to computing tasks, assuming the experimental setups described in [12]. In this perspective, our aim is to obtain a sufficiently abstract model of biological networks of neurons that will enable the design of large systems without dealing with the individual behavior of biological neurons. We think that the availability of such abstract models will be a crucial chokepoint that will have to be overcome if we want to build computing systems using real biological neurons.

In parallel, we intend to use the model developed in this article for biology-oriented studies, especially to investigate how evolution combined with such network structures can foster the emergence of new simple functions within a neural network.

## References

1. D.G. Albertson and J.N. Thomson. The pharynx of *Caenorhabditis elegans*. *Phil. Trans. R. Soc. London B*, 275:229–325, 1976.
2. L.A.N. Amaral, A. Scala, M. Barthélemy, and H.E. Stanley. Classes of small-world networks. *Proc. Natl. Acad. Sci. USA*, 10:11149–11152, 2000.

3. A.L. Barabasi and R. Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.
4. A.L. Barabasi and Z.N. Oltvai. Network biology: understanding the cell's functional organization. *Nature Rev. Genetics*, 5:101–113, 2004.
5. R. Ferrer Cancho, C. Janssen, and R.V. Solé. The topology of technology graphs: Small world patterns in electronic circuits. *Phys. Rev. E*, 64:32767, 2001.
6. C. Cherniak. Component placement optimization in the brain. *J. Neurosci.*, 14:2418–2427, 1994.
7. C. Cherniak. Neural component placement. *Trends Neurosc.*, 18:522–527, 1995.
8. D.B. Chklovskii, T. Schikorski, and C.F. Stevens. Wiring optimization in cortical circuits. *Neuron*, 34:341–347, 2002.
9. J. W. Clark and A. T. Eggebrecht. The small world of the noble nematode *Caenorhabditis elegans*. In J. da Providencia and F. B. Malik, editors, *Condensed Matter Theories*, volume 18. Nova Science Publishers, Hauppauge, N.Y. to appear.
10. R. Douglas and K. Martin. Neocortex. In G. M. Shepherd, editor, *The synaptic organization of the brain*, pages 459–509. Oxford University Press, Oxford, 1998.
11. V.M. Eguiluz, D.R. Chialvo, G. Cecchi, M. Baliki, and A.V. Apkarian. Scale-free brain functional networks. *Phys. Rev. Lett.*, 94:018102, 2005.
12. P. Fromherz. Neuroelectronic interfacing: Semiconductor chips with ion channels, nerve cells, and brain. In Rainer Waser, editor, *Nanoelectronics and information technology*, pages 781–810. Wiley–VCH, Berlin, 2003.
13. Seth Copen Goldstein and Mihai Budiu. NanoFabrics: Spatial computing using molecular electronics. In *Proceedings of the 28th International Symposium on Computer Architecture 2001*, 2001.
14. S. Haykin. *Neural Networks*. Prentice Hall Intl, London, UK, 2nd edition, 1999.
15. R. Ferrer i Cancho and R.V. Solé. Optimization in complex networks. In *Statistical physics of complex networks, Lecture notes in physics*, pages 114–125. Springer, Berlin, Germany, 2003.
16. Infineon. Neurochip with integrated electronics in research phase. [http://www.infineon.com/cgi/ecrm.dll/jsp/showfrontend.do?lang=EN & channel\\_oid=-11398](http://www.infineon.com/cgi/ecrm.dll/jsp/showfrontend.do?lang=EN&channel_oid=-11398), 2003.
17. H. Jeong, B. Tombor, R. Albert, Z.N. Oltvai, and A.L. Barabasi. The large-scale organization of metabolic networks. *Nature*, 407:651–654, 2000.
18. M. Kaiser and C.C. Hilgetag. Spatial growth of real-world networks. *Phys. Rev. E*, 69:036103, 2004.
19. M. Kimura, K. Saito, and N. Ueda. Modeling of growing networks with directional attachment and communities. *Neural Networks*, 17:975–988, 2004.
20. N. Mathias and V. Gopal. Small worlds: how and why. *Phys. Rev. E*, 63:021117, 2001.
21. S. Morita, K.-I. Oshio, Y. Osana, Y. Funabashi, K. Oka, and K. Kawamura. Geometrical structure of the neuronal network of *Caenorhabditis elegans*. *Physica A*, 298:553–561, 2001.
22. M.E.J. Newman. The structure of scientific collaboration networks. *Proc. Natl. Acad. Sci. USA*, 98:404–409, 2001.
23. K. Oshio, Y. Iwasaki, S. Morita, Y. Osana, S. Gomi, E. Akiyama, K. Omata, K. Oka, and K. Kawamura. Database of synaptic connectivity of *C. elegans* for computation. Technical Report 3, CCeP, Keio Future, Keio University, Japan, 2003. The connectivity database is available online at [http://ims.dse.ibaraki.ac.jp/research/database\\_en.html](http://ims.dse.ibaraki.ac.jp/research/database_en.html).
24. P. Rakic. Immigration denied. *Nature*, 427:685–686, 2004.
25. O. Sporns. Graph theory methods for the analysis of neural connectivity patterns. In R. Kötter, editor, *Neuroscience Databases: A Practical Guide*, pages 171–186. Klüwer, Boston, MA, 2002.

26. O. Sporns, D.R. Chialvo, M. Kaiser, and C.C Hilgetag. Organisation, development and function of complex brain networks. *Trends Cog. Sci.*, 8:418–425, 2004.
27. S. Valverde, R. Ferrer Cancho, and R.V. Solé. Scale-free networks from optimal design. *Europhys. Lett.*, 60:512–517, 2002.
28. D.J. Watts and S.H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393:440–442, 1998.
29. J.G. White, E. Southgate, J.N. Thomson, and S. Brenner. The structure of the nervous system of the nematode *caenorhabditis elegans*. *Phil. Trans. R. Soc. London B*, 314:1–340, 1986.
30. G. Zeck and P. Fromherz. Noninvasive neuroelectronic interfacing with synaptically connected snail neurons immobilized on a semiconductor chip. *Proc. Natl. Acad. Sci. USA*, 98(18):10457–10462, 2001.

# Cooperative Bees Swarm for Solving the Maximum Weighted Satisfiability Problem

Habiba Drias<sup>1</sup>, Souhila Sadeg<sup>2</sup>, and Safa Yahi<sup>2</sup>

<sup>1</sup>INI, National Institute of Informatics

<sup>2</sup>USTHB, Computer Science department,

Laboratory of research in Artificial Intelligence,

BP 32 El-Alia Ba-Ezzouar, 16111 Algiers, Algeria

Drias@wissal.dz

**Abstract.** Solving a NP-Complete problem precisely is spiny: the combinative explosion is the ransom of this accurateness. It is the reason for which we have often resort to approached methods assuring the obtaining of a good solution in a reasonable time. In this paper we aim to introduce a new intelligent approach or meta-heuristic named “Bees Swarm Optimization”, BSO for short, which is inspired from the behaviour of real bees. An adaptation to the features of the MAX-W-SAT problem is done to contribute to its resolution. We provide an overview of the results of empirical tests performed on the hard Johnson benchmark. A comparative study with well known procedures for MAX-W-SAT is done and shows that BSO outperforms the other evolutionary algorithms especially AC-SAT, an ant colony algorithm for SAT.

## 1 Introduction

In artificial intelligence, the class of the NP-Complete problems dons a fundamental importance. Indeed, it includes a considerable number of eminent problems as well on the theoretical plan as on the convenient plan.

The maximum weighted satisfiability problem or MAX-W-SAT is an NP-Complete Problem. It is central in both computational complexity theory and artificial intelligence discipline. It causes a particular interest in various domains. An efficient resolution for this problem permits to solve several other problems in automatic reasoning, in mathematical logic, in the theory of the graphs, in VLSI ... etc.

Swarm intelligence is an axis of research that has appeared recently in artificial intelligence. Its main concern is the design of intelligent methods based on the behaviour of social bugs. In spite of the simplicity of their individual behaviours, bugs such as bees achieve, thanks to their cooperation, very complex tasks that an individual endowed with a sophisticated intelligence would be enable to accomplish alone.

This manner to conceive solutions to problems, as being the result of the cooperation of several individuals, give birth to powerful and very efficient meta-heuristics such as the Ant Colony Optimization.

Our work consists in developing a new meta-heuristic that we have called "Bees Swarm Optimization" (BSO for short), It is inspired by the behaviour of the real bees.

We adapt it thereafter to the MAX-W-SAT problem which is described in the next section.

## 2 The MAX-SAT Problem

The problem of Satisfiability, SAT in abbreviation, is the dorsal thorn of the NP-Completeness discipline. In spite of the considerable number of the studies dedicated to it, it remains for many researchers one of their important preoccupations. SAT is defined by the following components :

- $X = \{x_1, x_2, \dots, x_n\}$ , a set of  $n$  Boolean variables.
- $C = \{c_1, c_2, \dots, c_m\}$ , a set of  $m$  clauses where:
  - every clause is a severance of literals.
  - every literal is a variable with or without negation.
- $D$ , the SAT data is a conjunction of clauses of  $C$ .

The problem consists in determining if an assignment to the variable  $x_i$  of  $X$  exists such that the data  $D$  is satisfied (estimated to TRUE). SAT is a decision problem that admits as an answer ‘Yes’ if an assignment to the variables that satisfies all the clauses exists or ‘No’ if such an assignment doesn't exist. Nevertheless, if the answer is by the negative, an important question arises: What is the maximum number of clauses that can be satisfied simultaneously? From this question, comes the definition of the problem ‘Maximum Satisfiability’, or in short, MAX-SAT. If besides, we associate to every clause  $c_i$  a weight  $w(c_i)$ , the objective comes back thus to find an assignment to the variable  $x_i$  of  $X$  that maximizes the sum of the weights of the clauses satisfied simultaneously, from where arises the definition of the problem Weighted MAX-SAT or MAX-W-SAT.

$$c_1 = \bar{x}_1 + x_2 \qquad w(c_1)=10$$

$$c_2 = \bar{x}_1 + \bar{x}_3 + x_4 \qquad w(c_2)=20$$

$$c_3 = x_4 \qquad w(c_3)=25$$

The negation of  $x_i$  is denoted by  $\bar{x}_i$ , the Boolean disjunction operator by  $+$  and the weight of clause  $c_i$  by  $w(c_i)$ . If  $(x_1, x_2, x_3, x_4) = (0, 0, 0, 1)$  then, the sum of the weights of the clauses satisfied simultaneously reaches its maximum.

## 3 Swarm Intelligence

The swarm intelligence domain is about emerged collective intelligence coming from simple agents acting together. The source of inspiration is social insects like for instance ants and bees. How can the interaction of a lot of simple ants create complicated nests and social structures? Is it possible to use some of the similar techniques in computer algorithms and robotics?

The societies of bugs have a remarkable capacity to solve problems in a manner which is very supple (the colony adapts to the abrupt changes of environment) and robust (the colony continues to function even when some individuals fail to accomplish their tasks). The daily problems solved by a colony are numerous and of very varied nature: research of food, construction of the nest, distribution of work and allowance of the tasks between the individuals,... etc.

In fact, this collective behaviour, often complex, is the fruit of an aggregation of individual behaviours dictated by very simple rules. It presents an auto-organized working model, based on a decentralized logic, founded on the cooperation of units having only local information.

This phenomenon, if a long time ago, ethnologists observed it, the programmers and the engineers had difficulties to appropriate it. The latter could transform models of the collective behaviour of social bugs in powerful methods for the design of algorithms for combinative optimization. A new domain of research appeared. It has for goal to transform the knowledge that the ethnologists have about the collective capacities of resolution of problems by the social bugs, in efficient techniques of problems solving while offering a high degree of flexibility and hardiness: It is *swarm intelligence*.

Among the techniques of intelligence in swarm, some of them have arrived to maturity. The algorithms of control and optimization inspired in particular from models of collective research of food among the ants, have known an unexpected success. These techniques are used in several application domains like the industrial computing, the robotics, the data analysis, the telecommunication network and the combinative optimization [1].

## 4 The BSO Meta-heuristic

In 1946 *Karl Von Fris* has decoded the language of the bees. He has observed that it is through the dance, that a bee communicates to its fellows upon its return to the hive, the distance, the direction and the wealth of the food source.

Bees of a same colony visit more than a dozen of potential exploitation areas. But the colony concentrates its efforts of harvest on a small number among them, the richest and the easiest of access. In addition, numerous observations make appear that a colony can displace its exploitation of a source quickly to another one.

In their experience made in 1991, Seely, Camazine and Sneyd have shown that when a colony of bees has the choice between the exploitation of two sources of food where the concentration in sugar is very unequal (1M and 2.5 M respectively) and situated in a diametrically opposite manner to the hive, one to the north and the other to the south, the colony is going to concentrate its effort of harvest on the richest. We explain that by the fact that the bee follows the bee which does the most vigorous dance, therefore the one that indicates the place of the richest source of food [2].

The meta-heuristic "Bees Swarm optimization" is inspired by the behaviour described above. It is based on a behaviour of swarm of artificial bees cooperating together to solve a problem.

First, a bee named *BeeInit* assigns to find a solution presenting good features that we call *Sref* and from which the other solutions of the research space are determined via a certain strategy. The set of these solutions is called *SearchArea*. Then, every bee will consider a solution from *SearchArea* as its starting point in the search. After accomplishing its search, every bee communicates through a table named *Dance* to its fellows the best solution visited. One of the solutions of this table will become the new solution of reference for the next iteration of the process. In order to avoid cycles, the solution of reference is stored every time in a taboo list.

The choice of the reference solution is made, first according to the quality criterion. However, if after a period of time, the swarm notes that the solution doesn't progress in term of quality, it integrates a second criterion of diversity that will allow it to escape from the region where it is possibly incarcerated to another one.

### General algorithm of BSO

```

begin
 Let Sref be the solution found by BeeInit;
 While not condition of stop do
 begin
 insert Sref in taboo list;
 determine SearchArea from Sref;
 affect a solution of SearchArea to each bee;
 for each Bee K do
 begin
 search starting with the solution
 affected to it;
 store the result in the table Dance;
 end;
 Choose the new solution of reference Sref;
 end;
 end;
end;

```

## 5 BSO for Solving Max-W-SAT

The adaptation of our meta-heuristic to MAX-W-SAT requires the design of the following components: The artificial world where the bees lives, the fitness function that evaluates solutions, the initial solution *Sref*, the strategies to determine the set of solutions *SearchArea* from *Sref*, the search procedure performed by each artificial bee, the quality and the diversity strategies and the choice rules of the reference solution *Sref* allowing the iteration of the process.

**The Artificial World or Search Graph.** According to the nature of MAX-W-SAT problem, a solution is a chain of  $n$  bits written as  $\{0,1\}^n$  and corresponding to a truth assignment of variables. The search graph is constituted by all such solutions whose



total number is equal to  $2^n$ . A move from one solution to another one consists in inverting a bit from the current solution. The total number of the moves is equal to  $n2^{n-1}$ .

The food sources are represented by the target solutions. The artificial bees are supposed to intensify the search in *SearchArea* in order to find a solution maximizing the fitness function. The artificial agents cooperate in order to emerge the optimal or a good quality solution.

**The Initial Solution.** Initially, *BeeInit* builds the solution of reference *Sref* via a heuristic. The heuristics that we have used are *John1*, *John2* [5] and the local search.

**The Solution Quality.** it is valued by the fitness function called *f* and expressed in MAX-W-SAT as the sum of the weights of the clauses that are satisfied simultaneously by the solution.

**The Degree of Diversity.** The degree of diversity offered by a solution *S* is measured by the minimum of the distances between *S* and the elements in the *Taboo* list. It is given by:

$$diversity(S) = \text{Min} \{ d(S,T) / T \in \text{Tabou} \} .$$

The distance between two solutions *S* and *T* is given by the distance of Hamming, that is :

$$d(S,T) = \sum_{i=1}^n S_i \oplus T_i$$

$$\text{where: } a \oplus b = \begin{cases} 0 & \text{if } a=b \\ 1 & \text{otherwise} \end{cases}$$

**The Determination of Search Area.** The region of exploitation *SearchArea* is represented by a set of *N* solutions (*N* being the number of bees constituting the swarm). Each of these solutions is calculated from *Sref* while reversing  $1/Flip$  its variables. The choice of the parameter *Flip* is not obvious since it determines the number of variables to reverse from *Sref*. Indeed, a too small value of this number implies that *Sref* is probably the local optimum of the new region of exploitation. Therefore the probability of an improvement is very weak. On the other hand, if this value is too important, the swarm will move away from the region containing *Sref* with the risk to lose good solutions. To proceed to the inversions, we propose two strategies assuring that the gotten solutions are as distinct as possible. If the number of solutions generated proves to be insufficient, we will call on a random approach.

In the first strategy, the solution *k* is generated while flipping the variable  $x_i$  from *Sref* as:  $i = (k-1) \text{ modulus } [Flip]$ . Whereas, in the second strategy, we consider *sref* as being a set of contiguous packets of bits. The solution *k* is generated while reversing the variables  $x_i$  of the packet *k* from *Sref*.

Let  $n=20$  (the number of variables) and  $Flip = 5$ . If the variables are subscripted from 0 to 19, then the first strategy consists in inverting the variables in the following positions: (0,5,10,15) , (1,6,11,16),(2,7,12,17),(3,8,13,18) et (4,9,14,19), while in the second strategy, the variables in the following positions are inverted: (0,1,2,3) , (4,5,6,7) , (8,9,10,11) , (12,13,14,15) et (16,17,18,19) .

**The Bee Search Process.** The bee search process is an iterative process. The number of iterations is called  $nbsearchit$  and is an empirical parameter. The process consists of two phases:

-A simple local research.

-A phase that consists in flipping the maximum of variables of the solution found in the first phase so that the fitness function of the new solution is superior or equal to it.

**The Choice of the Reference Solution  $S_{ref}$ .** Let  $S_{ref}(t)$  be the chosen reference solution at the  $t$  iteration, then the choice of  $S_{ref}(t+1)$  depends on the quantity  $\Delta f$  computed as:

$$\Delta f = f(S_{best}) - f(S_{ref}(t)); \quad S_{best} \text{ being the best solution at the iteration } t+1;$$

The algorithm is as follows:

```

begin
 if $\Delta f > 0$ then
 begin
 $S_{ref} \leftarrow$ the best solution in quality;
 If $NbChances < MaxChances$ then
 $NbChances \leftarrow MaxChances$;
 end
 else
 begin
 $NbChances \leftarrow NbChances - 1$;
 If $NbChances > 0$ then $S_{ref} :=$ the best
 solution in quality
 else
 begin
 $S_{ref} \leftarrow$ the best solution in diver-
 sity;
 $NbChances \leftarrow MaxChances$;
 end
 end
 end
 end
 end
end

```

*Remarks.* ( $S_{ref}$  is better in quality) is equivalent to ( $f(S_{ref}) = \text{Max } f(S)$ ) where  $S \in \text{Dance}$  and  $S \notin \text{Taboo}$ . ( $S_{ref}$  is better in diversity) is equivalent to ( $\text{diversity}(S_{ref}) = \text{Max diversity}(S)$ ) Where  $S \in \text{Dance}$ .

If two solutions  $S_1$  and  $S_2$  are equal in quality, that is, if they have the same value of the function objective then the one that has the largest degree of diversity will be considered.

In the same way, if two solutions  $S_1$  and  $S_2$  present the same degree of diversity, it is the one that improves the fitness function that will be chosen. It can happen, although very rarely, that all solutions of Dance exist in the Taboo list. For landing to this problem, the solution of reference will be generated at random. *maxchances* is an empirical parameter. It designs the maximum number of chances accorded to SearchArea.

## 6 Numerical Results

The algorithm BSO has been implemented in java on a Pentium personal computer and numerical tests have been performed on the benchmark instances available on the web site [http:// www.reserch.att.com/~mgcr/data/index.htm](http://www.reserch.att.com/~mgcr/data/index.htm). The weights of the clauses have been drawn from 1 to 1000 and assigned at random to clauses, the number of clauses being ranged from 800 to 950. The Johnson class namely 'jnh', has been used in many works for testing algorithms performance. It includes three subclasses characterized by the variable number, which is equal to 100 for all instances. The classes number is

800 for the subclass 1: Jnh201... Jnh 220.

850 for the subclass 2 : Jnh 01...Jnh 19.

900 for the subclass 3 : Jnh301...Jnh310.

Each subclass contains instances that are satisfied or not. On each instance, 10 executions have been undertaken. The solution quality as well as the running time, have been considered as performance criteria.

### 6.1 Parameters Setting

Preliminary tests have been carried out in order to fix the key parameters of the BSO algorithm for Max-W-SAT. Fig 1 shows an example of the results of the tests done on some Johnson problems for setting the parameter flip. Identical tests have been performed on all the other instances and for all the other parameters. Table 1 summaries the parameters values obtained after these extensive experiments.



**Fig. 1.** The results of tests for setting the parameter Flip

**Table 1.** Empirical parameters values of BSO for MAX-W-SAT

| NBees         | 10 |
|---------------|----|
| MaxSearchIter | 15 |
| MaxBsoIter    | 35 |
| Flip          | 5  |
| MaxChances    | 3  |

## 6.2 Performances Comparison

Comparison of BSO with evolutionary algorithms for SAT like GRASP[6], SSAT [3] and ACO[4] has been done. GRASP is a parallel greedy algorithm proposed for solving MAX-W-SAT. It has been considered in the comparison because of its high efficiency. SSAT is a scatter search algorithm recently developed. ACO is another recent meta-heuristic inspired by the behaviour of real ants. A schematic view of the results is shown through the curve of fig2, showing the performance of BSO. The same observation can be done on the performance of BSO.

## 7 Conclusion

In this paper, a new meta-heuristic called ‘swarm of bees’ or BSO has been proposed. It is inspired by the behaviour of the real bees, whose principle is to harvest the nectar of the easiest sources of access while always privileging the richest. The dance of the bees that is the only means of communication they have, is probably the secret of their efficiency since it is through it that a bee indicates to its fellows the place of a source and its quality. It leads the swarm very quickly to concentrate its efforts of research on the richest source.

In our application we were interested in one of the NP-Complete problems to solve, namely MAX-W-SAT. We have adapted the proposed meta-heuristic BSO to the features of the MAX-W-SAT problem and the results that we have gotten are very satisfactory.

## References

1. Eric Bonabeau, Marco Dorigo and Guy Theraulaz, *Swarm Intelligence: From Natural to Artificial Systems*, Oxford University Press, 1999.
2. Eric Bonabeau and Guy Theraulaz intelligence collective. Editions Hérmes, 1994.
3. Habiba Drias and Mohamed Khabzaoui. Scatter Search with Random Walk Strategy for Sat and Max-Sat Problems. *Springer Verlag Berlin Heidelberg 2001*.
4. Habiba Drias, Amine Taibi and Sofiane Zekour, Cooperative Ant Colonies for Solving the Maximum Weighted Satisfiability Problem, Springer-Verlag Berlin Heidelberg 2003.
5. D.S. Johnson, Approximate algorithms for combinatorial problems, JCSS, 256-278, 1974
6. L Ptisouli ,Parados and M.G. Resende.1997. approximate solution of weighted MAX-SAT problems Using GRASP AT & Research, Florham Park, NJ 07932 USA.

# Deriving Cortical Maps and Elastic Nets from Topology-Preserving Maps

Valery Tereshko

School of Computing, University of Paisley,  
Paisley PA1 2BE, Scotland  
valery.tereshko@paisley.ac.uk  
<http://cis.paisley.ac.uk/tere-ci0/>

**Abstract.** Soft topology-preserving map and its batch version are proven to be reduced to cortical map and elastic net, respectively. This verifies numerous results of numerical simulations described in the literature demonstrating similarities of neural patterns produced by lateral and elastic synaptic interactions.

## 1 Introduction

Competitive learning neural nets that utilize lateral interactions to perform a mapping from the stimulus space to the response space with preserving neighbourhood relations are called topology-preserving maps [1]. Well-known example of the above is Kohonen’s self-organizing map that became a standard unsupervised learning algorithm [2].

It is known that elastic synaptic interactions can forge a topology as well. The elastic net was first applied to solve the travelling salesman problem (TSP) [3]. Another application of elastic synaptic interactions is the preservation of topology in cortical mappings [4, 5, 6].

We already demonstrated the benefits of using both lateral and elastic interactions for controlling the receptive field patterns [5, 6]. In [7], we considered the model utilizing only lateral interactions, which, unlike elastic ones, are biologically plausible, and applied it to the problems previously solved only with elastic interactions. Our current aim consists in the proof of equivalence of both types of interactions. First, we derive the free energy function for an unsupervised net of stochastic neurons with lateral interactions. The temperature incorporated in this function serves as controlling parameter in the annealing schedule. Then, we consider the incremental and batch modes of learning resulting in corresponding versions of soft topology-preserving mapping. The mapping utilizes only nearest-neighbour lateral interactions that, in turn, are chosen to be weak. Considering the weight vector of a neuron as a “particle” moving in the space-time of imposed patterns, decompose this particle trajectory over these patterns. Using the decomposition for incremental and batch modes of soft topology-preserving map, we derive cortical map and elastic net respectively.

Numerical simulations justify the theoretical results: models utilizing lateral and elastic interactions produce similar patterns for given parameter values.

## 2 Topology-Preserving Maps

We consider a one-dimensional net of  $n$  stochastic neurons trained by  $N$  patterns. The energy of this net, for a given stimulus, is

$$E_i(\mu) = \frac{1}{2} \sum_{j=1}^n h_{ij} |\mathbf{x}_\mu - \mathbf{w}_j|^2, \quad (1)$$

where  $\mathbf{x}_\mu$  is a given sample pattern,  $\mathbf{w}_j$  are the weight vectors, and  $h(i, j)$  is the neighbourhood function.

Throughout, we consider nearest-neighbour lateral interactions

$$h_{ij} = \begin{cases} 1, & i = j; \\ \gamma, & |i - j| = 1; \\ 0, & |i - j| \geq 2, \end{cases} \quad (2)$$

where  $0 < \gamma \leq 1$ .

Instead of the “hard” assignment of Kohonen original algorithm with an unique winner, we assume a “soft” assignment where every  $i$ -th neuron is assigned to a given  $\mu$ -th pattern with a probability  $p_i(\mu)$ ;  $\sum_i p_i(\mu) = 1$  [5, 6, 7, 8, 9].

The assignment probabilities minimizing free energy of the system (that is a composite of the averaged energy and thermal noise energy) are found to be

$$p_i(\mu) = \frac{e^{-\beta E_i}}{\sum_{k=1}^n e^{-\beta E_k}}, \quad (3)$$

which gives the minimal free energy [5, 6, 7, 8, 9]

$$F(\mu) = -\frac{1}{\beta} \ln \left( \sum_{i=1}^n e^{-\beta E_i} \right). \quad (4)$$

Incremental learning strategies are derived through a steepest descent minimization of function (4). The dynamics follows of the free energy gradient, which result in soft topology-preserving mapping [5, 6, 7]:

$$\Delta \mathbf{w}_j = -\eta \frac{\partial F}{\partial \mathbf{w}_j} = \eta \sum_{i=1}^n p_i(\mu) h_{ij} (\mathbf{x}^\mu - \mathbf{w}_j). \quad (5)$$

Soft mapping is based on soft competition which allows all neurons to adjust their weights with probabilities proportional to their topographic distortion. This makes the weights move more gradually to the presented patterns. The strength of the competition is adjusted by a temperature. The underlying mechanism,

deterministic annealing, is derived from statistical physics: it mimics an ordering process during a system's cooling. At high temperatures, the competition is weak and the original energy landscape is smoothed by noise, which helps to eliminate local minima at the beginning of the ordering phase. On reducing the temperature, the competition becomes stronger, the smoothing effect gradually disappears, and the free energy landscape resembles the original one.

At low temperatures ( $\beta \rightarrow \infty$ ), equation (5) reduces to Kohonen's map with only nearest-neighbour interactions:

$$\Delta \mathbf{w}_j = -\eta \frac{\partial F}{\partial \mathbf{w}_j} = \eta h_{jj^*} (\mathbf{x}^\mu - \mathbf{w}_j), \quad (6)$$

where  $j^*$  is the winning unit.

The batch learning mode, when the updating rule is averaged over the set of training patterns before changing the weights, gives the following free energy:

$$\langle F \rangle = -\frac{1}{\beta N} \sum_{\mu=1}^N \ln \left( \sum_{i=1}^n e^{-\beta E_i} \right). \quad (7)$$

Minimization of energy (7) results in the batch version of soft topology-preserving map:

$$\Delta \mathbf{w}_j = -\eta \frac{\partial \langle F \rangle}{\partial \mathbf{w}_j} = \frac{\eta}{N} \sum_{\mu=1}^N \sum_{i=1}^n p_i(\mu) h_{ij} (\mathbf{x}_\mu - \mathbf{w}_j), \quad (8)$$

where  $\eta$  is the learning rate.

At low temperatures ( $\beta \rightarrow \infty$ ), (8) reduces to the batch mode of the Kohonen map. Goodhill applied the latter model with the special lateral interaction function to modelling the formation of topography and ocular dominance in the visual cortex [10].

### 3 Cortical Maps

The idea of cortex as a dimension-reducing map from high-dimensional stimulus space to its two-dimensional surface has proved to be fruitful [4, 11]. The backward projection of each position on the cortex sheet to the position in stimulus space is a convenient way to consider cortex self-organization — the way in which it fills stimulus space defines the receptive field properties. Performing such a mapping induces two conflicting tendencies: (i) the cortical surface should pass through the representative points in stimulus space; (ii) the area of the sheet should be kept a minimum. This ensures the formation of smooth receptive fields and, hence, the minimal “wiring” interconnecting the cortical cells, which, in turn, ensures the closeness of the cortical cells representing similar stimuli. The stripes and patches seen within cortical areas have been argued to be adaptations that allow the efficient wiring by such structures [12].

In cortical mappings, the topological order usually develops by elastic synaptic interactions [4]. Let us derive a cortical map from a topology-preserving map. Taking the Taylor series expansion (in power of  $\gamma$ ) results in

$$F = -\frac{1}{\beta} \ln \sum_{i=1}^n \exp\left(-\frac{\beta}{2} |\mathbf{x}_\mu - \mathbf{w}_i|^2\right) + \frac{\gamma}{2} \sum_{i=2}^{n-1} p_i(\mu) (|\mathbf{x}_\mu - \mathbf{w}_{i-1}|^2 + |\mathbf{x}_\mu - \mathbf{w}_{i+1}|^2). \quad (9)$$

Consider the weight vector as a ‘‘particle’’ moving in space-time  $\mathbf{x}$  and decompose this particle trajectory:

$$\mathbf{w}_j = \langle \mathbf{x}(j) \rangle = p_j(\nu) \mathbf{x}_\nu, \quad (10)$$

where  $\mathbf{x}(j)$  and  $\langle \mathbf{x}(j) \rangle$  are the position and the expected position of the particle at time  $j$  respectively.

Applying decomposition (10) to free energy (4) and taking the low temperature limit yield

$$F = -\frac{1}{\beta} \ln \sum_{i=1}^n \exp\left(-\frac{\beta}{2} |\mathbf{x}_\mu - \mathbf{w}_i|^2\right) + \frac{\gamma}{2} \sum_{i=1}^{n-1} |\mathbf{w}_{i+1} - \mathbf{w}_i|^2. \quad (11)$$

Minimization of free energy function (11) results in the elastic net algorithm:

$$\Delta \mathbf{w}_j = -\eta \frac{\partial F}{\partial \mathbf{w}_j} = \eta \left( \tilde{p}_j(\mu) (\mathbf{x}^\mu - \mathbf{w}_j) + \gamma (\mathbf{w}_{j+1} - 2\mathbf{w}_j + \mathbf{w}_{j-1}) \right), \quad (12)$$

where

$$\tilde{p}_j(\mu) = \frac{\exp(-\frac{\beta}{2} |\mathbf{x}^\mu - \mathbf{w}_j|^2)}{\sum_{k=1}^n \exp(-\frac{\beta}{2} |\mathbf{x}^\mu - \mathbf{w}_k|^2)} \quad (13)$$

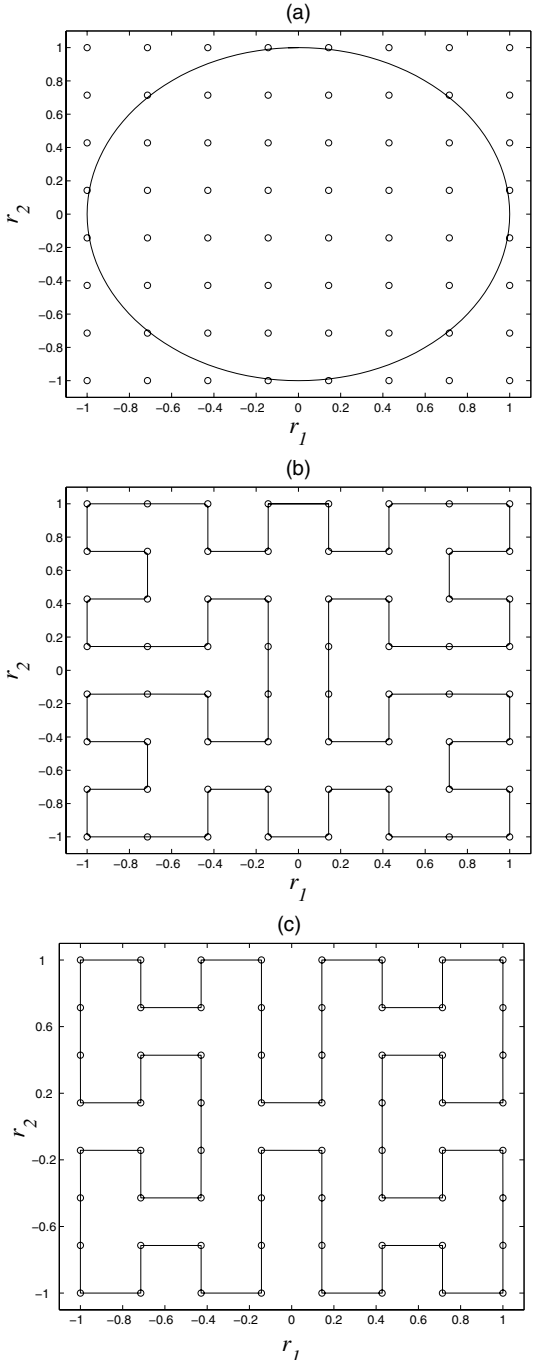
is a reduction of  $p_j(\mu)$  to the case of no lateral interactions.

## 4 Elastic Nets

The elastic net is based on elastic, diffusion-type, interactions [3]. This algorithm works like an elastic rubber ring: it gradually drags points on the ring towards the ‘‘cities’’ and an elastic force keeps neighbouring points close to one another.

Earlier, Simic showed the relationship between the Hopfield network and the elastic net: it derived the latter from Hopfield’s objective function for the TSP [13, 14]. Let us show how to derive the elastic net from the batch version of soft topology-preserving map.





**Fig. 1.** Weight vector distribution of the elastic ring: (a) initial, (b) and (c) after applying learning rules (8) and (17) respectively (see details in the text)

Taking the Taylor series expansion (in power of  $\gamma$ ) results in

$$\begin{aligned} \langle F \rangle = & -\frac{1}{\beta N} \sum_{\mu=1}^N \ln \sum_{i=1}^n \exp\left(-\frac{\beta}{2} |\mathbf{x}_\mu - \mathbf{w}_i|^2\right) \\ & + \frac{\gamma}{2N} \sum_{i=2}^{n-1} \sum_{\mu=1}^N p_i(\mu) (|\mathbf{x}_\mu - \mathbf{w}_{i-1}|^2 + |\mathbf{x}_\mu - \mathbf{w}_{i+1}|^2). \end{aligned} \quad (14)$$

Consider the weight vector as a “particle” moving in space-time  $\mathbf{x}$  and decompose this particle trajectory:

$$\mathbf{w}_j = \langle \mathbf{x}(j) \rangle = \sum_{\nu=1}^N p_j(\nu) \mathbf{x}_\nu, \quad (15)$$

where  $\mathbf{x}(j)$  and  $\langle \mathbf{x}(j) \rangle$  are the position and the expected position of the particle at time  $j$  respectively.

Applying decomposition (15) to free energy (7) and taking the low temperature limit yield

$$\langle F \rangle = -\frac{1}{\beta N} \sum_{\mu=1}^N \ln \sum_{i=1}^n \exp\left(-\frac{\beta}{2} |\mathbf{x}_\mu - \mathbf{w}_i|^2\right) + \frac{\gamma}{2N} \sum_{i=1}^{n-1} |\mathbf{w}_{i+1} - \mathbf{w}_i|^2. \quad (16)$$

Minimization of free energy function (16) results in the elastic net algorithm:

$$\Delta \mathbf{w}_j = -\eta \frac{\partial F}{\partial \mathbf{w}_j} = \frac{\eta}{N} \left( \sum_{\mu=1}^N \tilde{p}_j(\mu) (\mathbf{x}^\mu - \mathbf{w}_j) + \gamma (\mathbf{w}_{j+1} - 2\mathbf{w}_j + \mathbf{w}_{j-1}) \right). \quad (17)$$

Defining  $\beta \equiv \frac{1}{\sigma^2}$  with the Gaussian distribution width  $\sigma$ , energy (16) takes the exact form of the Durbin-Willshaw elastic net energy [3]. Shrinking the distribution width is, thus, equivalent to reducing the system temperature.

Let us demonstrate how different algorithms work for the TSP. The simulations are performed for 64 “cities” that are sites on a  $8 \times 8$  regular square. The elastic ring has 128 points. The training is cyclic with a fixed sequence, i.e. a particular order of pattern presentation is fixed. The inverse temperature  $\beta$  increases from 2 to 200 in steps of 0.01. The learning rate is linearly decreasing functions of time, i.e.  $\hat{\eta} = \frac{1}{N} \eta = \hat{\eta}_0 (1 - t/T)$  with  $\hat{\eta}_0 = 1$  and  $t = 0, \dots, T$ . Let us take  $\gamma = 0.06$ . Initially, the weight vectors are distributed equidistantly on the unit radius circle (Fig. 1(a)). Application of learning rules (8) to the task results in formation of an optimal tour (Fig. 1(b)). Typically, elastic strength is allowed to decrease with time passing [1]. Therefore, for the case of elastic net (learning rule (17)), we consider  $\gamma = \gamma_0 (1 - t/T)$  with  $\gamma_0 = 0.06$ , which provides a finer pattern than one for fixed  $\gamma$  (Fig. 1(c)). Thus, both nearest-neighbour lateral interactions and elastic synaptic interactions produce similar solutions (optimal tours in our case) for given interaction strength.

## 5 Conclusions

Remarkably, cortical map and elastic net can be derived from incremental and batch versions of soft topology-preserving map respectively. The nearest-neighbour lateral interactions are transformed to the elastic ones. In turn, the Gaussian variance of cortical map and elastic net is equivalent to the temperature of soft topology-preserving map. The latter elucidates indirect incorporation of soft competition and deterministic annealing into cortical maps and elastic nets.

One algorithm reduces to another in the limit of low temperature. At beginning of learning process, when the temperature is high, the state trajectory is very sensitive to any changes in the system and can take any possible direction. This is why the optimal tours in Fig. 1(b) and (c) are different. However, the algorithms tend to identity in working, and the tours formed are qualitatively the same.

## References

1. Hertz, J., Krogh, A., Palmer, R.G.: Introduction to the Theory of Neural Computation. Addison-Wesley, Reading (1991).
2. Kohonen, T.: Self-Organized Formation of Topologically Correct Feature Maps. *Biological Cybernetics* **43** (1982) 59-69.
3. Durbin, R., Willshaw, D.: An Analogue Approach to the Travelling Salesman Problem Using an Elastic Net Method. *Nature* **326** (1987) 689-691.
4. Durbin, R., Mitchison, G.: A Dimension Reduction Framework for Understanding Cortical Maps. *Nature* **343** (1990) 644-647.
5. Tereshko, V.: Topology-Preserving Elastic Nets. In: Mira, J., Prieto, A. (eds.): *Connectionist Models of Neurons, Learning Processes and Artificial Intelligence*. Lecture Notes in Computer Science, Vol. 2084. Springer-Verlag, Berlin Heidelberg New York (2001) 551-557.
6. Tereshko, V., Allinson, N.M.: Combining Lateral and Elastic Interactions: Topology-Preserving Elastic Nets. *Neural Processing Letters* **15** (2002) 213-223.
7. Tereshko, V.: Modelling Topology Preservation in a Cortex by Controlled Deformation of Its Energy Landscape. *Neurocomputing* **44-46C** (2002) p. 667-672.
8. Graepel, T., Burger, M., Obermayer, K.: Phase Transitions in Stochastic Self-Organizing Maps, *Phys. Rev. E* **56** (1997) 3876-3890.
9. Heskes, T.: Energy Functions for Self-Organizing Maps. In: Oja, E., Kaski, S. (eds.): *Kohonen Maps*. Elsevier, Amsterdam (1999) 303-315.
10. Goodhill, G.J.: Correlations, Competition, and Optimality: Modelling the Development of Topography and Ocular Dominance. *Cognitive Science Research Paper* 226. University of Sussex, Brighton (1992).
11. Swindale, N.V.: The Development of Topography in the Visual Cortex: a Review of Models. *Network: Comput. Neur. Syst.* **7** (1996) 161-247.
12. Mitchison, G.: Neuronal Branching Patterns and the Economy of Cortical Wiring. *Proc. Roy. Soc. B* **245** (1991) 151-158.
13. Simic, P.D.: Statistical Mechanics as the Underlying Theory of "Elastic" and "Neural" Optimisations. *Network: Comput. Neur. Syst.* **1** (1990) 89-103.
14. Berg, J. van den: *Neural Relaxation Dynamics: Mathematics and Physics of Recurrent Neural Networks with Applications in the Field of Combinatorial Optimization*. PhD Thesis. Erasmus University, Rotterdam (1996).

# Evolution of Cooperating ANNs Through Functional Phenotypic Affinity

F. Bellas, J.A. Becerra, and R.J. Duro

Grupo de Sistemas Autónomos, Universidade da Coruña, Spain  
{fran, ronin, richard}@udc.es

**Abstract.** This work deals with the problem of automatically obtaining ANNs that cooperate in modelling of complex functions. We propose an algorithm where the combination of networks takes place at the phenotypic operational level. Thus, we evolve a population of networks that are automatically classified into different species depending on the performance of their phenotype, and individuals of each species cooperate forming a group to obtain a complex output. The components that make up the groups are basic ANNs (primitives) and could be reused in other search processes as seeds or could be combined to generate new solutions. The magnitude that reflects the difference between ANNs is their affinity vector, which must be automatically created and modified. The main objective of this approach is to model complex functions such as environment models in robotics or multidimensional signals.

## 1 Introduction

The researchers dealing with modelling real environments tend to dedicate great efforts to the automatic modularization of the solutions, searching for models as simple and reusable as possible. In the field of evolutionary learning, an element of paramount importance is how to make efficient use of the learned structures to obtain other structures that can be applied in more complex modelling processes without performing a new learning process.

A very powerful technique for the instantiation of the models, when the learning process consists in modelling from input-output pairs, is the application of artificial neural networks (ANNs). In the case of evolutionary learning, these models adapt to the task through evolution and the modularity of solutions can be obtained creating complex ANN sets as a composition of simpler ones. In this sense, a very common approach in the literature [1], [2], [3] is to apply an evolutionary algorithm over two different populations, one made up of simple neuronal units and the other of patterns that indicate how those neural units must be combined. These approaches have been successful, for example, in classification problems.

A different approach is proposed here, where the basic structures participating in the process are complete networks evolving in the same population and they are combined at the phenotypic level into groups that collaborate in the solution of the problem. The work of Xin Yao and Paul Darwen [4] must be mentioned here. They propose an evolutionary learning approach to design modular systems automatically. To do this, they use the concept of speciation introduced by Goldberg [5] through a fitness sharing

technique [6]. The main novelty of this work lies in the fact that within each species they use coevolutionary techniques where the concept of just one individual solution does not exist. The authors apply this technique to the prisoner's dilemma and the algorithm works successfully.

This kind of modular approaches imply the existence of collaboration between individuals of the population to achieve a successful result. This is called in the literature symbiotic evolution and is a background problem in multiobjective optimization [7]. In this field, the development of fitness sharing techniques, where the fitness of a given individual is scaled through some similitude measure with respect to the other individuals, is common [8]. These approaches start from the knowledge of the Pareto front to be obtained, this is, they start from the knowledge of the desired objectives and search for individuals specialized in these objectives and that collaborate towards their achievement.

The problem we try to solve in this work is quite different because we look for an automatic decomposition of the model into simple and basic networks (primitives) that combined provide the desired solution. No knowledge about how to do it is assumed (we don't know the best decomposition into primitives). Each one of these simple networks is functionally different and, consequently, they belong to different species that evolve together.

A recent algorithm that is based on coevolution and species formation is COVNET [9], where authors use a subnetworks population (nodules) that is divided into species that evolve independently, and another network population that combine these nodules. The main difference with our approach lies at the species formation level. The number of species in COVNET is prefixed, whereas in our algorithm it arises automatically from the evolution itself.

In the next section we will introduce the main concepts used in this algorithm such as groups and affinity. The remaining sections provide an overview of the operation.

## 2 Groups and Affinity

The main objective of this work is very simple: to develop an algorithm capable of modelling a complex function using ANNs. Because of this complexity, we assume it will be simpler to find the solution using more than one ANN. So our approach is based on providing a search algorithm with the capability of obtaining the solution by aggregation of simple ANNs.

We do not want to impose knowledge about the way to carry out this combination because, in most cases, it is not possible to know how to extract simple parts (that could be modeled separately) from a complex function. This is, we know what we want to obtain from our ANNs (phenotype) but we don't know how to achieve this (genotype). This way, our efforts in the design of the algorithm are concentrated in obtaining a procedure to efficiently combine the outputs provided by different ANNs to reach a desired result.

Thus, the first important feature of our algorithm is that the solution is provided by a combination of ANNs (*a group*) that, in the simplest case, will be formed by just one network but it could contain several. All the ANNs in a group combine their outputs to provide a complete solution to the problem.

To conform the groups, we have developed a selection criteria based on a magnitude called *affinity* that classifies the ANNs depending on their phenotype. The main idea is that initially all the networks have a random affinity and through a self organization process these affinities become different automatically. Then, the groups that provide the solution to the problem are made up by combining ANNs with high affinity to each other. The details of this process will be explained in section 4.2.

### 3 Evolution

From the previous basic concepts of the algorithm two main requirements can be extracted: first that our approach is based on the use of a set of ANNs that can be combined and second that to improve the networks we must apply a search algorithm. Taking into account these two points, we propose the use of an evolutionary algorithm as search technique because it is based on a population (a set) of solutions and it is an established technique for adjusting the parameters of ANNs. Furthermore, evolutionary algorithms are very adequate for this approach because of their capabilities to create specialized individuals (*species*) in the population. This way, once some species appear in the population, we can make up the groups that conform the solutions using individuals from these species.

The differentiation between species must arise from the phenotypic capability of each ANN. It is the collaboration towards a common goal by a group what determines the different species through the association of individuals that are specialized in the different parts of the solution. This association is established through some type of affinity criterion based on the functional performance of the phenotype of the ANNs, so that individuals with higher affinity towards each other tend to collaborate in the tasks to be carried out.

We have developed a process of life interaction in order to evaluate the individuals in such a way that the survival of an individual is determined by the fitness obtained by the whole group in which the individual participates and not only on its particular fitness. The most important part of this group formation strategy is establishing the affinity among individuals, which must be a part of the evolutionary process and progressively adapt to the needs and virtues of each individual autonomously.

### 4 Details of the Algorithm

From the point of view of an evolutionary algorithm, the objective of group creation is to obtain individuals made up of subindividuals through the combination of their particular actuations, thus permitting a much more versatile way of reusing acquired knowledge.

As mentioned in the previous section, the proposed strategy is a part of any standard evolutionary algorithm where the individuals represent ANNs and where the solution is not a simple individual but an aggregation of some (a group) at a phenotypic level. Consequently, individuals will be evaluated (fitness calculation) as a part of a group and not individually. Evolution, on the other hand, takes place over the individuals as usual and the groups are formed in the fitness calculation stage of the algorithm.

The operation is simple and can be summarized in the following 6 steps:

1. Initial creation of a random population of ANNs, as usual in general evolutionary algorithms.
2. Initial creation of a random affinity value (represented by a vector in general) associated to each individual.
3. Fitness calculation of all the ANNs in population. To do this, we apply 5 basic steps:
  - a. Each individual in the population creates its own group by choosing other individuals from a window of the population using its and their affinity values.
  - b. Each group calculates a fitness value, resulting from the common application of its individuals to the modelling problem.
  - c. Each individual creates a new group and the fitness in these new groups is calculated. The best groups remain.
  - d. The fitness value obtained in step c) permits adjusting the affinity of the individuals depending on the increase or decrease of fitness.
  - e. Steps c) and d) are repeated a given number of iterations. At the end, we assign each individual the fitness of its own group.
4. Selection over the population of ANNs (like in an ordinary evolutionary algorithm) including the affinity value of each individual as a part of the selection criteria favoring reproduction of individuals with similar functional characteristics (same species).
5. Crossover and mutation as usual but including the affinity vector that is transmitted from parents to offspring.
6. Steps 3, 4 and 5 are repeated a given number of generations.

As we can see, this is the operation schema of a simple evolutionary algorithm where the fitness calculation is more complex and where the reproduction is directed towards the formation of species by mixing operationally similar individuals.

Each individual can form its own group selecting the other individuals from a candidate's window instead of the whole population, trying to reduce the number of possible combinations. In addition to its own group, every individual can participate in other groups without limitation of number.

#### 4.1 Fitness Calculation

Once a group is formed, its fitness is calculated in general as the fitness provided by a set of outputs that are the result of some combination of the individual outputs of its members. In the simplest case the combination is carried out through the addition of these outputs that must be clearly marked to discriminate which ANN affects which output. The group must provide values for all the outputs of the problem, otherwise it is penalized.

After calculating the fitness of all the groups, we carry out a self organization stage (steps from a) to e) in the algorithm), which allows each individual to create more affine groups. On each step of this stage, the individuals form new groups and their fitness is calculated. If the fitness of the new group is better than that of the old one, the group is maintained; otherwise the older group is preserved. After a set number of recombination steps, each individual has formed its most affine group using the technique explained in the next section.

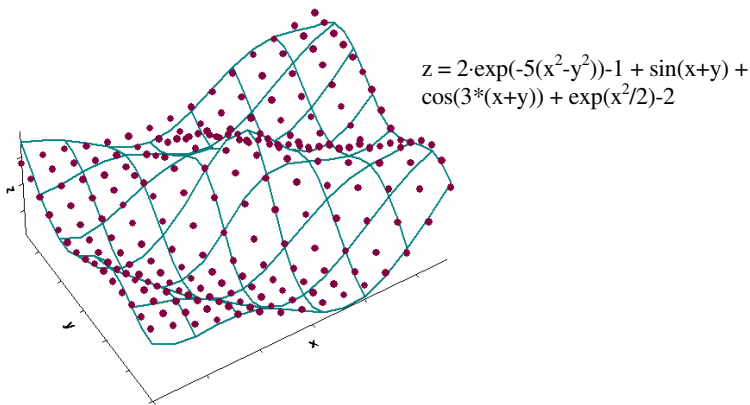
## 4.2 Affinity

Affinity is used in our algorithm as a label to classify the ANNs depending on the function they perform. It is represented by a numerical vector with fixed upper and lower limits, in which the dimension determines the complexity of the groups that may be formed.

Each element in the affinity vector is represented by a real number. When an individual forms a group it searches for other individuals in the population with *complementary* affinity vectors. For example, if each element of the vector is represented by a real number between  $-V$  and  $V$ , we can establish that the desired global affinity value is the zero vector. This way, a component value of  $-V_1$  is the complementary of a value of  $V_1$  and a value of  $-V_2$  is the complementary of a value of  $V_2$ . In most cases, an individual will not find another individual with a complementary affinity and it will be necessary to use two or three colleagues to reach the desired affinity.

As we can see, the number of individuals that conform a group depends on the dimension of the affinity vector. The higher the dimensionality the larger (on average) the number of individuals in the group necessary to obtain complementary affinities. For example, if we have an individual with a 3-dimensional affinity vector  $(V_1, V_2, -V_3)$  it could select one only individual with affinity  $(-V_1, -V_2, V_3)$  to form the group. But it is simpler to find two individuals of vectors  $(-V_4, -V_5, V_6)$  and  $(-V_7, -V_8, V_9)$  that verify  $-V_1 = -V_4 + (-V_7)$ ,  $-V_2 = -V_5 + (-V_8)$  and  $V_3 = V_6 + V_9$ . As we can see, higher dimensionalities imply a more complicated process to find individuals with complementary affinities.

Initially, the affinity vectors are random as no knowledge of the characteristics of the individuals and/or their shortcomings is available. As mentioned before, every time a group is evaluated (fitness calculation), if the group improves the fitness of the previous group in which the individual participated, the distance of the affinity vector of the individual (in Euclidean terms) with respect to the affinity vectors of the rest of the individuals in the group is increased. In addition, from this instant, this will be the new group for the individual. On the other hand, if the fitness of the group is worse than that



**Fig. 1.** Modelling result (points) of a 3D function (surface) made up by the addition of 4 basic functions



of the previous group, it is the distance with respect to the affinity of the members of the previous group that is increased. This method favours the formation of species, that is, of clusters of individuals with similar affinity vectors.

After each self organization stage, all the individuals have automatically adjusted their affinity vector depending on the function they carry out in the group. The individuals that are included in more than one group adjust their affinity vector in each one, maximizing the available information. As time progresses, each individual will have an affinity vector representing its preferences that has been automatically obtained.

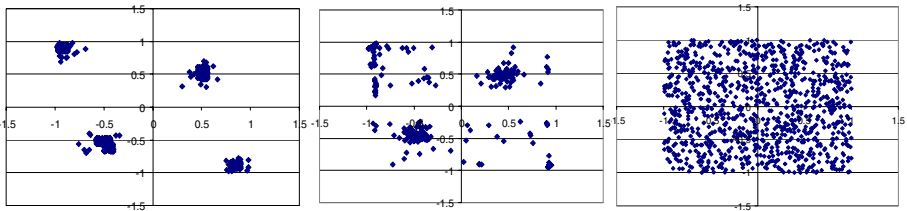
In the evolutionary process, we tend to select for reproduction individuals with similar affinity vectors, which are transmitted in the crossover from the parents to the offspring. This way, we are promoting the formation of species in the population. The feature that distinguishes these species (the affinity vector) has not been imposed by the designer, but it is adjusted in “lifetime” through the evolution.

## 5 Operation Example

Just to display the results the mechanism can produce and the way it works, we consider here a simple problem consisting of an objective function that was constructed by adding 4 different functions:  $2 \cdot \exp(-5(x^2 - y^2)) - 1$ ,  $\sin(x+y)$ ,  $\cos(3 \cdot (x+y))$  and  $\exp(x^2/2) - 2$  limited to a range from  $[-1.5, 1.5]$  in  $x$  and  $y$ . In this experiment, the affinity vector considered is two dimensional and the values of each dimension can vary between  $[-1, 1]$ . Thus, it seems that an optimal solution for this problem would be the generation of four species, each one specialized in modelling each one of the known functions or any other combination of functions that lead to the perfect modelling of the global function.

The networks that will be evolved in this case are standard multilayer perceptrons which have two inputs (the  $x$  and  $y$  coordinates) and one output (the  $z$  coordinate). The objective is obviously to obtain the network or combination of networks that best approximate the objective function.

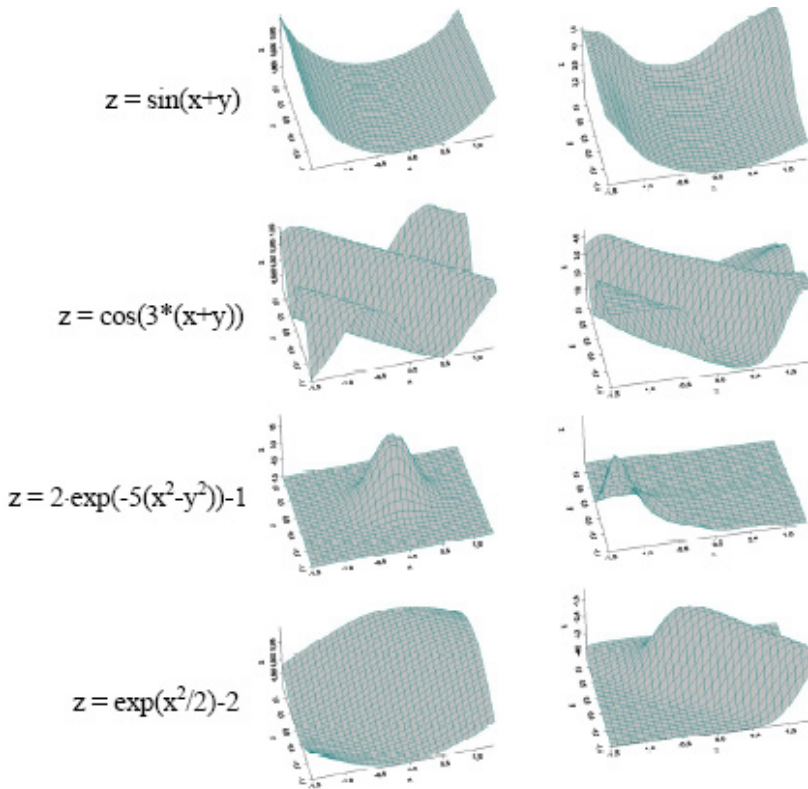
The results of modelling obtained with the best group are presented in Fig. 1, where we have represented the prediction (points) and a sampling of the objective function (surface). The results are very satisfactory taking into account the complexity of the expected signal. We are going to take a closer look at what have happened with the affinity values for the individuals and what types of species arose.



**Fig. 2.** Evolution of the affinity vectors for the whole population. Left graph represents the initial distribution, middle graph the distribution after 2000 generations and right graph the final distribution

Fig. 2 displays the evolution of the affinity vectors for the whole population (1000 individuals in this case). In the left graph we have represented the initial random affinities distributed from -1 to 1 in both axes. In the middle graph we show the affinity vectors in generation 2000. As we can see, the values are still not clustered, and some individuals do not belong clearly to a species, although four affinity areas are starting to be delimited. Finally, in the right graph we show the affinity vectors in generation 3600 (final) where 4 clear species have been formed with affinity vectors  $(-1, 1)$ ,  $(1, -1)$ ,  $(0.5, 0.5)$  and  $(-0.5, -0.5)$ .

The final group taken as solution of the modelling problem (that provides prediction shown in Fig. 1) is made up of 4 individuals, one from each species. In Fig. 3 we present the functions provided by each one of these individuals in the final group, as compared to the original four functions that were combined in order to produce the test problem. We can see that the decomposition into primitives is different from the original one which is normal because there are lots of possible mathematical combinations to reach the desired functions. What is important here is that these primitives have been obtained automatically and could be reused in other search processes.



**Fig. 3.** The left graphs show the four theoretical functions that make up the target in the operation example. The right graphs show the functions provided by each individual in the best group (phenotype corresponding to each specie)

## 6 Conclusions

In this work we have presented an affinity based strategy for obtaining groups of cooperating artificial neural networks as a solution for evolutionary learning problems. This strategy may be applied in general to any type of evolutionary algorithm where the objective is to obtain a group of solution points which, through their combination and cooperation in a phenotypic level produce a modular or decomposable solution to the problem so that parts of these solutions may be later reused in other problems. The main idea behind the phenotypic affinity based process is that during the evaluation phase of the evolutionary algorithm the individuals can try out how fit they are through collaboration with other individuals by forming groups. The results of these trials result in modifications of each individual's affinity vector. This process leads to the formation of species of individuals specialized in different subtasks of the global one, thus allowing for the modularity we desire. The arising of these species implies certain modifications in the general evolutionary algorithm, especially during the reproduction stage so that functionally similar individuals are chosen for reproduction. The results applying this strategy have been very fruitful. Here we have presented a quite complex case with the arising of four species into the population.

## Acknowledgements

This work was funded by the MCYT of Spain through project VEM2003-20088-C04-01 and Xunta de Galicia through project PGDIT03TIC16601PR.

## References

1. Smalz, R., Conrad, M.: Combining evolution with credit apportionment: A new learning algorithm for neural nets, *Neural Networks*, vol 7, num 2, 1994.
2. Opitz, D.W., Shavlik, J.W.: Actively searching for an effective neural network ensemble, *Connection Sci*, vol 8, num 3, 1996.
3. Moriarty, D.E., Miikkulainen, R.: Forming neural networks through efficient and adaptive coevolution, *Evolutionary Computation*, vol 4, num 5, 1998.
4. Darwen, P., Yao, X.: Speciation as automatic categorical modularization, *IEEE Transactions on Evolutionary Computation*, 1(2), 1997.
5. Goldberg, D., Richardson, J.: Genetic algorithms with sharing for multimodal function optimization, *Proceedings of 2nd International Conf. on Genetic Algorithms*, 1987.
6. Deb, K. and Goldberg, D. E.: An investigation of niche and species-formation in genetic function optimization, *Proc. 3<sup>rd</sup> Int. Conf. Genetic Algorithms*, pp 42-50, 1989.
7. Coello, C., Van Veldhuizen, D., Lamont, G.: *Evolutionary algorithms for Solving Multi-Objective Problems*, Kluwer Academic Publishers, 2002.
8. DellaCioppa, A., DeStefano, C., Marcelli, A.: On the Role of Population Size and Niche Radius in Fitness Sharing, *IEEE Trans on Evolutionary Computation*, vol 8, num 6, 2004.
9. García-Pedrajas, N., Hervás-Martínez, C., Muñoz-Pérez, J.: COVNET: A Cooperative Coevolutionary Model for Evolving Artificial Neural Networks, *IEEE Transactions on Neural Networks*, vol 14, num 3, 2003.

# Robust Growing Hierarchical Self Organizing Map\*

Sebastián Moreno<sup>1</sup>, Héctor Allende<sup>1</sup>, Cristian Rogel<sup>1</sup>, and Rodrigo Salas<sup>1,2</sup>

<sup>1</sup> Universidad Técnica Federico Santa María,  
Dept. de Informática; Casilla 110-V; Valparaíso-Chile,  
{smoreno, hallende, crogel}@inf.utfsm.cl

<sup>2</sup> Universidad de Valparaíso; Departamento de Computación  
rodrigo.salas@uv.cl

**Abstract.** The Growing Hierarchical Self Organizing Map (GHSOM) was introduced as a dynamical neural network model that adapts its architecture during its unsupervised training process to represent the hierarchical relation of the data.

However, the dynamical algorithm of the GHSOM is sensitive to the presence of noise and outliers, and the model will no longer preserve the topology of the data space as we will show in this paper. The outliers introduce an influence to the GHSOM model during the training process by locating prototypes far from the majority of data and generating maps for few samples data. Therefore, the network will not effectively represent the topological structure of the data under study.

In this paper, we propose a variant to the GHSOM algorithm that is robust under the presence of outliers in the data by being resistant to these deviations. We call this algorithm Robust GHSOM (RGHSOM). We will illustrate our technique on synthetic and real data sets.

**Keywords:** Growing Hierarchical Self Organizing Maps, Robust Learning Algorithm, Data Mining, Artificial Neural Networks.

## 1 Introduction

In the last decade, Neural Networks have proven to be a valuable tool in data mining with various engineering applications in pattern recognition, image analysis, process monitoring and fault diagnosis.

Most of the real data are highly dimensional and to find the best model capable to learn and generalize the data, the neural designer has the difficulty to decide in advance the architecture and the number of parameters of the model. To overcome the architectural design problem several algorithms with an adaptive structure during the training process have been proposed, as for example

---

\* This work was supported in part by Research Grant Fondecyt 1040365, DGIP-UTFSM, BMBF-CHL 03-Z13 from German Ministry of Education, DIPUV-22/2004 and CID-04/2003.

for the SOM based models we refer the GHSOM [12] and the Dynamic SOM [1] to name a few.

The GHSOM was introduced by [12] as a dynamical model that adapts its architecture during its unsupervised training process to represent the hierarchical relation of the data. The GHSOM has a hierarchical structure of multiple layers, where each layer consists of several independent growing SOM's.

Real data are not free of outliers, i.e., data items lying very far from the main body of the data. Unfortunately unsupervised neural networks are not robust to the presence of outliers as we have shown in early works (see [3] and [4]). Furthermore, it is also possible that the outliers are not erroneous but some data items really are strikingly different from the rest. For this reason it is not advisable to discard the outliers and instead special attention must be paid.

In this paper we propose a variant to the learning algorithm of the GHSOM that diminishes the influence of outliers, but still considers them during the training, we call this model RGHSOM. The remainder of this paper is organized as follows. The next section we introduce the M-estimators as robust method for parameter estimation. In the third section, we introduce the learning algorithm of the RGHSOM. Simulation results on synthetic and real data sets are provided in the fourth section. Conclusions and further work are given in the last section.

## 2 Robust M-Estimators for the Learning Process

The learning process of Artificial Neural Networks models can be seen as a parameter estimation process, and their inference relies on the data. When there exist observations substantially different from the bulk of data, known as outliers, they can influence badly the model structure bringing degradation in the estimates. In this work we seek for a robust estimator for the parameters of the RGHSOM by applying M-estimators introduced by Huber [8].

Let the data set  $\chi = \{\mathbf{x}_i\}_{i=1..n}$  consists of an independent and identically distributed (i.i.d.) sample of size  $n$  coming from the probability distribution  $F(\mathbf{x})$ . An M-estimator  $\hat{\underline{\theta}}_n^M$  is defined by

$$\hat{\underline{\theta}}_n^M = \arg \min_{\underline{\theta} \in \Theta} \{RL_n(\underline{\theta})\} \quad \text{where} \quad RL_n(\underline{\theta}) = \frac{1}{n} \sum_{i=1}^n \rho(\mathbf{x}_i, \underline{\theta})$$

where  $\Theta$  is the parametric space,  $RL_n(\underline{\theta})$  is a robust functional cost and  $\rho$  is the robust function that introduces a bound to the influence of outliers data during the training process. By assuming that  $\rho$  is differentiable, whose derivative is given by  $\psi(\mathbf{x}, \underline{\theta}) = \frac{\partial \rho(\mathbf{x}_i, \underline{\theta})}{\partial \underline{\theta}}$ , the M-estimator can be defined implicitly by the solution of  $\frac{1}{n} \sum_{i=1}^n \psi(\mathbf{x}_i, \hat{\underline{\theta}}_n^M) = \mathbf{0}$ .

### 3 The RGHSOM Model

The RGHSOM has a tree architecture where each node consists in a robust self organizing map (RSOM) [4] that models a particular region of the input space. The model is generated dynamically during the training process. In this section we propose to robustify the learning process of the GHSOM such that the model becomes insensitive to the presence of outlying observations.

**Locating the Neuron to Be Hierarchical Expanded.** When the layer 0 is created with the neuron  $\underline{m}^{[0]}$ , it must be located at the center of the data that it models before a new grid will be generated. The estimation of the location of the neuron is given by an M-estimator  $\underline{m}^{[0]}$  obtained as the minimization of the robust functional cost given by,

$$RL(\underline{m}^{[0]}, s^{[0]}) = \sum_{\underline{x}_i \in \mathcal{I}} \rho \left( \frac{\underline{x}_i - \underline{m}^{[0]}}{s^{[0]}} \right) \quad \mathcal{I} \neq \emptyset$$

where  $s^{[0]}$  its robust estimation of the standard deviation of the data of the input space  $\mathcal{I}$

For example for  $\rho(x) = x^2$  the M-estimator will be the mean  $\underline{m}^{[0]} = \overline{X}_n$  of the data, or for  $\rho(x) = |x|$  will be the median  $\underline{m}^{[0]} = \text{median}\{x_i, i = 1..n\}$ . In [7] several M-estimators can be found, in particular we are interested in estimators that diminish the influence of outliers. To estimate  $s^{[0]}$  we prefer the MEDA function given by  $s^{[0]} = 1.483 \text{ median} \{ |\underline{x}_i - \underline{m}^{[0]}| - \text{median}[\underline{x}_i - \underline{m}^{[0]}] \}$

After the location of the first neuron, a new grid with initial size of  $2 \times 2$  units is created beneath it. The map could be created randomly, where the prototypes are generated from a normal or an uniform distribution with mean  $\underline{m}^{[0]}$  and variance  $(s^{[0]})^2$ . It can be also created deterministically around the center.

**Robust Self Organizing Map (RSOM).** In [4] we introduced a robust learning algorithm for the Kohonen' SOM [9]. When the data  $\underline{x}$  is presented to the model, it searches for the best matching unit (*bm*u) that is nearest to the input and is obtained by  $\|\underline{x} - \underline{m}_{i,c}^{[u]}\| = \min_j \{ \|\underline{x} - \underline{m}_{i,j}^{[u]}\| \}$ . The winning unit and its neighbors adapt to represent the input by modifying their reference vectors towards the current input. The amount that the units learn will be governed by a neighborhood kernel  $h_c([u, l, j], t)$ , which is a decreasing function of the distance between the unit  $j$  and the *bm*u  $c$  on the map lattice  $l$  at layer  $u$  at time  $t$ . The kernel is usually given by a Gaussian function,  $h_c([u, l, j], t) = \alpha(t) \exp \left( \frac{-\|\underline{r}_{l,j}^{[u]} - \underline{r}_{l,c}^{[u]}\|^2}{2\sigma(t)^2} \right)$ , where  $\underline{r}_{l,j}^{[u]}$  and  $\underline{r}_{l,c}^{[u]}$  denote the coordinates of the neurons  $j$  and  $c$  in the lattice  $l$  at layer  $u$ ,  $\alpha(t)$  is the learning rate parameter and  $\sigma(t)$  is the neighborhood range. In practice the neighborhood kernel is chosen to be wide in the beginning of the learning process to guarantee global ordering of the map, and both its width and height decrease slowly during learning.

The influence of the outlier was diminished by introducing a robust  $\psi(\cdot)$  function,  $\psi : \chi \times \mathcal{M}_{l,j}^{[u]} \rightarrow \mathbb{R}^n$ ,  $\underline{m}_{l,j}^{[u]} \in \mathcal{M}_{l,j}^{[u]} \subseteq \mathbb{R}^n$ , in the update rule as follows:

$$\underline{m}_{l,j}^{[u]}(t+1) = \underline{m}_{l,j}^{[u]}(t) + h_c([u, l, j], t) \psi \left( \frac{\underline{x} - \underline{m}_{l,j}^{[u]}(t)}{s_{l,j}^{[u]}(t)} \right) \quad j = 1..M_l^{[u]} \quad (1)$$

where  $s_{l,j}^{[u]}(t)$  is a robust estimation of the standard deviation of the set of data  $\mathcal{C}_{l,j}^{[u]}$  modelled by the neuron  $j$  of the map  $l$  at layer  $u$ . To estimate  $s_{l,j}^{[u]}(t)$  we use a variant of the MEDA function given by:

$$s_{l,j}^{[u]}(t) = 1.483 \text{ median} \left\{ |h_c^*([u, l, j], t)[\underline{x} - \underline{m}_{l,j}^{[u]}] - \text{median}(h_c^*([u, l, j], t)[\underline{x} - \underline{m}_{l,j}^{[u]}])| \right\}$$

The learning parameter function  $\alpha(t)$  is a monotonically decreasing function with respect to time, for example this function could be linear  $\alpha(t) = \alpha_0 + (\alpha_f - \alpha_0)t/t_\alpha$  or exponential  $\alpha(t) = \alpha_0(\alpha_f/\alpha_0)^{t/t_\alpha}$ , where  $\alpha_0$  is the initial learning rate ( $< 1.0$ ),  $\alpha_f$  is the final rate ( $\approx 0.01$ ) and  $t_\alpha$  is the maximum number of iteration steps to arrive  $\alpha_f$ .

For example the Huber function could be used, which is given by  $\psi_H(\underline{x}, \underline{m}_j) = \text{sgn}(\underline{r}_j) \min\{|\underline{r}_j|, \delta\}$ ,  $\delta \in \mathbb{R}^+$ ,  $\underline{r}_j = (\underline{x} - \underline{m}_j)/s_j$ .

**Quality Measure and Global Network Control.** The quality of the adaptation of the RGHSOM to the data is measured in terms of the deviation between the units and it is given by the robust quantization error of each unit  $j$  of the map  $l$  at layer  $u$ :

$$rqe_{u,l,j} = \sum_{\underline{x}_i \in \mathcal{C}_{i,j}^{[u]}} \rho \left( \frac{\underline{x}_i - \underline{m}_{l,j}^{[u]}}{s_{l,j}^{[u]}} \right), \quad \mathcal{C}_{l,j}^{[u]} \neq \phi \quad (2)$$

where  $\mathcal{C}_{l,j}^{[u]}$  is the set of input vectors belonging to the Voronoi polygon of the unit  $j$  of the map  $l$  at layer  $u$ . When the training process is starting, we calculate the robust quantization error  $rqe_0$  of the first neuron representing the layer 0 over the whole data set  $\mathcal{I}$ . All units must represent their respective Voronoi polygons of data at a robust quantization error smaller than a fraction  $\tau$  of  $rqe_0$ :

$$rqe_{u,l,j} < \tau \cdot rqe_0 \quad (3)$$

The units that not satisfy this criterion require a more detailed data representation.

**Hierarchical Growth.** Every unit has to be checked for fulfillment of the global stopping criterion given by equation (3). If a unit  $j$  of the map  $l$  at layer  $u$  does not fulfill the criterion, then a new layer must be created for the vectors mapped onto this unit. The new grid with initial size of  $2 \times 2$  units is created beneath the unit  $\underline{m}_{l,j}^{[u]}$ . The map could be created randomly, where the prototypes are generated from a normal or a uniform distribution with mean  $\underline{m}_{l,j}^{[u]}$  and variance

$(s_{l,j}^{[u]})^2$ , where the last is calculated as the variance of the set  $\mathcal{C}_{l,j}^{[u]}$ . It can be also created deterministically around the center.

**Growth Process.** During the growing process, the map is trained with the RSOM algorithm, and all its units are analyzed with equation (2). Then, the unit with the highest *rqe*, called error unit  $e$  and its most dissimilar neighbor  $d$  are determined by

$$e = \arg \max_j \left\{ \sum_{\underline{x}_i \in \mathcal{C}_{l,j}^{[u]}} \rho \left( \frac{x_i - m_{l,j}^{[u]}}{s_{l,j}^{[u]}} \right) \right\}, \text{ and } d = \arg \max_j \left( \left\| \underline{m}_e^{[u]} - \underline{m}_{l,j}^{[u]} \right\| \right),$$

respectively, where  $\mathcal{C}_{l,j}^{[u]} \neq \phi$ ,  $\underline{m}_{l,j}^{[u]} \in \mathcal{N}_e, \mathcal{N}_e$  is the set of neighboring units of the error unit  $e$ . A row or column of units is inserted between  $e$  and  $d$  and their model vector are initialized as the means of their respective neighbors. After insertions, the map is trained again.

The growth process thus continues only until the map's robust mean quantization error given by  $RMQE_t^{[u]} = \frac{1}{n_u} \cdot \sum_{j \in \mathcal{U}} rqe_{u,l,j}$ ,  $n_u = |\mathcal{U}|$  reaches a certain fraction  $\gamma$  of the *rqe* of the corresponding unit  $\kappa$  in the upper layer  $RMQE_t^{[u]} < \gamma \cdot rqe_\kappa$ , where  $\mathcal{U}$  is the subset of the maps units onto which data is mapped. All the units of the map must fulfill the stopping criterion.

**Evaluation of the Adaptation Quality.** To evaluate the quality of adaptation to the data a common measure to compare the algorithms is needed and the following metric based on the mean square quantization error is proposed:

$$MSQE = \frac{1}{n} \sum_{\substack{\underline{m}_{l,j}^{[u]} \in \mathcal{N} \\ \underline{x}_i, i=1..n}} \sum_{\substack{h_c([u, l, j], t) \\ \alpha(t)}} \left\| \underline{x}_i - \underline{m}_{l,j}^{[u]} \right\|^2 \quad (4)$$

where  $\mathcal{N}$  is the set of all prototypes that are leaves in the hierarchical structure, i.e., it does not have a grid below.

## 4 Simulation Results

In this section we compare the computational capabilities of the RGHSOM with the GHSOM algorithm to show the lack of robustness of the last method and the improvement obtained by our algorithm. To validate the RGHSOM algorithm we follow the Prechelt test [11], for this reason we apply first the algorithm to computer generated data and then to two real data sets obtained from benchmarks and we compare the results with the GHSOM algorithm.

### 4.1 Experiment #1: Computer Generated Data

Five clusters of two-dimensional Gaussian distribution  $\underline{X}_k \sim \mathcal{N}(\mu_k, \Sigma_k)$ ,  $k = 1, \dots, 5$  were constructed, where  $\mu_k$  and  $\Sigma_k$  are the mean vector and the covariance matrix respectively of the cluster  $k$ . A total of 500 training samples and 500 test samples were drawn. The information about the parameters used to generate the clusters are



| Cluster | $n$ | $\mu$          | $\Sigma$       |
|---------|-----|----------------|----------------|
| 1       | 75  | $[0.5; 0.5]^T$ | $0.03^2 * I_2$ |
| 2       | 75  | $[0.6; 0.4]^T$ | $0.03^2 * I_2$ |
| 3       | 100 | $[0.1; 0.1]^T$ | $0.10^2 * I_2$ |
| 4       | 100 | $[0.8; 0.8]^T$ | $0.15^2 * I_2$ |
| 5       | 150 | $[0.6; 0.9]^T$ | $0.01^2 * I_2$ |

The observational process is obtained by adding additive outliers:  $Z_k = X_k + V_k U_k$ , where  $V_k$  is zero-one process with  $P(V_k \neq 0) = \gamma$ ,  $0 < \gamma \ll 1$  and  $U_k$  has distribution  $\mathcal{N}(\underline{0}, \Sigma_{U_k})$  with  $|\Sigma_{U_k}| \gg |\Sigma_k|, k = 1, \dots, 5$ . The generating process was affected with  $\gamma = 0\%, 5\%$  and  $10\%$  of outliers with  $\Sigma_{U_k} = 0.5^2 * I_2$ .

The simulation results are shown in table 1, where the column *Algorithm* is the type of learning algorithm, *%out* is the percentage of additive outliers, *Neurons*, *Grids* and *Layers* gives the number of prototypes, maps and grids generated respectively, column *% neurons* is the percentage of neurons that model at least one data, the *MSQE* is the mean square quantization error (4) of the test set consisting in 500 samples. Furthermore we introduce the Akaike information criterion (AIC),  $AIC = n \ln(MSQE) + 2p$ , and the Bayesian information criterion (BIC),  $BIC = n \ln(MSQE) + p + p \ln(n)$  both penalize the the number of parameters (see [2]),  $p$  is the number of parameters (neurons) in the model.

As can be noted, the RGHSOM outperforms the GHSOM algorithm in all the cases. With increasing percentage of outliers the RGHSOM obtained a lower *MSQE* and its error increases slowly with the degree of outliers. The number of prototypes, grids and layers needed to model the topology of the data are lower than the GHSOM in one order of magnitude. The percentage of neurons usage given by the column *% neurons* of the RGHSOM is closer to 60% while the GHSOM model is around 30%, i.e., the last model has neurons and grids modelling no data. Furthermore, if we analyze the information criterion we can appreciate that the RGHSOM with negative values is better than the GHSOM with positive values when the *MSQE* is penalized with the number of parameters. Both models were trained with 500 data but the GHSOM needed over 600 neurons to model the topology structure, and this is unacceptable.

**Table 1.** Summary results showing the comparative performance of the GHSOM v/s RGHSOM algorithms using the synthetic dataset

| Algorithm | %out. | Neurons | Grids | Layers | % neurons    | MSQE         | AIC      | BIC      |
|-----------|-------|---------|-------|--------|--------------|--------------|----------|----------|
| GHSOM     | 0     | 977     | 145   | 7      | 29.89        | 0.306        | 2452.816 | 6456.979 |
| RGHSOM    | 0     | 126     | 16    | 4      | <b>53.17</b> | <b>0.128</b> | -772.747 | -115.706 |
| GHSOM     | 5     | 647     | 98    | 7      | 29.83        | 0.192        | 469.9    | 3844     |
| RGHSOM    | 5     | 61      | 6     | 4      | <b>57.38</b> | <b>0.113</b> | -967.6   | -649.5   |
| GHSOM     | 10    | 638     | 97    | 7      | 28.21        | 1.509        | 430.892  | 3757.812 |
| RGHSOM    | 10    | 63      | 7     | 4      | <b>66.67</b> | <b>0.745</b> | -937.644 | -609.123 |

## 4.2 Experiment #2: Real Datasets

In the second experiment we test the algorithm with two real datasets known as the *Wisconsin Breast Cancer Database* obtained from the UCI Machine Learning repository [5] and the *The Phoneme Database* of the French and Spanish phoneme recognition problem from the Elena project [6].

The *Cancer* data was collected by Dr. Wolberg, N. Street and O. Mangasarian at the University of Wisconsin [10]. The samples consist of visually assessed nuclear features of fine needle aspirates (FNAs) taken from patients' breasts. It consist in 569 instances, with 30 real-valued input features and a diagnosis ( $M$  = malignant,  $B$  = benign) for each patient. We partitioned the dataset into the Training and Test with size of 300 and 269 data respectively.

The *Phoneme* data of the French and Spanish phoneme recognition problem from the Elena project [6] consists in distinguishing between nasal (AN, IN, ON) and oral (A, I, O, E) vowels. The data has 5404 examples, each with 5 continuous valued inputs (the normalized amplitudes of the five first harmonics), and 1 binary output. We partitioned the dataset into the Training and Test with size 3000 and 1404 data respectively.

**Table 2.** Summary results showing the comparative performance of the GHSOM v/s RGHSOM algorithms using the Wisconsin Breast Cancer Database and the Phoneme Database

| Dataset        | Algorithm | Neurons | Grids | Layers | % neurons    | MSQE         | AIC      | BIC      |
|----------------|-----------|---------|-------|--------|--------------|--------------|----------|----------|
| <i>Cancer</i>  | GHSOM     | 696     | 108   | 4      | 16.95        | <b>2.064</b> | 1587.04  | 4787.957 |
|                | RGHSOM    | 372     | 27    | 3      | <b>44.89</b> | 2.085        | 941.70   | 2650.93  |
| <i>Phoneme</i> | GHSOM     | 3814    | 543   | 7      | 20.66        | <b>0.381</b> | 5310.750 | 31188.30 |
|                | RGHSOM    | 822     | 83    | 5      | <b>51.58</b> | 0.547        | 197.254  | 5774.430 |

The simulation results are shown in table 2 and has the same fields than table 1. We have similar conclusions than the Synthetic case. The difference is that the GHSOM algorithm has a lower *MSQE* than the RGHSOM, but as mentioned before, if we analyze the information criterion we can appreciate that the RGHSOM outperforms the GHSOM when the *MSQE* is penalized with the number of parameters. It is important to mention that the Cancer and the Phonema data have 300 and 3000 training samples respectively and the GHSOM needed 696 and 3814 neurons to learn the topology, this model is clearly overfitted and it is undesirable to have more parameters than data. The percentage of neurons usage given by the column *% neurons* of the RGHSOM is closer to 45% while the GHSOM model is around 20%, i.e., the last model has a lot of neurons and grids modelling no data.

## 5 Concluding Remarks

In this paper we have introduced the Robust Growing Hierarchical Self Organizing Maps (RGHSOM) for the analysis of data that were affected by outliers.

The performance of our algorithm shows better results in the simulation study in both the synthetic and real data sets. In the synthetic data we study several degree of contamination while in the real case, we investigated two benchmarks data known as *Cancer* and *Phoneme* databases. The comparative study with the GHSOM shows that our algorithm outperforms the alternative models while conserving its complexity rather low avoiding overfitting. The outliers introduce an influence to the GHSOM model during the training process by locating prototypes far from the majority of data and generating maps for few samples data. Therefore, the network will not effectively represent the topological structure of the data under study.

Further studies are needed in order to analyze the convergence, the ordering properties together with the possible applications of the RGHSOM.

## 6 Acknowledgements

The authors wish to thank Prof. Dr. Claudio Moraga and Carolina Saavedra for their valuable comments and for all the stimulating debates.

## References

1. D. Alahakoon, S. Halgamuge, and B. Srinivasan, *Dynamic self-organizing maps with controlled growth for knowledge discovery*, IEEE Trans. on Neural Networks **11** (2000), no. 3, 601–614.
2. H. Allende, C. Moraga, and R. Salas, *Artificial neural networks in time series forecasting: A comparative analysis*, Kybernetika **38** (2002), no. 6, 685–707.
3. H. Allende, S. Moreno, C. Rogel, and R. Salas, *Robust neural gas for the analysis of data with outliers*, IEEE-CS Press. SCCC 2004 (2004), 149–155.
4. ———, *Robust self organizing maps*, CIARP. LNCS **3287** (2004), 179–186.
5. C.L. Blake and C.J. Merz, *UCI repository of machine learning databases*, 1998.
6. University College London Neural Network Group, *The elena project*, <http://www.dice.ucl.ac.be/neural-nets/Research/Projects/ELENA/elena.htm>.
7. F.R. Hampel, E.M. Ronchetti, P.J. Rousseeuw, and W.A. Stahel, *Robust statistics*, Wiley Series in Probability and Mathematical Statistics, 1986.
8. Peter J. Huber, *Robust statistics*, Wiley Series in probability and mathematical statistics, 1981.
9. T. Kohonen, *Self-Organizing Maps*, Springer Series in Information Sciences, vol. 30, Springer Verlag, Berlin, Heidelberg, 2001, Third Extended Edition 2001.
10. O. Mangasarian, W. Street, and W. Wolberg, *Breast cancer diagnosis and prognosis via linear programming*, Operations Research **43** (1995), no. 4, 570–577.
11. L. Prechelt, *Proben1 - a set of benchmarks and benchmarking rules for neural training algorithms*, Technical Report 21/94, Fakultat fur Informatik, Universitat Karlsruhe, Germany, 1994.
12. A. Rauber, D. Merkl, and M. Dittenbach, *The growing hierarchical self-organizing map: Exploratory analysis of high-dimensional data*, IEEE Trans. on Neural Networks **13** (2002), no. 6, 1331–1341.

# Web Usage Mining Using Support Vector Machine

Sung-Hae Jun

Department of Statistics, Cheongju University, 360-764 Chungbuk, Korea  
shjun@cju.ac.kr

**Abstract.** The web contains rich and dynamic collections of hyperlink information, web page access, and usage information providing rich sources for data mining. From this, we need a system to recommend a visitor good information. This recommendation system can be constructed by web usage mining process. The web usage mining mines web log records to discover user access patterns of web pages. Also it is the application of data mining techniques to large web log data in order to extract usage patterns from user's click streams. In general, the size of web log records is so large that we have difficulty to analyze web log data. To make matter worse, the web log records are very sparse. So it is very hard to estimate the dependency between the web pages. In this paper, we solved this difficulty of web usage mining using support vector machine. In the experiments, we verified our proposed method by given data from UCI machine learning repository and KDD cup 2000.

## 1 Introduction

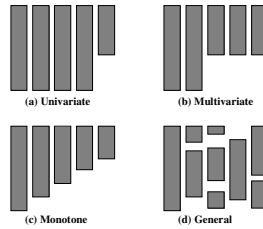
The web log records contain much collection of hyperlink information and the usage transactions of web page access. The size of web log data is very large, but web log data are very sparse. So we have a serious difficulty for web mining. It is very difficult to estimate the dependency of all web pages from spare web data. We have found that the statistical learning theory by Vapnik was a good approach for analyzing the sparse data because of its  $\mathcal{E}$ -insensitive loss function[16]. Using the missing value imputation by statistical learning theory, the spare data set is changed to complete data set. This provides a useful strategy for sparse data pre-processing like web log data. In this paper, we use SVR(support vector regression) among statistical learning models. The SVR is the regression version of SVM(support vector machine) by Vapnik[7]. The SVM can be applied to the case of regression, maintaining all the main features that characterize the maximal margin. Using SVR we make an efficient missing value imputation model to preprocess sparse web log. In our work, the SVR method offered a good result in spare web log data. Through experiments by UCI machine learning repository and KDD Cup 2000 data we verified these results[17],[18].

## 2 Support Vector Machine for Web Usage Mining

### 2.1 A Problem of Web Usage Mining

The sparseness of web log file is a problem of web usage mining. This is occurred by several reasons. Frequently it happens when the not visited web pages are much larger

than the visited web pages in web log file. The click stream data of cleaned web log are very sparse. So we have a difficulty of web log analysis as web usage mining with web information recommendation, next web page prediction, and web page duration time forecasting. The click stream data with sparseness is hard to analyze by general methods as regression, MLP, and others[6]. In this case the SVR is very useful tool for analyzing sparse data. Because of the sparseness of web log file, the structure of click stream data is incomplete. These incomplete data have extremely many missing cells. The missing data patterns from given data matrix are following figure[10].



**Fig. 1.** Typical examples of missing data

We show 4 missing data patterns in figure 1. In the (a) and (b) of figure 1, the missing data are eliminated by simple imputation method as mean and conditional mean methods. The (c) shows monotone missing pattern. The method for this pattern is statistical missing data analysis models as multiple imputations. But the (d) is a very difficult pattern for imputing missing values. We know the missing data pattern of web log file is (d). So our study will use an effective method which is SVR for analyzing missing data.

**2.2 Support Vector Machine for Regression**

Our given training data consist of  $N$  pairs  $(x_1, y_1), \dots, (x_N, y_N)$ , where  $x$  denotes the input patterns and  $y$  is target variable. In SVR with  $\mathcal{E}$ -insensitive loss function, our goal is to find a function  $f(x)$  that has at most  $\mathcal{E}$ -deviation from the actually obtained targets  $y_i$  for all the training data, and at the same time, is as flat as possible[15]. In other words, we do not care about errors as long as they are less than  $\mathcal{E}$ , but will not accept any deviation larger than this. The  $\mathcal{E}$ -insensitive loss function is defined as,

$$M(y, f(x, \alpha)) = L(|y - f(x, \alpha)|_{\mathcal{E}}) \tag{1}$$

We denote,

$$|y - f(x, \alpha)|_{\mathcal{E}} = \begin{cases} 0, & \text{if } |y - f(x, \alpha)| \leq \mathcal{E}, \\ |y - f(x, \alpha)| - \mathcal{E}, & \text{o.w.} \end{cases} \tag{2}$$

$\alpha$  is a positive constant. The loss is equal to 0 if the discrepancy between the predicted and the observed values is less than  $\mathcal{E}$ . The case of linear function  $f$  is described.

$$f(x) = \langle w, x \rangle + b \tag{3}$$

where  $\langle \cdot, \cdot \rangle$  denotes the dot product. For SVR, the Euclidean norm  $\|w\|^2$  is minimized. Formally this problem can be written as a convex optimization problem by requiring [16]. Analogously to the loss function in [16], we introduce slack variables  $\xi_i, \xi_i^*$  to copy with otherwise infeasible constraints of the optimization problem.

$$\text{minimize } \frac{1}{2} \|w\|^2 + C \sum_{i=1}^N (\xi_i + \xi_i^*) \tag{4}$$

$$\text{subject to } \begin{cases} y_i - \langle w, x_i \rangle - b \leq \mathcal{E} + \xi_i \\ \langle w, x_i \rangle + b - y_i \leq \mathcal{E} + \xi_i^* \\ \xi_i, \xi_i^* \geq 0 \end{cases} \tag{5}$$

The constant  $C > 0$  determines the trade off between the flatness of  $f$  and the amount up to which deviations larger than  $\mathcal{E}$  are tolerated. Using a standard dualization method utilizing Lagrange multipliers, the parameters are determined from equation (4) and (5) [5].

### 2.3 Web Usage Mining Using Support Vector Regression

In this paper, for the sparseness elimination from click stream data, the missing value imputation approach was used. Our imputation method is SVR. This has a good performance for sparse data analysis because of its  $\mathcal{E}$ -insensitive loss function [16]. And this satisfies conditions for consistency of risk minimization principle [16]. Figure 2 shows sparse click stream data from web log file. This must be complete for web usage mining.

|       | Page1 | Page2 | Page3 | ... | PageN |
|-------|-------|-------|-------|-----|-------|
| User1 |       | 8     | 17    | ... |       |
| User2 | 6     |       |       | ... |       |
| User3 |       | 5     |       | ... |       |
| User4 | 11    |       |       | ... | 3     |
| User5 |       |       | 21    | ... |       |
| ⋮     | ⋮     | ⋮     | ⋮     | ⋮   |       |
| ⋮     | ⋮     | ⋮     | ⋮     | ⋮   |       |
| UserM |       | 7     |       | ... |       |

Fig. 2. Incomplete click-stream data from web

Each cell of table in above figure contains a duration time of user accessing. The sparseness of cells is very serious. Therefore, general preprocessing methods like the missing value imputation methods as single and multiple imputations are not suitable to solve this problem. This fact will be verified next experiment in section 5.1. In our research, the SVR was used for sparseness elimination from web log file. Figure 3 shows complete table without sparseness using SVR method.

|       | Page1 | Page2 | Page3 | ... | PageN |
|-------|-------|-------|-------|-----|-------|
| User1 | 8     | 8     | 17    | ... | 3     |
| User2 | 6     | 9     | 13    | ... | 2     |
| User3 | 10    | 5     | 11    | ... | 1     |
| User4 | 11    | 6     | 10    | ... | 3     |
| User5 | 9     | 4     | 21    | ... | 4     |
| ⋮     | ⋮     | ⋮     | ⋮     | ⋮   | ⋮     |
| ⋮     | ⋮     | ⋮     | ⋮     | ⋮   | ⋮     |
| ⋮     | ⋮     | ⋮     | ⋮     | ⋮   | ⋮     |
| UserM | 6     | 7     | 12    | ... | 3     |

Fig. 3. Complete data without missing value

The duration time of  $i$  th page is estimated as following equation.

$$\hat{T}_{Pagei} = f(T_{Page1}, \dots, T_{Page(i-1)}, T_{Page(i+1)}, \dots, T_{PageN}) \tag{6}$$

In above equation,  $T_{PageK}$  was defined the duration time of page K by user accessing and  $\hat{T}_{Pagei}$  was defined the estimated duration time.  $\hat{T}_{Pagei}$  was computed by the SVR method of (N-1) pages,  $T_{Page1}, \dots, T_{Page(i-1)}, T_{Page(i+1)}, \dots, T_{PageN}$  out of  $i$  page. Therefore we predicted each page’s duration time as using estimating missing cells. This approach changes sparse table of figure 2 into complete table of figure 3.

### 3 Experimental Results

Since this paper restricted itself to the consideration of web usage mining by SVR, we tried three experiments related with web usage mining. First, the experiment of missing value imputation for preprocessing was performed using the abalone data set from UCI machine learning repository. Next, using the web log data of real web site(gazelle.com) from KDD cup 2000, our method of web usage mining was confirmed experimentally. In the final experiment, we verified our recommendation system for web usage mining.

#### 3.1 Experiment 1: Missing Value Imputations

In this section, we showed the experimental results of the performance between single and multiple imputations by abalone data set from UCI machine learning

repository[18]. The number of instances of given data was 4,177. The 7 attributes which are length(x1), diameter(x2), height(x3), whole weight(x4), shucked weight(x5), viscera weight(x6), and shell weight(x7) represent the abalone’s physical state. The data were originally complete. For our experiments, we made given data incomplete on purpose. The incomplete data consisted of sub data sets with missing ratios as 5%, 10%, 20%, 25%, 30%, 40%, 50%, and 60%. We compared the multiple imputations methods with single imputation methods. In current experiment, the MSE(mean squared error) of each table was computed as following[2].

$$MSE = \frac{1}{N} \sum_{i=1}^N (T_i - O_i)^2 \tag{7}$$

where,  $T_i$  is the target variable(known) and  $O_i$  is the predictive variable(unknown). The smaller the value of MSE is, the better the performance of method is. We verified this experimental result by following table.

**Table 1.** MSE of imputation methods for preprocessing

| Methods             |          | Missing ratios |        |        |        |        |        |        |
|---------------------|----------|----------------|--------|--------|--------|--------|--------|--------|
|                     |          | 5%             | 10%    | 20%    | 30%    | 40%    | 50%    | 60%    |
| Single Imputation`  | Tree     | 0.0068         | 0.0070 | 0.0078 | 0.0083 | 0.0089 | 0.0091 | 0.0108 |
|                     | A’s wave | 0.0540         | 0.0585 | 0.0505 | 0.0591 | 0.0603 | 0.0631 | 0.0673 |
|                     | D-based  | 0.1413         | 0.1459 | 0.1357 | 0.1461 | 0.1491 | 0.1501 | 0.1533 |
|                     | Huber    | 0.1309         | 0.0540 | 0.0588 | 0.0611 | 0.0673 | 0.0679 | 0.0736 |
| Multiple Imputation | P-score  | 0.0021         | 0.0031 | 0.0039 | 0.0045 | 0.0051 | 0.0053 | 0.0067 |
|                     | MCMC(1)  | 0.0013         | 0.0015 | 0.0022 | 0.0024 | 0.0036 | 0.0055 | 0.0057 |
|                     | MCMC(2)  | 0.0011         | 0.0014 | 0.0020 | 0.0022 | 0.0033 | 0.0051 | 0.0052 |
|                     | MCMC(3)  | 0.0010         | 0.0013 | 0.0018 | 0.0020 | 0.0029 | 0.0047 | 0.0048 |

In the above table, Tree, A’s wave, D-based, and Huber of single imputation methods were tree imputation, Andrew’s wave, distribution based imputation, and Huber imputation respectively[3],[8],[9],[13],[14]. P-score of multiple imputations methods was propensity score[10]. The MCMC(m)s of multiple imputations methods were MCMC imputation methods. In each MCMC(m), the missing data were filled in  $m$  times to generate  $m$  complete data sets. In the above result, the more missing ratio was increased, the larger the value of MSE was. Also we knew the MSEs of multiple imputations methods were smaller than the MSEs of single imputation methods. The result of above MCMC imputations showed that the values of MSEs were smaller according to increase the value of  $m$ . So we compared proposed SVR method with multiple imputations methods in following section.

### 3.2 Experiment 2: Preprocessing Web Log Data

In this section, we showed the results of experiment using KDD Cup 2000 data[17]. The data set had web log file of real internet shopping mall(gazelle.com). The capacity of given data was 1.2GB. We used he one-third of given data for the validation and the other two-thirds for training[11]. After data cleaning, we showed the basic information of given data in the following table.



**Table 2.** Summarized data set

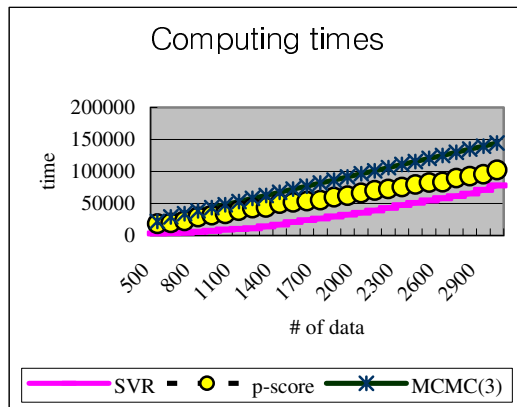
| Attributes    | Value range        |
|---------------|--------------------|
| Cookie-id     | 13,109 (users)     |
| assortment-id | 269 (web pages)    |
| duration-time | 0~1000 (second(s)) |

In table 2, the cookie-id was the index of user accessing to web site. The assortment-id represented each web page containing the descriptive contents of each item in the shopping mall and the duration-time of web page had the value between 0 and 1000 seconds. By previous experimental result, the propensity score and MCMC methods were compared with our SVR approach. Table 3 showed the result of this experiment.

**Table 3.** Result of evaluation

| Methods             |          | MSE (total) | MSE (upper 50%) |
|---------------------|----------|-------------|-----------------|
| Multiple Imputation | P. Score | 3.10        | 2.38            |
|                     | MCMC(3)  | 2.36        | 1.98            |
| SVR                 |          | 1.69        | 1.21            |

In this result, the MSE values of total and upper 50% of testing data were computed. The MSE of SVR was smaller than propensity score and MCMC. Therefore, we found the SVR had a good performance. Also next figure showed a computing time among propensity score, MCMC, and SVR methods. In figure 4, SVR had the smallest computing time in the compared methods. Because web usage mining was dynamically on-line process, SVR was a good method for recommendation system of web contents in web usage mining. So, in next experiment, we compared the recommendation system by SVR with other system.



**Fig. 4.** Computing times of compared methods

### 3.3 Experiment 3: Recommendation of Web Pages

We made an experiment on the performance of SVR by comparing with Pearson's correlation and collaborative filtering methods[1],[4],[12]. For this experiment, the 150 users, those who accessed more than 20 web pages, were used. On the assumption that 10 pages of above given data were already visited, the other 10 pages were used for estimating the preference of each web page. The preferences of web pages were computed by SVR. According to the order of preference value, the web pages standing a high 50% rank of given data were defined 'High' and the others were defined 'Low'. And we used following confidence measure for the evaluation of recommendation system[6].

$$\text{Confidence}(X \Rightarrow Y) = P(Y | X) \quad (8)$$

Above equation is the conditional probability of Y given that X has occurred. The X and Y were the web pages in our experiment.

**Table 4.** Accuracy of recommendation between SVR and Pearson's correlation

| Confidence   | SVR  | Pearson's Correlation | Collaborative Filtering |
|--------------|------|-----------------------|-------------------------|
| P(High High) | 0.41 | 0.35                  | 0.33                    |
| P(Low Low)   | 0.38 | 0.31                  | 0.29                    |
| P(High Low)  | 0.16 | 0.16                  | 0.19                    |
| P(Low High)  | 0.12 | 0.13                  | 0.18                    |

In above table, P(High|High) was showed the probability of visited High(web pages) after visited High(web pages). So we wanted that the values of P(High|High) and P(Low|Low) were large and the values of P(High|Low) and P(Low|High) were small. From table 4 we knew that the confidence of SVR was the best in the comparative methods.

## 4 Conclusion

The objective of this paper was to use SVR for an effective web usage mining. Though the SVR had been used in statistical machine learning models for the data without web log data, we tried to use SVR for web log data analysis. In our work, we verified the performance of SVR by the accuracy of preprocessing and the confidence of recommendation in web usage mining. Our future works will be to develop the hybrid SVR combined with the competitive co-evolving computing for intelligent web usage mining.

## References

1. Agrawal, R., Imielinski, T., Swami, A.: Mining association rules between sets of items in large databases, Proceeding of the ACM SIGMOD International Conference on Management of Data, (1993)

2. Casella, G., Berger, R. L.: *Statistical Inference*, Duxbury Press. (1990)
3. Conversano, C., Cappelli, C.: Missing data incremental imputation through tree based methods, 14<sup>th</sup> Conference on Computational Statistics, pp. 24-28, (2002)
4. Cooley, R., Mobasher, B., Srivastava, J.: *Web Mining: Information and Pattern Discovery on the World Wide Web*, Proceeding of the 9<sup>th</sup> IEEE International Conference on Tools with Artificial Intelligence, (1997)
5. Fletcher, R.: *Practical Methods of Optimization*, John Wiley & Sons, (1989)
6. Han, J., Kamber, M.: *Data Mining Concepts and Techniques*, Morgan Kaufmann, (2001)
7. Haykin, S.: *Neural Networks*, Prentice Hall, (1999)
8. Hoaglin, D. C., Mosteller, F., Tukey, J. W.: *Understanding robust and exploratory data analysis*, John Wiley & Sons, (2000)
9. Lavori, R. J. A., Dawson, R., Shera, D.: A Multiple Imputation Strategy for Clinical Trials with Truncation of Patent Data, *Statistics in Medicine*, vol. 14, 1913-1925, (1995)
10. Little, R. J. A., Rubin, D. B.: *Statistical Analysis with Missing Data*, Wiley Inter-Science, (2002)
11. Mitchell, T. M.: *Machine Learning*, McGraw-Hill, (1997)
12. Resnick, P., Lacovou, N., Suchak, M., Berfstrom, P., Riedl, J.: GroupLens: An Open Architecture for collaborative filtering of Netnews, *Proceedings of ACM 1994 Conference on Computer Supported Cooperative Work*, (1994)
13. Rubin, D. B.: *Multiple Imputation for Nonresponse in Surveys*, John Wiley & Sons, (1987)
14. Schafer, J. L.: *Analysis of Incomplete Multivariate Data*, Chapman and Hall, (1997)
15. Smola, A.J.: *Regression estimation with support vector learning machines*, Master's thesis, Technische University, (1996)
16. Vapnik, V. Z.: *Statistical Learning Theory*. John Wiley & Sons, Inc. (1998)
17. <http://www.ecn.purdue.edu/KDDCUP>
18. <http://www.ics.uci.edu/mlearn/MLRepository.html>

# Multi-kernel Growing Support Vector Regressor\*

D. Gutiérrez-González, E. Parrado-Hernández, and A. Navia-Vázquez

Dept. Signal Processing and Communications, Universidad Carlos III de Madrid,  
Avda Universidad 30, 28911-Leganés, Madrid, Spain  
{davidg, emipar, navia}@tsc.uc3m.es

**Abstract.** This paper presents a method to iteratively grow a compact Support Vector Regressor so that the balance between size of the machine and its performance can be user-controlled. The algorithm is able to combine Gaussian kernels with different spread parameter, skipping the ‘a priori’ parameter estimation by allowing a progressive incorporation of nodes with decreasing values of the spread parameter, until a cross-validation stopping criterion is met. Experimental results show the significant reduction achieved in the size of the machines trained with this new algorithm and their good generalization capabilities.

## 1 Introduction

Support Vector Regressors (SVRs) extend the excellent properties of the Support Vector Machines (SVMs) to solving function approximation problems [11]. One common way of designing a SVR is through the  $\epsilon$ -insensitive cost function, that mainly consists in assuming as error-free all the samples with error smaller than a threshold  $\epsilon$ . Moreover, the expression of the regressor is determined as a linear combination of kernel functions ‘centered’ in some patterns also called Support Vectors (SVs). SVRs are specially attractive because the regressor is the result of a convex optimization problem, that guarantees a globally optimal unique solution. Moreover, the machine size is automatically determined after solving the optimization problem, what skips the previous estimation of the optimal architecture, a drawback of traditional neural networks.

However, SVRs may be of reduced usability in certain applications since the resulting machine size can be extremely high. The regressor can be regarded as a one hidden layer neural network -every kernel computation representing a node in the hidden layer- with a node in the output layer that combines the outputs of the hidden one. Besides, the performance of the machine is usually severely determined by the parameters of the kernel function used in the SVR [9]. That involves two more restrictions on the SVR: (i) the ‘a priori’ estimation of optimal kernel parameters, and (ii) the difficulty to carry out a multiresolution analysis of data, i.e., to tackle different parts of a same problem with different granularity.

---

\* This work has been partially supported by Spanish Government CICYT grant TIC2002-03713.

With respect to the complexity of the machine, some previous works have tried to reduce the final machine size through either a postprocessing stage, like the Reduced Set Method of [8] or a prior reduction of the kernel matrix, like the sparse greedy matrix approximation of [10]. However, the former incurs in extra computational cost (training a conventional SVR plus the pruning stage) and the latter may lead to suboptimal results since the reduction of the kernel matrix is not oriented by the problem to be solved.

Recently, a method to iteratively Grow a Support Vector Classifier (GSVC) has been proposed [6]. This method enables to balance the trade-off between the size of the classifier and the classification error. The core of GSVC is WLS-SVC (Weighted Least Squares SVC) [4], an algorithm that constructs a semiparametric approximation to a SVC based on a previously selected architecture. Moreover, WLS-SVC allows to separate the selection of the kernel representatives -'centroids'- from the determination of the coefficients of the ensemble of kernel functions. This property supports the classification algorithm MK-GSVC (Multi-kernel GSVC) [6], that enables the combination of different kernel functions in the same SVC. This paper introduces the algorithm Multi-Kernel Growing Support Vector Regressors (MK-GSVR), that extends the ideas underlying MK-GSVC to functional approximation problems. When combined with a Gaussian kernels of different spread parameter  $\sigma$ , MK-GSVR approximates the functions as mixtures of heterogeneous Gaussian functions and avoids the previous estimation of the optimal  $\sigma$ . Although not explored in this paper, combinations of linear kernels and varying degree polynomials can also be considered.

The rest of the paper is organized as follows. Section 2 reviews the construction of compact SVRs from a given architecture, that forms the core of the MK-GSVR algorithm, described in Section 3. Some numerical results are presented in Section 4, and finally, Section 5 presents the concluding remarks.

## 2 Incremental Training to Control the Complexity of a SVR

As introduced before, MK-GSVR iteratively builds a parametric approximation SVR. Therefore, MK-GSVR includes two main procedures: the selection of hyperparameters of the kernels introduced in the ensemble at each iteration and the solution of the optimization problem that determines the weights of the ensemble so that the structural risk is globally minimized. This last function is carried out by WLS-SVR (Weighted Least Squares SVR), a version of WLS-SVC that constructs compact SVRs with  $\epsilon$ -insensitive cost function, whose formulation is presented in this section. The starting point of WLS-SVR is a method to determine the architecture of the SVR by the solution of a Iterative Re-Weighted Least Squares problems (IRLS-SVR) [7]. Given a projection onto a suitable feature space  $\phi(\cdot)$  associated to a kernel  $k(\cdot, \cdot)$ , and a training dataset  $\{(\mathbf{x}_i, y_i)\}_{i=1}^l$ , with  $\mathbf{x} \in \mathcal{X}$  and  $y \in \mathbb{R}$ , the analytical expression of the SVR in feature space is:

$$f(\mathbf{x}) = \phi^T(\mathbf{x})\mathbf{w} + b \quad (1)$$

If we drop the bias term,  $\mathbf{w}$  is the solution of the following equation [7]:

$$[I + \Phi D_{\mathbf{a}+\mathbf{a}^*} \Phi^T] \mathbf{w} = \Phi D_{\mathbf{a}+\mathbf{a}^*} [\mathbf{y} - \mathbf{E}\epsilon] \quad (2)$$

where  $\Phi = [\phi(\mathbf{x}_1), \dots, \phi(\mathbf{x}_l)]$ ,  $\mathbf{E} = D_{\mathbf{a}+\mathbf{a}^*}^{-1} D_{\mathbf{a}-\mathbf{a}^*} \mathbf{1}$  and  $D_{\mathbf{a}\pm\mathbf{a}^*}$  are diagonal matrices with  $(D_{\mathbf{a}\pm\mathbf{a}^*})_{ii} = a_i \pm a_i^*$ . Coefficients  $a_i^{(*)1}$  are computable as [4]:

$$a_i = \frac{\alpha_i}{y_i - f(\mathbf{x}_i) - \epsilon}, \quad a_i^* = \frac{\alpha_i^*}{f(\mathbf{x}_i) - y_i - \epsilon}. \quad (3)$$

At this point, we assume a parametric solution to equation (2),  $\mathbf{w} = \Psi\beta$ . The solution is supported by some ‘a priori’ chosen centroids  $\{\mathbf{c}\}_{i=1}^R$ , so  $\Psi = [\phi(\mathbf{c}_1), \dots, \phi(\mathbf{c}_R)]$ :

$$[I + \Phi D_{\mathbf{a}+\mathbf{a}^*} \Phi^T] \Psi\beta = \Phi D_{\mathbf{a}+\mathbf{a}^*} [\mathbf{y} - \mathbf{E}\epsilon] \quad (4)$$

multiplying both terms by  $\Psi^T$  we arrive at:

$$[\Psi^T \Psi + \Psi^T \Phi D_{\mathbf{a}+\mathbf{a}^*} \Phi^T \Psi] \beta = \Psi^T \Phi D_{\mathbf{a}+\mathbf{a}^*} [\mathbf{y} - \mathbf{E}\epsilon] \quad (5)$$

The application of the kernel trick [9] in equation (5) allows to solve the system in the input space by substituting the dot products with kernels. This way, substituting  $K = \Phi^T \Psi$ ,  $(K)_{ij} = k(\mathbf{x}_i, \mathbf{c}_j)$  and  $I_\Psi = \Psi^T \Psi$ ,  $(I_{Psi})_{ij} = k(\mathbf{c}_i, \mathbf{c}_j)$ , equation (5) turns to

$$[I_\Psi + K^T D_{\mathbf{a}+\mathbf{a}^*} K] \beta = K^T D_{\mathbf{a}+\mathbf{a}^*} [\mathbf{y} - \mathbf{E}\epsilon] \quad (6)$$

Equation (6) is the actual system to iteratively solve by means of WLS-SVR, in a manner analogous to the one described in [4] for the classification scenario. For this purpose, variables  $a_i^{(*)}$  are initialized to  $\{0, C\}$  with equal probability, and equation (6) is solved to obtain an initial set of weights  $\beta$ . Then, coefficients  $a_i^{(*)}$  are updated according to a procedure analogous to that described in [4]:

$$a_i = \begin{cases} 0 & \text{if } y_i - \mathbf{k}_i \beta - \epsilon < 0 \\ M & \text{if } \frac{C}{M} \geq y_i - \mathbf{k}_i \beta - \epsilon \geq 0 \\ C/(y_i - \mathbf{k}_i \beta - \epsilon) & \text{if } y_i - \mathbf{k}_i \beta - \epsilon \geq \frac{C}{M} \end{cases} \quad (7)$$

$$a_i^* = \begin{cases} 0 & \text{if } \mathbf{k}_i \beta - y_i - \epsilon < 0 \\ M & \text{if } \frac{C}{M} \geq \mathbf{k}_i \beta - y_i - \epsilon \geq 0 \\ C/(\mathbf{k}_i \beta - y_i - \epsilon) & \text{if } \mathbf{k}_i \beta - y_i - \epsilon \geq \frac{C}{M} \end{cases} \quad (8)$$

where  $\mathbf{k}_i$  represents the  $i$ -th row of  $K$  and  $M$  is a constant typically above  $C^2$ . This updating is equivalent to approaching the abrupt change in the step function ( $\alpha$  as a function of the slack variables) of SVM by a steep function with slope  $M$  [4]. This procedure of solving for  $\beta$  and updating the  $a_i^{(*)}$  is repeated until convergence.

<sup>1</sup> We refer to both  $a_i$  and  $a_i^*$  with the notation  $a_i^{(*)}$ .

### 3 Growing Support Vector Regressors

In this section we describe an algorithm for iteratively growing the architecture of a SVR, controlling its complexity, analogous to that one in [6] for the classification scenario. Then the formulation for a case with mixture of different kernels is presented (Gaussian kernels with varying width), and the GSVR algorithm is extended to the multi-kernel case.

#### 3.1 Growing Support Vector Regressor Algorithm: GSVR

For initialization of the GSVR algorithm, the first  $N$  centroids are picked up at random from the training set, since no information about the function is available yet. Then an initial kernel matrix  $K_0$  is computed and the initial function  $\beta(0)$  and coefficients  $\mathbf{a}_i^{(*)}(0)$  are obtained via WLS-SVR.

The prediction error obtained with this initial machine can be used to select new centroids to increase the representational capabilities of the machine, by incorporating them into the architecture and updating weights. In our case, we force the training procedure to concentrate on the worst predicted patterns (those outside the regression tube), thereby improving the representation in  $\mathcal{F}$  of this region of the input space. The method for selecting new centroids is analogous to the one described in [2]. Note that, every time the regressor is expanded with new nodes, the GSVR algorithm only updates the parameters for step “ $n$ ” using those at step “ $n - 1$ ” as a starting point, not needing re-training from scratch. Furthermore, all the kernels computed at step “ $n$ ” do not need to be re-computed at following steps, because their centroids are kept.

To avoid overfitting and to get good generalization characteristics, we stop the algorithm by means of a validation procedure that detects when performance stops increasing. Validation is applied on a reserved subset (about 20%) of the training data that is not fed into WLS-SVR until the architecture is completely determined. Once the growth has finished, the training and validation subsets are merged and WLS-SVR is invoked for a last update of the regressor weights, to maximally exploit the available data for training the best machine.

#### 3.2 Mixture of Kernels to Solve a Multi-resolution Regressor: MK-GSVR

In some applications, it is not clear *a priori* which kernel function is the most appropriate, and it might be desirable to train a more flexible SVM by combining different kernels to solve a given problem. In what follows we will restrict to the Gaussian case, where combining kernels with different  $\sigma$  value can be interpreted as a multi-resolution approach, observing the problem at different scales of detail, although, as shown in [1], mixture of different kernels is also possible.

It is necessary to reformulate the algorithms to support multi-kernel, by combining several discriminant functions with different kernels and globally maximizing the flatness of the resulting function. The multi-kernel SVR is:

$$f(\mathbf{x}) = \sum_{j=1}^J \phi_j^T(\mathbf{x}) \mathbf{w}_j + b, \quad (9)$$

where  $\phi_j(\cdot)$  are the mapping functions corresponding to the  $J$  different kernels and  $b$  includes all the bias terms. We obtain  $\mathbf{w}_j^T$  by jointly maximizing the flatness of the function in each space  $\mathcal{F}_j$  ( $\mathcal{F}_j$  corresponds to projection  $\phi_j(\cdot)$ )

$$L_{MK} = \sum_{j=1}^J \frac{\delta_j}{2} \|\mathbf{w}_j\|^2 - \sum_{i=1}^l \alpha_i [\epsilon - y_i + f(\mathbf{x}_i)] - \sum_{i=1}^l \alpha_i^* [\epsilon + y_i - f(\mathbf{x}_i)], \quad (10)$$

where coefficients  $\{\delta_j\}$  represent a set of (possible) weights for the combination of kernels. Then, dropping the bias term for simplicity, substituting (3) and (9) in (10) and minimizing we arrive to

$$\delta_j \mathbf{w}_j - \Phi_j^T D_{\mathbf{a}} [\mathbf{y} - \Phi W - \epsilon] + \Phi_j^T D_{\mathbf{a}^*} [\Phi W - \mathbf{y} - \epsilon] = \mathbf{0} \quad j = 1, \dots, J, \quad (11)$$

where  $\Phi_j = [\phi_j(\mathbf{x}_1) \cdots \phi_j(\mathbf{x}_N)]^T$ ,  $D_{\mathbf{a}^{(*)}}$  are a diagonal matrices with  $(D_{\mathbf{a}^{(*)}})_{ii} = a_i^{(*)}$ ,  $\Phi = [\Phi_1 \cdots \Phi_J]$  and  $W = [\mathbf{w}_1^T \cdots \mathbf{w}_J^T]^T$ . We can further combine the equations in (11) obtaining:

$$\Lambda W - \Phi^T D_{\mathbf{a}} [\mathbf{y} - \Phi W - \epsilon] + \Phi^T D_{\mathbf{a}^*} [\Phi W - \mathbf{y} - \epsilon] = \mathbf{0}, \quad (12)$$

where  $\Lambda$  is a diagonal matrix defined as

$$\Lambda = \begin{bmatrix} \delta_1 I_1 & 0 & \cdots & 0 \\ 0 & \delta_2 I_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \delta_J I_J \end{bmatrix}$$

where  $\{I_j\}$  are identity matrices whose sizes are in concordance with  $\{\mathbf{w}_j\}$ . Finally we rearrange (12) leaving the terms that depend on  $W$  in the left-hand side of the equation and moving the independent terms to the right-hand side:

$$[\Lambda + \Phi^T D_{\mathbf{a}+\mathbf{a}^*} \Phi] W = \Phi^T D_{\mathbf{a}+\mathbf{a}^*} [\mathbf{y} - \mathbf{E}\epsilon], \quad (13)$$

where  $\mathbf{E} = D_{\mathbf{a}+\mathbf{a}^*}^{-1} D_{\mathbf{a}-\mathbf{a}^*} \mathbf{1}$  and we have applied the identity  $D_{\mathbf{a}\pm\mathbf{a}^*} = D_{\mathbf{a}} \pm D_{\mathbf{a}^*}$  for diagonal matrices. At this point, we introduce the parametric approximation, and we assume that the solution of the equation has a form

$$\mathbf{w}_j = \Psi_j^T \beta_j \quad j = 1, \dots, J, \quad (14)$$

where  $\Psi_j = [\phi_j(\mathbf{c}^j_1) \cdots \phi_j(\mathbf{c}^j_{R_j})]^T$ . Now, we can write the solution of the system in terms of the parametric approximation  $W = \Psi^T \beta$ , where  $\Psi$  is a diagonal matrix with  $(\Psi)_{ii} = \Psi_i$  and  $\beta = [\beta_1^T \cdots \beta_J^T]^T$ . This parametric solution is included in (13). Multiplying each side of (13) by  $\Psi$  and grouping terms, we arrive to:

$$[\Psi \Lambda \Psi^T + \Psi \Phi^T D_{\mathbf{a}+\mathbf{a}^*} \Phi \Psi^T] \beta = \Psi \Phi^T D_{\mathbf{a}+\mathbf{a}^*} [\mathbf{y} - \mathbf{E}\epsilon]. \quad (15)$$



Products between matrices  $\Phi$  and  $\Psi$  are inner products of vectors in the projected spaces, and can be written in their kernel forms:

$$[I_\Psi + K^T D_{\mathbf{a}+\mathbf{a}^*} K] \boldsymbol{\beta} = K^T D_{\mathbf{a}+\mathbf{a}^*} [\mathbf{y} - \mathbf{E}\boldsymbol{\epsilon}], \quad (16)$$

where  $I_\Psi = \Psi \Lambda \Psi^T$  and  $K = \Phi \Psi^T$ :

$$I_\Psi = \begin{bmatrix} \delta_1 \Psi_1 \Psi_1^T & 0 & \dots & 0 \\ 0 & \delta_2 \Psi_2 \Psi_2^T & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \delta_J \Psi_J \Psi_J^T \end{bmatrix}, \quad K = [\Phi_1 \Psi_1^T \dots \Phi_J \Psi_J^T]. \quad (17)$$

Here, it is worth remarking that, due to the block-diagonal structure of matrices  $I_\Psi$  and  $\Psi$ , inner products between projected vectors corresponding to different mappings associated to different kernels  $k_j$ ,  $k_i$  with  $j \neq i$  never appear, i.e. all the inner products involve well defined projections and can be calculated via the corresponding kernel.  $I_\Psi$  and  $K$  can be calculated with kernels using:

$$I_{\Psi_j} = \delta_j \Psi_j \Psi_j^T, \quad (I_{\Psi_j})_{mn} = \delta_j k_j(\mathbf{c}_m^j, \mathbf{c}_n^j), \quad m, n = 1, \dots, R_j, \quad (18)$$

$$K_j = \Phi_j \Psi_j^T, \quad (K_j)_{ik} = k_j(\mathbf{x}_i, \mathbf{c}_k^j), \quad i = 1, \dots, l, \quad k = 1, \dots, R_j. \quad (19)$$

Once we have calculated matrices  $I_\Psi$  and  $K$ , (16) can be solved via WLS-SVR to obtain the parametric approximation to the MK-SVR formulated in (13).

A growing scheme for the Gaussian MK-SVR case can be easily implemented based on the lines explained in Section 3.1. The machine is initialized with few centroids and with a high value of parameter  $\sigma$ , so that data are examined with a coarse resolution. The GSVR proceeds normally until the MSE of the validation set shows that training has saturated. Then,  $\sigma$  is decreased in order to process in a deeper detail (lower resolution) the parts of the regression function that are still badly learned. The value of  $\sigma$  is decreased after each saturation of the learning until no improvement is observed in the validation error. Moreover, from iteration to iteration there is no need to recalculate all the kernel matrices, but just to add the blocks corresponding to the new nodes.

## 4 Experimental Results

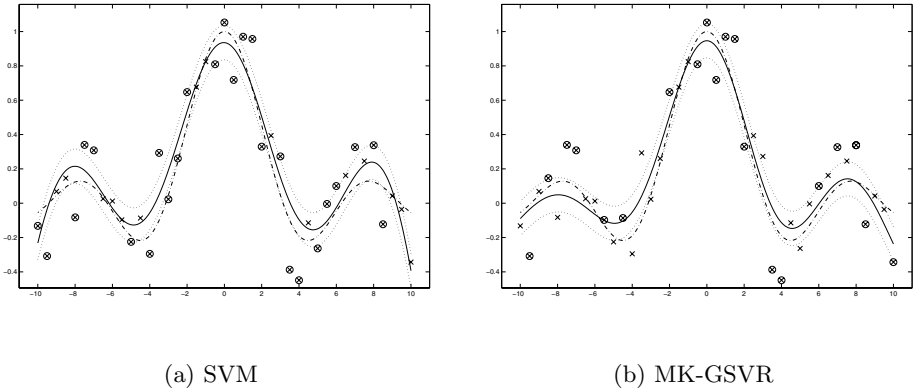
In this section we present some results showing the performance of our algorithm when compared with the standard SVR algorithm<sup>2</sup>. First we present a toy example in 1D that allows us to see graphically how both algorithms work. Then we present the results on three real-world problems.

### 4.1 Toy Example

The sinc function is largely used to present the SVR capabilities [7], because it is a well-known simple one dimensional curve (so it can be plotted). We have used

<sup>2</sup> We have used for our experiments SVMlight [3].

51 training samples, which were equally spaced ranging from -10 to 10 for the variable  $x$  and we have added zero-mean gaussian noise with  $\sigma_n = 0.18$  to the variable  $y = \frac{\sin(x)}{x}$ . An RBF kernel has been selected, and cross-validation has been used for selecting the optimal parameters for each case:  $C = 1000$ ,  $\epsilon = 0.1$  and  $\sigma = 5$  for SVMlight, and  $C = 2$ ,  $\epsilon = 0.1$  and  $ncc = 1$  (number of centroids added at each iteration) for MK-GSVR. Then we test the resulting architectures with 401 samples, equally spaced ranging from -10 to 10 for the variable  $x$  and the variable  $y = \frac{\sin(x)}{x}$  without noise.



**Fig. 1.** Regressor functions. Continuous line is the regressor function, dashed line is the original sinc function, dotted line shows the insensitive region, “x” shows training patterns and “o” denotes the support vectors (a) or centroids (b)

Both results are depicted in Figure 1. MK-GSVR performs an error reduction around 40% with respect to SVMlight, with  $MSE_{MK-GSVR} = 0.0047$  and  $MSE_{SVMlight} = 0.0075$ . Also the resulting architecture is smaller in MK-GSVR: 21 centroids for MK-GSVR and 26 support vectors for SVMlight. We can also see how MK-GSVR selects centroids not only among the patterns outside the  $\epsilon$ -tube (the support vectors of the standard SVM), but also inside.

## 4.2 Real-World Problems

We have used two real data sets to test the performance (in MSE prediction error and size of the resulting machine) between the QP-SVR [3] and the MK-GSVR. Those data sets are Boston Housing, CPU-performance<sup>3</sup> and Kin<sup>4</sup>. We have used a five-fold cross-validation method for selecting the optimal parameters as described in section 4.1. The results are shown in Table 1. We can see how the performance of MK-GSVR is comparable to QP-SVM, in terms of Mean Squared

<sup>3</sup> Taken from <ftp://ftp.ics.uci.edu/pub/machine-learning-databases/>.

<sup>4</sup> From <http://www.cs.toronto.edu/delve/data/kin/desc.html>.

**Table 1.** Results for Boston Housing and CPU performance. Mean values and standard deviation (in parenthesis) of minimum squared error and machine size are shown

| Data set        | SVMLight               |               | MK-GSVR              |                     |
|-----------------|------------------------|---------------|----------------------|---------------------|
|                 | MSE                    | SVs           | MSE                  | centroids           |
| Boston Housing  | 19.24 (17.84)          | 397.4 (1.67)  | <b>18.15 (12.33)</b> | <b>94.4 (43.53)</b> |
| CPU performance | <b>5182.3 (7044.8)</b> | 158.6 (3.91)  | 5285.9 (6970.4)      | <b>49 (18)</b>      |
| Kin             | <b>0.0599 (0.0029)</b> | 4696.8 (31.5) | 0.0625 (0.0035)      | <b>63.6 (18.1)</b>  |

Error. What is more important, MK-GSVR yields much smaller machines, about 1/4, 1/3 and even 1/75 smaller, respectively.

## 5 Concluding Remarks

We have presented an algorithm that iteratively grows a support vector regressor (GSVR) in a problem-oriented form. In its multi-kernel version (MK-GSVR), it is capable of solving function regression problems in a multi-resolution approach, with the advantage of skipping the problem of an ‘a priori’ estimation of kernel hyperparameters. The algorithm presented here is simple, efficient and allows to control the trade-off between machine complexity and performance in terms of prediction error. Experimental results in several benchmark problems point out that MK-GSVR generalization is comparable to SVRs. Most prominently, the experiments show an important reduction on the final machine size built by MK-GSVR with respect to the original SVR.

Ongoing research includes schemes of Growing Support Vector Regressors with mixtures of different types of kernels, not just kernels with variable hyperparameters [1]. Under this general formulation we are also developing adaptive and distributed [5] versions of the algorithm, in order to deal with new problems present in Data Mining applications over communication networks.

## References

1. J. P. Cabrera-Borges, E. Parrado-Hernández, A. R. Figueiras-Vidal and A. Navia-Vázquez. Kernel hybrid adaptive Recursive Least Squares. *Proceedings of the Learning’04 Conference*, pages 30-38, Elche (Spain), 2004.
2. D. Gutiérrez-González, E. Parrado-Hernández and A. Navia-Vázquez. Mega-GSVC: Training SVMs with millions of data. *Proceedings of the Learning’04 Conference*, pages 73-78, Elche (Spain), 2004.
3. T. Joachims. Making large scale SVM learning practical. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods—Support Vector Learning*, pages 169–184. M.I.T. Press, Cambridge, (MA), 1999.
4. A. Navia-Vázquez, F. Pérez-Cruz, A. Artés-Rodríguez, and A. R. Figueiras-Vidal. Weighted least squares training of support vector classifiers leading to compact and adaptive schemes. *IEEE Trans. Neural Networks*, 12(5):1047 – 1059, 2001.

5. A. Navia-Vázquez, D. Gutiérrez-González and E. Parrado-Hernández. Distributed Support Vector Machines. Submitted to *IEEE Trans. Neural Networks*, 2004.
6. E. Parrado-Hernández, I. Mora-Jiménez, J. Arenas-García, A. R. Figueiras-Vidal, and A. Navia-Vázquez. Growing support vector classifiers with controlled complexity. *Pattern Recognition*, 36:1479–1488, 2003.
7. F. Pérez-Cruz, A. Navia-Vázquez, P. Alarcón-Diana, and A. Artés-Rodríguez. An IRWLS procedure for SVR. In *Proc. of the EUSIPCO'00*, 2000.
8. B. Schölkopf, S. Mika, C. J. C. Burges, P. Knirsch, K-R. Müller, G. Rätsch, and A. J. Smola. Input space versus feature space in kernel-based methods. *IEEE Trans. Neural Networks*, 10(5):1000–1017, 1999.
9. B. Schölkopf and A. J. Smola. *Learning with kernels*. MIT Press, Cambridge MA, 2002.
10. A. J. Smola and B. Schölkopf. Sparse greedy matrix approximation for machine learning. In P. Langley, editor, *Proc. of the 17th International Conference on Machine Learning*, pages 911 – 918, San Francisco CA, 2000. Morgan Kaufman.
11. V. Vapnik. *The nature of statistical learning theory*. Springer-Verlag, New York, 1995.

# Stability Results for Cellular Neural Networks with Time Delays

Daniela Danciu<sup>1</sup> and Vladimir Răsvan<sup>2</sup>

University of Craiova, Department of Automatic Control,  
13, A.I. Cuza str., 200585 - Craiova, Romania

{Daniela.Danciu, Vladimir.Rasvan}@automation.ucv.ro

**Abstract.** Cellular neural networks (CNNs) introduced by Chua and Yang in 1988 are recurrent artificial neural networks. Due to their cyclic connections and to the neurons' nonlinear activation functions, recurrent neural networks are nonlinear dynamic systems, which display stable and unstable fixed points, limit cycles and chaotic behavior. Since the field of neural networks is still a recent one, improving the stability conditions for such systems is an obvious and quasi-permanent task. This paper focuses on CNNs affected by time delays. We are interested to obtain sufficient conditions for the asymptotic stability of a cellular neural network with time delay feedback and zero control templates. Due to their sector restricted nonlinearities, stability of the neural networks is strongly connected to robust stability. With respect to this we shall use a quadratic Liapunov functional constructed *via* the technique due to V. L. Kharitonov for uncertain linear time delay systems, combined with an approach suggested by Malkin for systems with sector restricted nonlinearities.

## 1 Introduction

Cellular neural networks (CNNs), introduced by Chua and Yang in 1988 [2], are artificial recurrent neural networks displaying 2D or 3D arrays of identical nonlinear dynamical systems (the so-called cells) locally interconnected. CNNs have been successfully applied to signal and image processing, shape extraction and edge detection. In such applications stability and other problems of dynamical behavior of the CNN are equally important. These properties are necessary for the network to achieve its goal and have to be checked on the CNN mathematical model.

In the last ten years the research was oriented towards the dynamics of the networks affected by time delays due to the signal propagation at the synapses level of the biologic brain or the reacting lag in the case of the artificial neural network. These lags may introduce oscillations or may lead to instability of the network.

Our primary interest is to obtain sufficient conditions for the asymptotic stability of a CNN with time delay feedback and zero control template. With respect to this it is important to point out that the nonlinearities of the neural networks being sigmoidal, belong to the class of the sector restricted (Lurie type) nonlinearities what sends to the absolute stability problem; at its turn this problem (of stability with respect to an entire *class* of nonlinear functions) appears in a more contemporary setting as a robustness problem.

Since the simplest Liapunov function(al) in this case is the quadratic one, we focus on such function(al)s. In the linear case a quadratic Liapunov functional may provide necessary and sufficient conditions for exponential stability, but in the time delay case the sharpest most general quadratic Liapunov function (as suggested by the papers of Datko and Infante with Castelan - their exact references are to be found in [6] is rather difficult to manipulate. On the other hand, the simplified versions which are currently used (including our earlier reference [3], [4], [5]) deserve improvement.

For this reason we adopted the construction of V. L. Kharitonov [6], [9] which is mainly based on the *algebraic Liapunov equation* idea: given a certain structure of the Liapunov function(al) derivative, the functional itself, together with its derivative is constructed (solving  $A^T P + P A = -Q$  with respect to  $P$  when  $Q$  is given, but in the case of the time delay systems). Now, the approach of Kharitonov is suitable mainly for uncertain linear systems, *the uncertainty being subject to some quadratic inequalities*.

The *sector restricted nonlinearities* are also *subject to quadratic inequalities* hence they are within the same class of problems. We shall apply a method due to I. G. Malkin in the early stage of the absolute stability studies [1], [10] which takes advantage of the properties of the quadratic Liapunov function(al), being rather simple and straightforward. Since this approach of Malkin was introduced and applied to systems with a single nonlinearity, the present paper is also *an extension of the method of Malkin to the case of several nonlinearities and time delay systems*.

## 2 The Mathematical Model – Problem Statement

**A.** Consider a cellular neural network with time delay feedback and zero control templates

$$\dot{z}_i(t) = -a_i z_i(t) + \sum_{j \in \mathcal{N}} c_{ij} g_j(z_j(t - \tau_j)) + I_i, \quad i = \overline{1, n} \tag{1}$$

where  $j$  is the index for the cells of the nearest neighborhood  $\mathcal{N}$  of the  $i^{th}$  cell,  $a_i$  is a positive parameter,  $c_{ij}$  are synaptic weights (which can have an inhibitory effect if  $c_{ij} < 0$ , or an excitatory one if  $c_{ij} > 0$ ),  $I_i$  is the bias and  $\tau_j$  are positive delays.

The nonlinearities for the cellular neural networks are e.g. of the bipolar ramp type:

$$g_i(z_i) = \frac{1}{2} (|z_i + 1| - |z_i - 1|) \tag{2}$$

hence they are bounded, nondecreasing and globally Lipschitzian functions, with the Lipschitz constant  $L_i = 1$ . Worth mentioning that other sigmoidal nonlinear functions which are to be met in neural networks may also be considered: they are also nondecreasing and globally Lipschitzian, possibly with other Lipschitz constants.

Without loss of the generality, using a change of the coordinates,  $x_i = z_i - z_i^*$ , one can shift the equilibrium point  $z^*$  to the origin so that system (1) can be written into the form:

$$\dot{x}_i(t) = -a_i x_i(t) + \sum_{j \in \mathcal{N}} c_{ij} f_j(x_j(t - \tau_j)), \quad i = \overline{1, n} \tag{3}$$

where we denoted

$$f_j(\sigma) = g_j(\sigma + z_j^*) - g_j(z_j^*), \quad \forall j. \tag{4}$$

These functions are subject to sector restrictions. For instance, if  $g_j(z_j)$  are given by (2) then since they are nondecreasing and globally Lipschitzian with the Lipschitz constant  $L_i = 1$  we shall have, taking also into account the above definition of  $f_i$  that

$$0 < \frac{f_i(\sigma)}{\sigma} \leq 1 \tag{5}$$

Denoting

$$A_0 = \text{diag}(-a_i)_1^n, \quad C_j = \begin{pmatrix} 0 & \dots & 0 & c_{1j} & 0 & \dots & 0 \\ 0 & \dots & 0 & c_{2j} & 0 & \dots & 0 \\ \vdots & \dots & \vdots & \vdots & \vdots & \dots & \vdots \\ 0 & \dots & 0 & c_{nj} & 0 & \dots & 0 \end{pmatrix}, \quad A_j = C_j \cdot \text{diag}(k_i)_1^n \tag{6}$$

with  $c_{ij} = 0$  when  $j \notin \mathcal{N}$  system (3) may be written into the form

$$\dot{x}(t) = A_0x(t) + \sum_{j=1}^n A_jx(t - \tau_j) \tag{7}$$

provided we take  $f_j(\sigma) \equiv k_j\sigma$ , with the initial condition  $x_i(\theta) = \varphi(\theta)$ , for  $\theta \in [-\tau, 0]$ , where  $\tau = \max_j \tau_j$ ,  $\varphi \in \mathcal{C}(-\tau, 0; \mathbb{R}^n)$ . Remark that in the nonlinear case system (3) reads as

$$\dot{x}(t) = A_0x(t) + \sum_{j=1}^n C_j \text{diag}(f_\ell(x_\ell(t - \tau_j))) \tag{8}$$

**B.** In order to make the approach of the paper more clear, we shall discuss briefly two robustness problems concerning a general linear system of the form (7) where  $A_j$  are defined by (6) and  $C_j$  possibly by (6). Assume (7) be exponentially stable for given  $k_j$ ,  $j = 1, \dots, n$ , and consider the perturbed system

$$\dot{y}(t) = A_0y(t) + \sum_{j=1}^n C_j \text{diag}(k_\ell + b_\ell)y(t - \tau_j) \tag{9}$$

with  $-\underline{k}_i \leq b_i \leq \bar{k}_i$ ,  $i = 1, \dots, n$ . This system may be re-written as

$$\dot{y}(t) = A_0y(t) + \sum_{j=1}^n (A_j + \Delta_j)y(t - \tau_j) \tag{10}$$

with  $\Delta_j = C_j \text{diag}(b_\ell)$ . The robustness problem with respect to these linear uncertainties is to find conditions such that the perturbed system (10) remains exponentially stable for all  $b_i \in (-\underline{k}_i, \bar{k}_i)$ ,  $\forall i = \overline{1, n}$ . This is ensured by the Kharitonov-like approach [6], [9].

Now, for given  $\sigma \neq 0$  we may always find some  $b_j(\sigma)$  from

$$b_j(\sigma) = \frac{f_j(\sigma)}{\sigma} - k_j \tag{11}$$

and consider the system

$$\dot{y}(t) = A_0 y(t) + \sum_{j=1}^n (A_j + \Delta_j(\sigma)) y(t - \tau_j) \tag{12}$$

with  $\Delta_j(y) := C_j \text{diag}(b_\ell(y_\ell(t - \tau_j)))$ . The idea of this substitution belongs to Malkin. If we succeed in showing stability preservation for  $b_i \in (-\underline{k}_i, \bar{k}_i), \forall i = \overline{1, n}$ , then we have obtained absolute(robust) stability for the nonlinear functions satisfying

$$0 \leq k_i - \underline{k}_i < \frac{f_i(\sigma)}{\sigma} < k_i + \bar{k}_i \tag{13}$$

### 3 The Construction of the Quadratic Liapunov Functional – The Robustness Approach

Given positive definite  $n \times n$  matrices  $P_0, P_j, R_j, j = \overline{1, n}$  let us define on  $\mathcal{C}(-\tau, 0; \mathbb{R}^n)$ , following [6], [9], the positive definite functional

$$W(\phi(\cdot)) = \phi^T(0)P_0\phi(0) + \sum_{j=1}^n \phi^T(-\tau_j)P_j\phi(-\tau_j) + \sum_{j=1}^n \int_{-\tau_j}^0 \phi^T(\theta)R_j\phi(\theta)d\theta. \tag{14}$$

We focus now on system (7) with  $A_j$  defined as above, system that we assume exponentially stable for some  $k_j$ . According to the general theory due to Datko, Infante and Castelan and others, there exists a positive definite quadratic Liapunov functional defined e.g. on  $\mathcal{C}(-\tau, 0; \mathbb{R}^n)$  with its derivative along the solutions of (7), also quadratic and negative definite on the same space. The construction of [6], [9] will give the quadratic functional  $V(\phi)$  such that

$$\frac{d}{dt}V(x_t) = -W(x_t) \tag{15}$$

with  $W$  defined by (14); here  $x_t(\cdot) := x(t + \cdot)$  is, as usual, the state of the time delay system in the Krasovskii-Halanay-Hale notation.

The corresponding Liapunov-Krasovskii functional results of the form

$$\begin{aligned} V(x_t) = & x^T(t)U(0)x(t) + \sum_{j=1}^n 2x^T(t) \int_{-\tau_i}^0 U(-\tau_i - \theta)A_j x(t + \theta)d\theta + \\ & + \sum_{k=1}^n \sum_{j=1}^n \int_{-\tau_k}^0 x^T(t + \theta_2)A_j^T \left( \int_{-\tau_i}^0 U(\theta_1 - \theta_2 + \tau_k + \tau_j)A_j d\theta_1 \right) d\theta_2 + \\ & + \sum_{j=1}^n \int_{-\tau_i}^0 x^T(t + \theta) [(\tau_j + \theta)R_j + P_j] x(t + \theta)d\theta \tag{16} \end{aligned}$$



where, since the system (7) is exponentially stable, the matrix valued function

$$U(\tau) = \int_0^\infty K^T(t)[P_0 + \sum_{j=1}^n (P_j + \tau_j R_j)]K(t + \tau)dt \tag{17}$$

is well defined for all  $t \in \mathbb{R}$ ; here  $K(t)$  is the fundamental (Cauchy) matrix or the state transition matrix associated to the system (7) (see [9]).

Following the steps in [9], the time derivative of Liapunov-Krasovskii functional along the solutions of the perturbed system (10) is

$$\begin{aligned} \frac{d}{dt}V(y_t) &= -W(y_t) + \\ &+ 2[\sum_{j=1}^n \Delta_j y(t + \tau_j)]^T [U(0)y(t) + \sum_{j=1}^n \int_{-\tau_j}^0 U^T(\tau_j + \theta)A_j y(t + \theta)d\theta] \end{aligned} \tag{18}$$

where  $W$  is that of (14). From now on we consider several cases. First, let the uncertainties be linear, hence  $b_j$  are some constant real numbers. We have

$$\Delta_j^T \Delta_j = (C_j \text{diag}(b_\ell))^T C_j \text{diag}(b_\ell) \leq \left(\sum_{i=1}^n c_{ij}^2\right) b_j^2 I \leq \left(\sum_{i=1}^n c_{ij}^2\right) \bar{k}_j^2 I = \varrho_j I \tag{19}$$

i.e. a quadratic restriction on the uncertainties. This will be useful for the estimate of the perturbing term in (18) in order to still obtain a non-positive derivative of the Liapunov functional (16) along (10). It is not difficult, using standard inequalities (in the line of [6]) to obtain the following estimate

$$\begin{aligned} \frac{d}{dt}V(y_t) &\leq -y^T(t)[P_0 - \mu U^T(0)U(0)]y(t) - \sum_{j=1}^n y^T(t - \tau_j)[P_j - \frac{2}{\mu} \rho_j I]y(t - \tau_j) - \\ &- \sum_{j=1}^n \int_{-\tau_j}^0 y^T(t + \theta)[R_j - \mu A_j^T U(\tau_j + \theta)U^T(\tau_j + \theta)A_j]y(t + \theta)d\theta \end{aligned} \tag{20}$$

for some  $\tilde{\mu} > 0$  and  $\mu = \tilde{\mu} \sum_{j=1}^n \bar{k}_j^2 \sum_{i=1}^n c_{ij}^2$ . This is clear from the definition of  $b_j$  and from the fact that  $b_j \leq \bar{k}_j$  hence (20) is fulfilled.

We have constructed a Liapunov - Krasovskii quadratic functional which is strictly positive definite and with the derivative along linear system's solutions at least non-positive; this last property is preserved with respect to the considered uncertainties and this shows a possible robust exponential stability of the linearized system (7). But, as already mentioned, the idea of Malkin [1], [10] gives more - exponential stability of the nonlinear system (8). This will become clear from the short description of the method. Let  $b_i(\sigma)$  be a nonlinear function defined below

$$b_i(\sigma) = \frac{f_i(\sigma)}{\sigma} - k_i, \quad \forall i \tag{21}$$

Now, if the Liapunov function(al) and its derivative - both being quadratic forms - have good sign properties for all  $b_i \in (-\underline{k}_i, \overline{k}_i)$ , then for any fixed  $x_i \neq 0$  one can obtain  $b_i$  from (21) and for  $b_i(x_i) \in (-\underline{k}_i, \overline{k}_i)$  the properties of the Liapunov function(al) do not change.

Moreover it is quite clear that the terms  $b_i$  may be even time varying what shows that  $f_i$  may be time varying within the interval  $(-\underline{k}_i, \overline{k}_i)$  provided they are at least integrable with respect to  $t$ . (Integrability is necessary just to secure existence of the solution for the Cauchy problem in the Carathéodory sense). Also the Lipschitz property has now to hold uniformly with respect to  $t$ .

We may thus state the following

**Theorem 1.** *Let system (7) be exponentially stable. Then system (8) is exponentially stable for all nonlinearities satisfying (13) if there exist positive definite matrices  $P_0, P_j, R_j$ , and a positive value  $\mu$ , such that*

$$P_0 > \mu U^T(0)U(0), \quad P_j > \frac{2}{\mu} \rho_j I, \quad j = 1, \dots, n$$

$$\inf_{-\tau_j \leq \theta \leq 0} (R_j - \mu A_j^T U(\tau_j + \theta) U^T(\tau_j + \theta) A_j) > 0 \tag{22}$$

*Sketch of the proof:* If we use the standard properties of the eigenvalues of positive definite matrices and the ordering of the quadratic forms, the following estimates are obtained

$$\delta \|\phi\|^2 \leq V(\phi) \leq \gamma \|\phi\|^2, \quad \widetilde{W}(\phi) \geq \varepsilon W(\phi)$$

for some positive  $\delta, \gamma, \varepsilon$  and with  $\|\phi\| = \sup_{-\tau \leq \theta \leq 0} |\phi(\theta)|$ , the usual norm of the uniform convergence on  $\mathcal{C}(-\tau, 0; \mathbb{R}^n)$ , where  $\widetilde{W}$  is the right hand side of (20). From now on we have only to apply standard results of stability theory for time delay systems based on quadratic functionals [7], [8], [13] to obtain the result.

### 4 The Case of the Dynamics of the Neural Networks

The system discussed previously had a structure which was alike (3) but under a general structure for  $A_0$  and with a stability assumption for some  $k_i$  in the linear case. This assumption is much alike to the so-called *minimal stability* introduced by V. M. Popov [11]: in order to obtain stability for *all* nonlinear (and linear) functions from some sector, it is (minimally) necessary to have this property for *a single linear function* within this sector.

For (3) we have  $A_0$  diagonal and of Hurwitz type since  $A_0 = -diag(a_i)$  and  $a_i > 0$ . Therefore, taking also into account (5), we may take  $k_i = 0$ . The state transition matrix  $K(t)$  in (17) is now the transition matrix of  $A_0$  namely  $K(t) = \exp(A_0 t) = diag(\exp(-a_i t))$ . Therefore  $U(\tau) = U(0) \exp(A_0 \tau)$  and

$$V(\phi) = \phi^T(0)U(0)\phi(0) + \sum_{j=1}^n \int_{-\tau_j}^0 \phi^T(\theta)(P_j + (\tau_j + \theta)R_j)\phi(\theta)d\theta \tag{23}$$

which clearly is strictly positive definite - from the properties of  $U(\cdot), P_i, R_j$ . The conditions of **Theorem 1** are modified taking into account that  $k_j = 0$  hence  $A_j = 0$ . The result in the case of (3) is as follows

**Theorem 2.** *System (3) is exponentially stable for sigmoidal functions, in particular for (2) provided there exist positive definite matrices  $P_0, P_j, R_j$ , and a positive value  $\mu$ , such that*

$$P_0 > \mu U^T(0)U(0), \quad R_j > 0, \quad P_j - 2(\bar{k}_j^2 (\sum_{i=1}^n c_{i,j}^2) / \mu) I > 0 \quad (24)$$

Remark that the conditions are valid regardless the values of the delays hence we obtained what is usually called *delay independent stability*. Also stability is *exponential* since both the Liapunov functional and its derivative are quadratic functionals. Stability is also *global* since the functional and the inequality for the derivative are valid globally.

## 5 Conclusions and Future Research

The paper presents new results concerning stability of CNNs with time delays; it is somehow a continuation of previous research in the field of CNNs [3], [4] [5]. The approach being based on the Liapunov method, only sufficient stability conditions have been obtained, like in other research. From this point of view our results are in the standard line. Their specific features come from the extended use of the methods of the absolute stability in the area of time delay systems with sector restricted nonlinearities [12]. It is this point of view that lead us to consider the approach of Malkin in the absolute stability applied to our case. The first contribution of the paper is thus *application of this approach to systems with several nonlinear functions*: this class of systems is unavoidable if we deal with neural networks while in the original work of Malkin [10] as well as in further development [1] only single nonlinearity systems have been considered.

Next result of the paper is connected to the Liapunov functional involved throughout the paper. As it is the case in the study of the absolute stability, the chosen functional has been of quadratic type. The construction of a quadratic Liapunov functional for linear time delay systems is in fact the problem of solving the Liapunov operator equation  $A^T P + P A = -Q$  on some functional space,  $A$  being a specific  $C_0$ -semigroup generator. The early paper of Repin, further results of Datko and Infante with Castelan (see [6], [9] for complete reference) showed the difficulties of the construction. Usual simplifications, also used in our previous papers lead to conservative results. The genuine breakthrough from [6], [9] allowed us to obtain improved sufficient stability criteria. We think this is due also to the context offered by this Liapunov functional construction to the approach of Malkin; *this approach thus knew an additional extension to the time delay case*.

But, as already pointed out, obtaining sharper criteria is a quasi-permanent task. This goal is achievable by improving sharpness of the estimates i.e. by making use of sharper inequalities. The researchers are thus in position to apply the entire set of procedures and results of the Liapunov methods, also to complete and to extend it. Moreover, the Liapunov method remains the basic one in coping with such problems as oscillations and several equilibria; worth mentioning that these problems are crucial in the dynamics of the neural networks.

## References

1. Barbashin, E.A.: Liapunov functions (in Russian), 58–60. Nauka Publ. House, Moscow (1970)
2. Chua, L., Yang, L.: Cellular neural networks: theory and applications. *IEEE Trans. Circuits and Systems CAS-35* (1988) 1257–1290
3. Danciu, D.: Stability analysis of cellular neural networks with time delays. In: *Int. Symp. Syst. Th., Rob., Comp. & Proc. Inf. - SINTES 11, Craiova, Romania* (2003) 32–34
4. Danciu, D.: A method for time delay cellular neural networks stability analysis. In: *Int. Carpathian Contr. Conf. '2004 - ICC'2004, Zakopane, Poland* (2004) 65–70
5. Danciu, D., Răsvan, V.I.: Stability Criteria for Cellular Neural Networks. *Annals "Dunărea de Jos" University of Galați, Series: Electrotechnics, Electronics, Automatic Control, Informatics* **3** (2000)
6. Gu, K., Kharitonov, V.L., Chen, J.: *Stability and robust stability of time delay systems*. Birkhäuser, Boston (2003)
7. Halanay, A.: *Differential Equations. Stability. Oscillations. Time Lags* (in Romanian). Editura Academiei, Bucharest (1963) (English improved version by Academic Press, Mathematics in Science and Engineering Series **23**, 1966)
8. Hale J. K., Verduyn Lunel, S. M.: *Introduction to Functional Differential Equations*. Springer Verlag, Berlin-Heidelberg-New York (1993)
9. Kharitonov V.L., Zhabko, A.P.: Lyapunov-Krasovskii approach to the robust stability analysis of time-delay systems. *Automatica* **39** (2003) 15–20
10. Malkin, I.G.: *Stability of Motion*. (in Russian) Gostekhizdat, Moscow (1952)
11. Popov, V. M.: *Hyperstability of Control Systems*. (in Romanian) Editura Academiei, Bucharest (1966) (English improved version by Springer Verlag, 1973)
12. Răsvan, V.I.: *Absolute Stability of Time Lag Control Systems* (in Romanian). Editura Academiei, Bucharest (1975) (Russian improved version by Nauka Publ. House, Moscow, 1983)
13. Yoshizawa, T.: *Stability theory by Liapunov's second method*. Math. Soc. Japan, Tokyo (1966)

# Global Exponential Stability Analysis in Cellular Neural Networks with Time-Varying Coefficients and Delays<sup>\*</sup>

Qiang Zhang<sup>1</sup>, Dongsheng Zhou<sup>2</sup>, Xiaopeng Wei<sup>1</sup>, and Jin Xu<sup>1</sup>

<sup>1</sup> University Key Lab of Information Science & Engineering,  
Dalian University, Dalian, 116622, China

<sup>2</sup> School of Mechanical Engineering, Dalian University of Technology,  
Dalian, 116024, China  
zhangq30@yahoo.com

**Abstract.** Global exponential stability of cellular neural networks with time-varying coefficients and delays is considered in this paper. By utilizing a delay differential inequality, a new sufficient condition ensuring global exponential stability for cellular neural networks with time-varying coefficients and delays is presented. Since the condition does not require that the delay function be differentiable or the coefficients be bounded, the results here improve and extend those given in the earlier literature.

## 1 Introduction

Since cellular neural networks with delay were introduced [1], they have been extensively studied in the past decade and successfully applied to moving image treatment. Since these applications rely on qualitative properties of stability, the stability of cellular neural networks with delay have been deeply investigated and many important results on the global asymptotic stability and global exponential stability of one unique equilibrium point have been given, see, for example, [2]-[21] and references cited therein. However, few studies have considered dynamics for cellular neural networks with time-varying coefficients and delays [22]-[25]. In this paper, by using a delay differential inequality, we discuss the global exponential stability of cellular neural networks with time-varying coefficients and delays and obtain a new sufficient condition.

## 2 Preliminaries

The dynamic behavior of a continuous time cellular neural networks with time-varying coefficients and delays can be described by the following state equations:

---

<sup>\*</sup> The project was supported by the National Natural Science Foundation of China (Grant No. 60403001) and China Postdoctoral Science Foundation.

$$\begin{aligned}
 x'_i(t) &= -c_i(t)x_i(t) + \sum_{j=1}^n a_{ij}(t)f_j(x_j(t)) \\
 &+ \sum_{j=1}^n b_{ij}(t)f_j(x_j(t - \tau_j(t))) + I_i(t).
 \end{aligned}
 \tag{1}$$

where  $n$  corresponds to the number of units in a neural networks;  $x_i(t)$  corresponds to the state vector at time  $t$ ;  $c_i(t) > 0$  represents the rate with which the  $i$ th neuron will reset its potential to the resting state in isolation when disconnected from the network and external inputs.  $f(x(t)) = [f_1(x_1(t)), \dots, f_n(x_n(t))]^T \in R^n$  denotes the activation function of the neurons;  $A(t) = [a_{ij}(t)]_{n \times n}$  is referred to as the feedback matrix,  $B(t) = [b_{ij}(t)]_{n \times n}$  represents the delayed feedback matrix, while  $I_i(t)$  is an external bias vector at time  $t$ ,  $\tau_i(t)$  is the transmission delay along the axon of the  $j$ th unit and satisfies  $0 \leq \tau_i(t) \leq \tau$ .

Throughout this paper, we will assume that the real valued functions  $c_i(t) > 0, a_{ij}(t), b_{ij}(t), I_i(t)$  are continuous functions. The activation functions  $f_i, i = 1, 2, \dots, n$  are assumed to satisfy the following conditions (H)

$$|f_i(\xi_1) - f_i(\xi_2)| \leq L_i |\xi_1 - \xi_2|, \forall \xi_1, \xi_2.$$

This type of activation functions is clearly more general than both the usual sigmoid activation functions and the piecewise linear function (PWL):  $f_i(x) = \frac{1}{2}(|x + 1| - |x - 1|)$  which is used in [26].

The initial conditions associated with system (1) are of the form

$$x_i(s) = \phi_i(s), \quad s \in [-\tau, 0]$$

in which  $\phi_i(s)$  are continuous for  $s \in [-\tau, 0]$ .

We say that system (1) is globally exponentially stable if there exist constants  $\varepsilon > 0$  and  $M \geq 1$  such that for any two solutions  $x(t) = (x_1(t), x_2(t), \dots, x_n(t))^T$  and  $z(t) = (z_1(t), z_2(t), \dots, z_n(t))^T$  with the initial conditions  $x(s) = \phi(s)$  and  $z(s) = \varphi(s)$  for  $s \in [-\infty, 0]$  respectively, one has

$$\|x(t) - z(t)\| \leq M \|\phi - \varphi\| e^{-\varepsilon t}$$

for all  $t \geq 0$ , where the norm is defined as

$$\|\phi\| = \sup_{-\infty \leq s \leq 0} \left[ \sum_{i=1}^n |\phi_i(s)|^2 \right]^{1/2}$$

**Lemma 1.** [27] *Let  $x(t)$  be a continuous nonnegative function on  $t \geq t_0 - \tau$  satisfying inequality (2) for  $t \geq t_0$ .*

$$D^+x(t) \leq -k_1(t)x(t) + k_2(t)\bar{x}(t) \tag{2}$$

where  $\bar{x}(t) = \sup_{t-\tau \leq s \leq t} \{x(s)\}$ . If  $k_1(t)$  or  $k_2(t)$  is bounded, and  $\alpha = \inf_{\{t \geq t_0\}} \{k_1(t) - k_2(t)\} > 0$ , then there must exist a positive  $\eta > 0$  such that

$$x(t) \leq \bar{x}(t_0) \exp\{-\eta(t - t_0)\} \tag{3}$$

holds for all  $t \geq t_0 - \tau$ .

### 3 Stability Analysis

In this section, we will use the above Lemma to establish the exponential stability of system (1). Consider two solutions  $x(t)$  and  $z(t)$  of system (1) for  $t > 0$  corresponding to arbitrary initial values  $x(s) = \phi(s)$  and  $z(s) = \varphi(s)$  for  $s \in [-\tau, 0]$ . Let  $y_i(t) = x_i(t) - z_i(t)$ , then we have

$$\begin{aligned}
 y'_i(t) = & -c_i(t)y_i(t) + \sum_{j=1}^n a_{ij}(t) (f_j(x_j(t)) - f_j(z_j(t))) \\
 & + \sum_{j=1}^n b_{ij}(t) (f_j(x_j(t - \tau_j(t))) - f_j(z_j(t - \tau_j(t))))
 \end{aligned} \tag{4}$$

Let  $g_j(y_j(t)) = f_j(y_j(t) + z_j(t)) - f_j(z_j(t))$ , one can rewrite Eq.(4) as

$$y'_i(t) = -c_i(t)y_i(t) + \sum_{j=1}^n a_{ij}(t)g_j(y_j(t)) + \sum_{j=1}^n b_{ij}(t)g_j(y_j(t - \tau_j(t))) \tag{5}$$

Note that the functions  $f_j$  satisfy the hypothesis (H), that is,

$$\begin{aligned}
 |g_i(\xi_1) - g_i(\xi_2)| & \leq L_i|\xi_1 - \xi_2|, \forall \xi_1, \xi_2. \\
 g_i(0) & = 0
 \end{aligned} \tag{6}$$

**Theorem 1.** *Let*

$$\begin{aligned}
 k_1(t) = & \min_i \left[ 2c_i(t) - \sum_{j=1}^n L_j^{2\alpha_{ij}} |a_{ij}(t)|^{2\beta_{ij}} \right. \\
 & \left. - \sum_{j=1}^n \frac{\gamma_j}{\gamma_i} L_i^{2-2\alpha_{ji}} |a_{ji}(t)|^{2-2\beta_{ji}} - \sum_{j=1}^n L_j^{2\zeta_{ij}} |b_{ij}(t)|^{2\eta_{ij}} \right] \\
 k_2(t) = & \max_i \left[ \sum_{j=1}^n \frac{\gamma_j}{\gamma_i} L_i^{2-2\zeta_{ji}} |b_{ji}(t)|^{2-2\eta_{ji}} \right]
 \end{aligned} \tag{7}$$

where  $\alpha_{ij}, \beta_{ij}, \zeta_{ij}$  and  $\eta_{ij}$  are real constants and  $\gamma_i > 0$  is a positive constant. Eq.(1) is globally exponentially stable if

$$\alpha = \inf_{t \geq t_0} \{k_1(t) - k_2(t)\} > 0$$

*Proof.* Let  $z(t) = \frac{1}{2} \sum_{i=1}^n \gamma_i y_i^2(t)$ , Calculating the derivative of  $z(t)$  along the solutions of (5), we get

$$\begin{aligned}
 z'(t) & = \sum_{i=1}^n \gamma_i y_i(t) y'_i(t) \\
 & = \sum_{i=1}^n \gamma_i y_i(t) \left[ -c_i(t)y_i(t) + \sum_{j=1}^n a_{ij}(t)g_j(y_j(t)) \right.
 \end{aligned}$$

$$\begin{aligned}
 & \left. + \sum_{j=1}^n b_{ij}(t)g_j(y_j(t - \tau_j(t))) \right] \\
 \leq & \sum_{i=1}^n \gamma_i \left[ -c_i(t)y_i^2(t) + \sum_{j=1}^n L_j |a_{ij}(t)| |y_i(t)| |y_j(t)| \right. \\
 & \left. + \sum_{j=1}^n L_j |b_{ij}(t)| |y_i(t)| |y_j(t - \tau_j(t))| \right] \tag{8}
 \end{aligned}$$

Recall that the inequality  $2ab \leq a^2 + b^2$  holds for any two real constants  $a, b$ . By using this inequality, we have

$$\begin{aligned}
 L_j |a_{ij}(t)| |y_i(t)| |y_j(t)| &= (L_j^{\alpha_{ij}} |a_{ij}(t)|^{\beta_{ij}} |y_i(t)|) \left( L_j^{1-\alpha_{ij}} |a_{ij}(t)|^{1-\beta_{ij}} |y_j(t)| \right) \\
 &\leq \frac{1}{2} L_j^{2\alpha_{ij}} |a_{ij}(t)|^{2\beta_{ij}} y_i^2(t) \\
 &\quad + \frac{1}{2} L_j^{2-2\alpha_{ij}} |a_{ij}(t)|^{2-2\beta_{ij}} y_j^2(t) \tag{9}
 \end{aligned}$$

Similarly, we have

$$\begin{aligned}
 & L_j |b_{ij}(t)| |y_i(t)| |y_j(t - \tau_j(t))| \\
 &= \left( L_j^{\zeta_{ij}} |b_{ij}(t)|^{\eta_{ij}} |y_i(t)| \right) \left( L_j^{1-\zeta_{ij}} |b_{ij}(t)|^{1-\eta_{ij}} |y_j(t - \tau_j(t))| \right) \\
 &\leq \frac{1}{2} L_j^{2\zeta_{ij}} |b_{ij}(t)|^{2\eta_{ij}} y_i^2(t) \\
 &\quad + \frac{1}{2} L_j^{2-2\zeta_{ij}} |b_{ij}(t)|^{2-2\eta_{ij}} y_j^2(t - \tau_j(t)) \tag{10}
 \end{aligned}$$

Substituting (10) and (9) into (8), we get

$$\begin{aligned}
 z'(t) &\leq \sum_{i=1}^n \gamma_i \left[ -c_i(t)y_i^2(t) + \frac{1}{2} \sum_{j=1}^n L_j^{2\alpha_{ij}} |a_{ij}(t)|^{2\beta_{ij}} y_i^2(t) \right. \\
 &\quad + \frac{1}{2} \sum_{j=1}^n L_j^{2-2\alpha_{ij}} |a_{ij}(t)|^{2-2\beta_{ij}} y_j^2(t) + \frac{1}{2} \sum_{j=1}^n L_j^{2\zeta_{ij}} |b_{ij}(t)|^{2\eta_{ij}} y_i^2(t) \\
 &\quad \left. + \frac{1}{2} \sum_{j=1}^n L_j^{2-2\zeta_{ij}} |b_{ij}(t)|^{2-2\eta_{ij}} y_j^2(t - \tau_j(t)) \right] \\
 &\leq -\frac{1}{2} \sum_{i=1}^n \gamma_i \left[ 2c_i(t) - \sum_{j=1}^n L_j^{2\alpha_{ij}} |a_{ij}(t)|^{2\beta_{ij}} \right. \\
 &\quad \left. - \sum_{j=1}^n \frac{\gamma_j}{\gamma_i} L_i^{2-2\alpha_{ji}} |a_{ji}(t)|^{2-2\beta_{ji}} - \sum_{j=1}^n L_j^{2\zeta_{ij}} |b_{ij}(t)|^{2\eta_{ij}} \right] y_i^2(t)
 \end{aligned}$$



$$\begin{aligned}
 & + \frac{1}{2} \sum_{i=1}^n \gamma_i \left[ \sum_{j=1}^n \frac{\gamma_j}{\gamma_i} L_i^{2-2\zeta_{ji}} |b_{ji}(t)|^{2-2\eta_{ji}} \right] \bar{y}_i^2(t) \\
 & \leq -k_1(t)z(t) + k_2(t)\bar{z}(t)
 \end{aligned}$$

According to Lemma above, if the condition (3) is satisfied, then we have

$$\begin{aligned}
 \frac{1}{2} \gamma_{\min} \sum_{i=1}^n y_i^2(t) & \leq z(t) \leq \bar{z}(t_0) \exp\{-\eta(t - t_0)\} \\
 & = \frac{1}{2} \sum_{i=1}^n \gamma_i \bar{y}_i^2(t_0) \exp\{-\eta(t - t_0)\} \\
 & \leq \frac{1}{2} \gamma_{\max} \sum_{i=1}^n \bar{y}_i^2(t_0) \exp\{-\eta(t - t_0)\}
 \end{aligned}$$

which implies that  $\sum_{i=1}^n y_i^2(t) \leq \frac{\gamma_{\max}}{\gamma_{\min}} \sum_{i=1}^n \bar{y}_i^2(t_0) \exp\{-\eta(t - t_0)\}$ . This completes the proof.

As consequence of Theorem 1, we have the following corollary.

**Corollary 1**

$$\begin{aligned}
 k_1(t) & = \min_i \left[ 2c_i(t) - \sum_{j=1}^n L_j |a_{ij}(t)| \right. \\
 & \quad \left. - \sum_{j=1}^n L_i |a_{ji}(t)| - \sum_{j=1}^n L_j |b_{ij}(t)| \right] \\
 k_2(t) & = \max_i \left[ \sum_{j=1}^n L_i |b_{ji}(t)| \right]
 \end{aligned} \tag{11}$$

Eq.(1) is globally exponentially stable if

$$\alpha = \inf_{t \geq t_0} \{k_1(t) - k_2(t)\} > 0$$

In fact, (11) is a special case of (7) as  $\alpha_{ij} = \beta_{ij} = \zeta_{ij} = \eta_{ij} = \frac{1}{2}(i, j = 1, 2, \dots, n)$ . Therefore, by Theorem 1 we can see that Corollary 1 is true.

*Remark 1.* Note that the criteria obtained here are independent of delay and the coefficients  $c_i(t), a_{ij}(t)$  and  $b_{ij}(t)$  may be unbounded.

## 4 An Example

In this section, we will give an example showing the effectiveness of the condition given here.

*Example 1.* Consider the following non-autonomous delayed neural networks

$$\begin{aligned} x_1'(t) &= -c_1(t)x_1(t) + a_{11}(t)f(x_1(t)) + a_{12}(t)f(x_2(t)) \\ &\quad + b_{11}(t)f(x_1(t - \tau_1(t))) + b_{12}(t)f(x_2(t - \tau_2(t))) \\ x_2'(t) &= -c_2(t)x_2(t) + a_{21}(t)f(x_1(t)) + a_{22}(t)f(x_2(t)) \\ &\quad + b_{21}(t)f(x_1(t - \tau_1(t))) + b_{22}(t)f(x_2(t - \tau_2(t))) \end{aligned} \tag{12}$$

where the activation function is described by  $f_i(x) = \tanh x$ . Clearly,  $f_i(x)$  satisfy hypothesis (H) above with  $L_1 = L_2 = 1$ . For system (12), taking

$$\begin{aligned} c_1(t) &= e^t + \frac{1}{2} \sin t - \frac{1}{2} \cos t + 6; \\ c_2(t) &= 8 - \frac{3}{2} \sin t - \frac{1}{2} \cos t; \\ a_{11}(t) &= e^t + \sin t + 1, \quad a_{12}(t) = 1 - \sin t; \\ a_{21}(t) &= 2 - \sin t, \quad a_{22}(t) = 1 - \cos t; \\ b_{11}(t) &= 1 + \sin t, \quad b_{12}(t) = 1 - \cos t; \\ b_{21}(t) &= 3 - \sin t, \quad b_{22}(t) = 2 + \cos t; \\ \tau_1(t) &= \tau_2(t) = \frac{1}{2} (|t + 1| - |t - 1|). \end{aligned}$$

then we can easily check that

$$\begin{aligned} k_1(t) &= 5; k_2(t) = 4 \\ \alpha &= \inf_{t \geq t_0} \{k_1(t) - k_2(t)\} = 1 > 0 \end{aligned}$$

Therefore, it follows from Corollary 1 that the system (12) is globally exponentially stable.

*Remark 2.* Since the delay function  $\tau(t)$  in Eq.(12) is not differentiable, the results in [4] and in [22]-[25] can not be applied to this example. Furthermore, due to the unboundedness of  $c_1(t)$  and  $a_{11}(t)$ , the results in [21] are not applicable for this example. Hence, the results here improve and extend those established earlier in [4], [22]-[25] and [21].

## References

1. Roska, T., Chua, L.: Cellular neural networks with delay type template elements and nonuniform grids. *International Journal of Circuit Theory and Application* **20** (1992) 469–481
2. Arik, S.: An improved global stability result for delayed cellular neural networks. *IEEE Trans.Circuits Syst.I* **49** (2002) 1211–1214
3. Arik, S.: An analysis of global asymptotic stability of delayed cellular neural networks. *IEEE Trans.Neural Networks* **13** (2002) 1239–1242

4. Cao, J., Wang, J.: Global asymptotic stability of a general class of recurrent neural networks with time-varying delays. *IEEE Trans.Circuits Syst.I* **50** (2003) 34–44
5. Chen, A., Cao, J., Huang, L.: An estimation of upperbound of delays for global asymptotic stability of delayed Hopfield neural networks. *IEEE Trans.Circuits Syst.I* **49** (2002) 1028–1032
6. Feng, C.H., Plamondon, R.: On the stability analysis of delayed neural networks systems. *Neural Networks* **14** (2001) 1181–1188
7. Huang, H., Cao, J.: On global asymptotic stability of recurrent neural networks with time-varying delays. *Appl.Math.Comput.* **142** (2003) 143–154
8. Liao, X., Chen, G., Sanchez, E.N.: LMI-based approach for asymptotically stability analysis of delayed neural networks. *IEEE Trans.Circuits Syst.I* **49** (2002) 1033–1039
9. Liao, X.X., Wang, J.: Algebraic criteria for global exponential stability of cellular neural networks with multiple time delays. *IEEE Trans.Circuits Syst.I* **50** (2003) 268–274
10. Mohamad, S., Gopalsamy, K.: Exponential stability of continuous-time and discrete-time cellular neural networks with delays. *Appl.Math.Comput.* **135** (2003) 17–38
11. Roska, T., Wu, C.W., Chua, L.O.: Stability of cellular neural network with dominant nonlinear and delay-type templates. *IEEE Trans.Circuits Syst.***40** (1993) 270–272
12. Zeng, Z., Wang, J., Liao, X.: Global exponential stability of a general class of recurrent neural networks with time-varying delays. *IEEE Trans.Circuits Syst.I* **50** (2003) 1353–1358
13. Zhang, J.: Globally exponential stability of neural networks with variable delays. *IEEE Trans.Circuits Syst.I* **50** (2003) 288–290
14. Zhang, Q., Ma, R., Xu, J.: Stability of cellular neural networks with delay. *Electron. Lett.* **37** (2001) 575–576
15. Qiang, Z., Ma, R., Chao, W., Jin, X.: On the global stability of delayed neural networks. *IEEE Trans. Automatic Control* **48** (2003) 794–797
16. Zhang, Q., Wei, X.P. Xu, J.: Global Exponential Convergence Analysis of Delayed Neural Networks with Time-Varying Delays. *Phys.Lett.A* **318** (2003) 537–544
17. Zhang, Q., Wei, X.P. Xu, J.: Global Asymptotic Stability of Hopfield Neural Networks with Transmission Delays. *Phys.Lett.A* **318** (2003) 399–405
18. Zhang, Q., Wei, X.P. Xu, J.: An Analysis on the Global Asymptotic Atability for Neural Networks with Variable Delays. *Phys.Lett.A* **328** (2004) 163–169
19. Zhang, Q., Wei, X.P. Xu, J.: On Global Exponential Stability of Delayed Cellular Neural Networks with Time-Varying Delays. *Appl.Math.Comput.* **162** (2005) 679–686
20. Zhang, Q., Wei, X.P. Xu, J.: Delay-Dependent Exponential Stability of Cellular Neural Networks with Time-Varying Delays. *Chaos, Solitons & Fractals* **23** (2005) 1363–1369
21. Zhou, D., Cao, J.: Globally exponential stability conditions for cellular neural networks with time-varying delays. *Appl.Math.Comput.* **131** (2002) 487–496
22. Jiang, H., Li, Z., Teng, Z.: Boundedness and stability for nonautonomous cellular neural networks with delay. *Phys.Lett.A* **306** (2003) 313–325
23. Jiang, H., Li, Z., Teng, Z.: Boundedness and stability for nonautonomous cellular neural networks with delay. *Phys.Lett.A* **306** (2003) 313–325
24. Liang, J., Cao, J.: Bounded and stability for recurrent neural networks with variable coefficients and time-varying delays. *Phys.Lett.A* **318** (2003) 53–64

25. Rehim, M., Jiang, H., Teng, Z.: Bounded and stability for nonautonomous cellular neural networks with delay. *Neural Networks* **17** (2004) 1017–1025
26. Chua, L.O., Yang, L.: Cellular neural networks: theory and applications. *IEEE Trans. Circuits Syst. I* **35** (1988) 1257–1290
27. Hou, C., Qian, J.: Remarks on quantitative analysis for a family of scalar delay differential inequalities. *IEEE Trans. Automatic Control* **44** (1999) 334–336

# Diversity and Multimodal Search with a Hybrid Two-Population GA: An Application to ANN Development

Juan R. Rabuñal, Julián Dorado, Marcos Gestal, and Nieves Pedreira

Univ. A Coruña, Fac. Informática, Campus Elviña, 15071, A Coruña, Spain  
{juanra, julian, mgestal, nieves}@udc.es

**Abstract.** Being based on the theory of evolution and natural selection, the Genetic Algorithms (GA) represent a technique that has been proved as good enough for the resolution of those problems that require a search through a complex space of possible solutions. The maintenance of a population of possible solutions that are in constant evolution may lead to its diversity being lost, consequently it would be more difficult, not only the achievement of a final solution but also the supply of more than one solution. The method that is described here tries to overcome those difficulties by means of a modification in traditional GA's. Such modification involves the inclusion of an additional population that might avoid the mentioned loss of diversity of classical GA's. This new population would also provide the piece of exhaustive search that allows to provide more than one solution.

**Keywords:** Evolutionary Computation, Genetic Algorithms, Artificial Neural Networks, Homogenisation, Diversity.

## 1 Introduction

Up to day, Evolutionary Computation (EC) techniques [1] have been highly efficient problem solvers through the use of mechanisms that mimic natural evolution [2]. One of these techniques, Genetic Algorithms (GA) [3] has proved to be specially useful when solving problems that require the optimisation of a group of parameters [4]. As a general rule, EC techniques, and more specifically GA, develop a population of solutions that through consecutive generations, they become progressively near of the ideal solution. This evolution is based on the achievement of brand new solutions by means, not only of the recombination of individuals in the population (crossover operation), but also of the random modification of some of the solutions (mutation operation). Nevertheless, this way of working causes that, when population of solutions is excessively homogeneous, GA tends to explore several times the same area of the search space. Diversity loss causes that crossover operations barely explore new regions within the space of solutions. Therefore, only mutation may add new information that allows access to unexplored areas of search space. This fact may represent a significant delay when trying to find the final solution to the problem.

Another inconvenience that is associated to the lack of diversity turns up when trying to solve those problems that have multiple global solutions (multiple global minimums) or several solutions that have a validity slightly lower than the optimal one (unique optimal solution with multiple quasi-global minimums). In such cases, classical GA tendency regarding population would be to keep only values that solely correspond to one of the solutions or to some of them.

## 2 State of the Art

In order to minimise the impact of homogenisation, or at least to tend that it may only affect later states of searching phase, several alternatives have been designed, based most of them on heuristics. Some of the approaches try to solve this problem by means of the dynamic variation of crossover and mutation rates [5]. In that way, when diversity decreases, a higher amount of mutations are done in order to increase the exploration through the search space; when diversity increases, the number of mutations is decreased in order to increase the number of crossings with the aim of improving exploitation in optimal solution search.

Other approaches consist of the use of new crossover or replacement algorithms. Some of the crossover algorithms that improve diversity and that should be highlighted are BLX (Blend Crossover) [6], SBX (Simulated Binary Crossover) [7], PCX (Parent Centric Crossover) [8], CIXL2 Confidence Interval Based Crossover using L2 Norm [9] or UNDX (Unimodal Normally Distributed Crossover) [10]. Regarding replacement algorithms, schemes that may keep population diversity have been also looked for. An example of this type of schemes is Crowding crossover, not only deterministic [11] but also probabilistic [12].

The main inconvenience of the techniques previously described (replacement and crossover operators) lies in the fact that they add new parameters that should be configured according the process of execution of GA. This process may be disturbed by the interactions among those parameters [13].

## 3 Performance of Proposed GA

The GA proposed in this work is based on the addition of a new population (genetic pool) to a traditional GA, the aim of which would be to force an homogeneous search throughout the search space. In order to achieve this, the genetic pool divides the search space into sub-regions. Every one of the individuals of the genetic pool has its own fenced range for gene variation. In this way, every one of these individuals would represent a specific sub-region within the global search space and they will carry out an exhaustive exploration of their respective regions. On the other hand, the group of individual ranges in which any gene may have its value, is extended over the whole of those possible values that a given gene may have. Therefore, the genetic pool would sample the whole of the search space. Depending on both type and complexity of the problem that it is intended to solve, the user has to set the number of sub-regions that require into the search space. It should be borne in mind that a

traditional GA performs its search considering only one sub-region (the whole of the search space). In addition to the genetic pool, it is also used another population (secondary population), where a classical GA would develop its individuals in an interactive fashion with those individuals of the genetic pool. This secondary population would contribute the solutions, whereas the genetic pool would act as a support, keeping search space homogeneously explored.

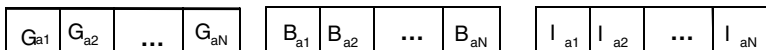
## 4 Description of Populations

### 4.1 Genetic Pool

As it has been previously mentioned, every one of the individuals at the genetic pool represents a sub-region of the global search space. Therefore, they should have the same structure or gene sequence than when using a traditional GA.

The difference lies in the range of values that these genes might have. When offering a solution, traditional GA may have any valid value, whereas in the proposed GA, the range of possible values is restricted. Total value range is divided into the same number of parts than individuals in genetic pool, so that a sub-range of values is allotted to each individual. Those values that a given gene may have will remain within its range for the whole of the performance of the proposed GA.

In addition to all that has been said, every individual at the genetic pool will be in control of which are the genes that correspond to the best found solution up to then (meaning whether they belong to the best individual at secondary population). This Boolean value would be used to avoid the modification of those genes that, in some given phase of performance, are the best solution to the problem.



**Fig. 1.** Format of the individuals of genetic pool

Every one of the genes in an individual has an I value associated which indicates the relative increment that would be applied to the gene during a mutation operation based only on increments and solely applied to individuals of the genetic pool. It is obvious that this incremental value should have to be lower than the maximum range in which gene values may vary. The structure of the individuals at genetic pool is shown at Fig.1. As these individuals do not represent global solutions to the problem that has to be solved, knowing their fitness value will not be compulsory.

### 4.2 Secondary Population

Unlike at genetic pool, the genes of individuals of secondary population may adopt values throughout the whole of the solutions area. In this way, they may offer global solutions to the problem, which is something that none of the individuals at genetic

pool is able to do due to the restriction of the value of their genes to a sub-range. Therefore, not genetic pool but secondary population would provide the user with global solutions to the problem that is wanted to be solved.

The evolution of the individuals at this population will be carried out by means of traditional GA on one condition regarding crossover operation. The two individuals that take part in this operation do not belong to the same population, both of them, genetic pool and secondary population, are represented instead.

## 5 Genetic Operators

As it happens with classical GA, in the method proposed here, the development of solutions is carried out by means of the use of crossover and mutation operators.

### 5.1 Crossover

This operator recombines the genetic material of an individual from secondary population with a representative of the genetic pool. In the proposed method, the former individual is randomly chosen, therefore, there is no need for the secondary population to be organized attending to the fitness value of its individuals, consequently there would be an improvement regarding the required lapse for execution. A representative is also chosen from the genetic pool, so that means that its genetic information is used when building a new individual for every given crossover operation. During the building process, the value for every gene of the representative is randomly and independently chosen among all the individuals (see Fig. 2). The individual that has been generated as previously described, represents, unlike every one of the individuals separately, not a partial but a global solution.

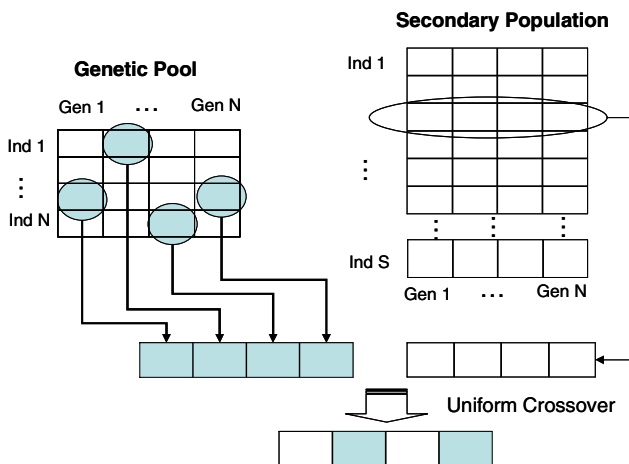


Fig. 2. Crossover operation



After the selection of parents, a uniform crossover is performed aiming for the greatest genetic recombination that may be possible. During the implementation performed and as a result from this crossover, only one individual was generated after the random selection of every one of its genes from both its ancestors. The insertion into the secondary population has been done by means of a non destructive replacement algorithm for ancestors in such way that the insertion is solely performed when offspring level of fitness is higher than the level of the previously selected ancestor at secondary population.

If the application of the crossover operator shows a better solution to the problem than those already stored ones, the genes that were provided by the representative to the genetic pool (Boolean values  $B_{ij}$  at Fig.1) are consequently marked at the pool.

## 5.2 Mutation

The application of the mutation operator involves selecting at random one of the individuals at the genetic pool in order to apply to one of its genes the specified increment that is set by the  $I$  value associated to that gene. The selected gene should not be from the best stored solution at secondary population, meaning that the individual should not have the  $B_i$  value already marked. When the addition of current gene value plus the increment exceeds the upper end of the sub-range in which the gene may vary ( $LIM\_SUP\_IND$ ), it takes on the lesser value at the sub-range ( $LIM\_INF\_IND$ ). Furthermore, it is also reduced ( $\Delta$ ) the incremental value that is going to be applied to subsequent mutations of that gene. In such way, it is intended to do a more exhaustive route through all the values that a given gene may have as generations advance. The mutation operation would be solely applied to the individuals that belong to the genetic pool.

## 6 Description of Performed Tests

At previous works [14][15], it has been proved that the proposed system obtains solutions not only in quite less time but also with a lower number of generations that when using a traditional GA. The proposed algorithm is going to be used at a complex and real case in which GA may be applied to, the artificial neural networks (ANN) training, serving also this test as a comparison with classical GA. To undertake what is intended, ANN is going to be used in order to solve the classification problem of the iris flower.

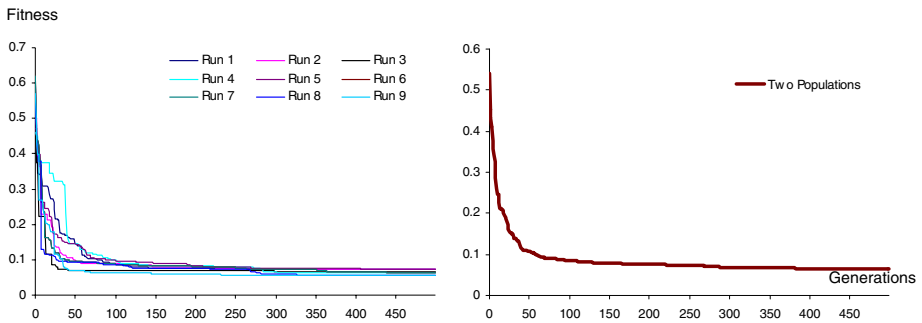
The iris flower data (SAS Institute, 1988) were originally published by Fisher [16] for examples in discriminant analysis and cluster analysis. Four parameters, including sepal and petal length and width, were measured in millimetres on fifty iris specimens from each of three species, *Iris setosa*, *Iris versicolor*, and *Iris virginica*. So, given the four parameters, one should be able to determine which of the three classes a specimen is categorized to. There are 150 data points listed in the database.

We trained different architectures of ANN, and the one that the best results produces are the following: 1 hidden layer with 3 neurons and hyperbolic tangent activation functions in all the neurons.

With this fixed architecture, the training is performed by means of classical GA together with different types of selection algorithms in order to eventually compare the evolution of the training with that one obtained using the algorithm of two populations. In this way, the behaviour of the proposed algorithm may be observed when tackling a real problem that involve some complexity.

## 7 Results

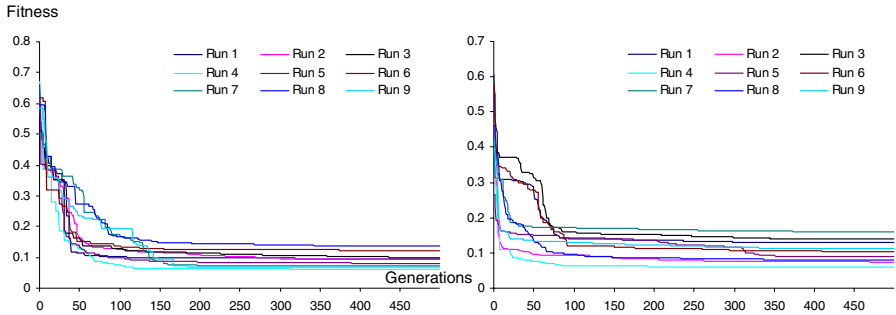
Populations of both algorithms have been allowed to evolve until having 500 generations in order to compare both evolutions. The evolution of the proposed algorithm after 9 independent executions and its average execution may be observed at Fig. 3. Regarding classical GA, they were used the selection algorithms known as roulette and tournament. The results are shown at Fig.4.



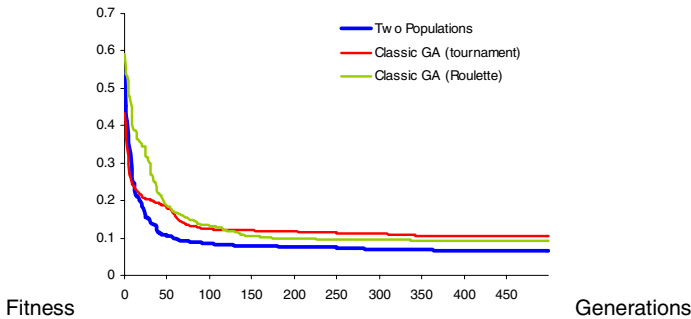
**Fig. 3.** Evolution of the fitness obtained by means of two population GA and average fitness after nine executions

When trying out classical GA, 1000 individuals of the population were used and the parameters that achieve better results use 65% crossover, 25% mutation and 10% copy rates. Two-population GA was tested with the same number of individuals from secondary population (for better comparison with classical GA) and with 20 individuals from genetic pool. In the later algorithm, the parameters that achieve better results use 40% crossover, 15% mutation and 45% copy rates.

The evolution of 500 generations of both algorithms (average values after 9 executions) may be observed at Fig.5, where it is obvious that two-population GA has a greater convergence speed than classical one. It can be also noticed that the time required to obtain 500 generations by two-population GA is 1/3 less of that required by classical GA, being this fact mainly due to its lower genetic operator rates for each generation.



**Fig. 4.** Evolution of the fitness obtained by means of classical algorithm using the roulette selection algorithm (left) and tournament selection algorithm (right)



**Fig. 5.** Comparison between two-population and classical GA (roulette and tournament)

## 8 Conclusions

Given the results of tests performed, it could be stated that the proposed GA fulfils what was suggested regarding systematic exploration through the space of solutions. The use of a genetic pool leads the system to divide the problem into sub-regions in which, every individual will focus its search for solutions, thus, avoiding not only that the search might be focused within only one region, but also that the population might undergo its homogenisation.

It could be confirmed as well that the proposed system achieve the solution using a lower number of interactions than a classical GA, precisely due to the maintenance of a higher diversity within the population. It is also worthy of mention the fact that the proposed GA only requires one additional parameter being adjusted from classical GA: the number of individuals at the genetic pool. Future works

One of the working lines that is intended to follow would try to achieve the automatism in the treatment of the regions that were defined by the genetic pool. Depending on the track that the search might follow, in some occasions it will be

interesting to define a higher number of sub-regions than in some others. In order to define this adjustment dynamically, it would be necessary to establish some metrics that might determine when and in which way the structure of the genetic pool should be modified. It is also intended to check the level of efficiency of a hybrid solution that consist on both, a search method based on populations and the GA that was proposed here. In such case, the fist method would be executed until the moment that the genetic population reached a critic diversity threshold and then, the two-population GA would start its execution until reaching an acceptable diversity level. Another research line would be focused on the distribution of the proposed GA. The development of the algorithm might allow the exchange regarding, not only individuals among several distributed populations, but also sub-regions for search.

## Acknowledgements

This work was supported in part by the Spanish Department of Science and Technology (MCYT) (TIC2003-07593), by the Dirección Xeral de Investigación - Xunta de Galicia (Ref PGIDT04PXIC10503PN and PGIDT04PXIC10504PN) and by the Instituto de la Salud Carlos III (Red n°: G03/160).

## References

1. T. Bäck, *Evolutionary Algorithms in Theory and Practice*, Oxford University Press, 1996.
2. C. Darwin, *The Origin of Species by Means of Natural Selection*, 1859.
3. J. Holland, *Adaptation in Natural and Artificial Systems*, University of Michingan Press, Ann Arbor, 1975.
4. A. D. Bethke, *Genetic Algorithms and Function Optimizers*, Tesis Doctoral, University of Michigan, 1981.
5. R. K. Ursem, "Diversity-Guided Evolutionary Algorithms", Proceedings of VII Parallel Problem Solving from Nature, pp. 462-471. 2002.
6. L. J. Eshelman, J.D. Schaffer, "Real coded genetic algorithms and interval schemata", Foundations of Genetic Algorihtms (2), pp. 187-202, 1993.
7. K. Deb, S. Agrawal, "Simulated binary crossover for continuous search space", Complex Systems 9(2), pp. 115-148. 1995.
8. K. Deb, A. Anand, D. Joshi, "A computationally Efficient Evolutionary Algorithm for Real Parameter Optimization", KanGAL report: 2002003, 2002.
9. D. Ortiz, C. Hervás, N. García. "CIXL2: A crossover operator for evolutionary algorithms based on population features". Journal of Artificial Intelligence Research, 2005.
10. I. Ono, S. Kobayashi. "A real-coded genetic algorithm for function optimization using unimodal normal distribution", Proceedings of International Conference on Genetic Algorithms, pp. 246-253. 1999.
11. K. A. DeJong, *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*. Tesis Doctoral, University of Michigan, Ann Arbor. 1975.
12. O.J. Mengshoel, D.E. Goldberg, "Probabilistic Crowding: Deterministic Crowding with Probabilistic Replacement", Proceedings of Genetic and Evolutionary Computation, pp. 409-416. 1999.

13. P.J. Ballester, J.N. Carter, "Real-Parameter Genetic Algorithm for Finding Multiple Optimal Solutions in Multimodel Optimizaton", *Proceedings of Genetic and Evolutionary Computation*, pp. 706-717. 2003.
14. J. Dorado, A. Santos, J.R. Rabuñal, N. Pedreira, A. Pazos, "Hybrid Two-Population Genetic Algorithm", *Proceedings of 7th Fuzzy Days in Dortmund*, pp. 464-470. 2001.
15. J. Dorado, J.R. Rabuñal, A. Pazos, A. Santos, N. Pedreira, "Algoritmo genético de dos poblaciones para búsqueda homogénea en el espacio de soluciones", *Actas de Algoritmos Evolutivos y Bioinspirados*, pp. 241-245. 2002.
16. Fisher, R.A. The use of multiple measurements in taxonomic problems, *Annals of Eugenic*. pp.179-188. 1936.

# Identification of Fuzzy Systems with the Aid of Genetic Fuzzy Granulation

Sung-Kwun Oh<sup>1</sup>, Keon-Jun Park<sup>1</sup>, Yong-Soo Kim<sup>2</sup>, and Tae-Chon Ahn<sup>3</sup>

<sup>1</sup> Department of Electrical Engineering, The University of Suwon, San 2-2 Wau-ri, Bongdam-eup, Hwaseong-si, Gyeonggi-do, 445-743, South Korea  
ohsk@suwon.ac.kr

<sup>2</sup> Division of Computer Engineering, Daejeon University, 96-3 Yongun-dong, Dong-gu, Daejeon, 300-716, South Korea

<sup>3</sup> Department of Electrical Electronic and Information Engineering, Wonkwang University, 344-2, Shinyong-Dong, Iksan, Chon-Buk, 570-749, South Korea

**Abstract.** We propose the identification of fuzzy systems with the aid of genetic fuzzy granulation to carry out the model identification of complex and nonlinear systems. The proposed fuzzy model implements system structure and parameter identification with the aid of genetic algorithms and information granulation. To identify the structure of fuzzy rules we use genetic algorithms. Granulation of information realized with Hard C-Means clustering help determine the initial parameters of fuzzy model such as the initial apexes of the membership functions and the initial values of polynomial functions being used in the premise and consequence part of the fuzzy rules. And the initial parameters are tuned effectively with the aid of the genetic algorithms and the least square method. An example is given to evaluate the validity of the proposed model.

## 1 Introduction

Fuzzy modeling has been a focal point of the technology of fuzzy sets from its very inception. Fuzzy modeling has been studied to deal with complex, ill-defined, and uncertain systems in many other avenues. In the early 1980s, linguistic modeling [2], [4] and fuzzy relation equation-based approach [5], [6] were proposed as primordial identification methods for fuzzy models. The general class of Sugeno-Takagi models [8] gave rise to more sophisticated rule-based systems where the rules come with conclusions forming local regression models. While appealing with respect to the basic topology (a modular fuzzy model composed of a series of rules) [9], [10], these models still await formal solutions as far as the structure optimization of the model is concerned, say a construction of the underlying fuzzy sets - information granules being viewed as basic building blocks of any fuzzy model. Some enhancements to the model have been proposed by Oh and Pedrycz [11], [12], yet the problem of finding “good” initial parameters of the fuzzy sets in the rules remains open.

This study concentrates on the central problem of fuzzy modeling that is a development of information granules-fuzzy sets. Taking into consideration the essence of the granulation process, we propose to cast the problem in the setting of clustering

techniques and genetic algorithms. The design methodology emerges as a hybrid structural optimization and parametric optimization. Information granulation with the aid of HCM clustering help determine the initial parameters of fuzzy model such as the initial apexes of the membership functions and the initial values of polynomial function being used in the premise and consequence part of the fuzzy rules. And the initial parameters are tuned (adjusted) effectively with the aid of the genetic algorithms and the least square method. The proposed model is evaluated using a numerical example.

## 2 Information Granulation (IG)

Information granules [13], [14] are informally viewed as linked collections of objects (data point, in particular) drawn together by the criteria of proximity, similarity, or functionality. Granulation of information is an inherent and omnipresent activity of human beings carried out with intent of better understanding of the problem. In particular, granulation of information is aimed at splitting the problem into several manageable chunks. In this way, we partition this problem into a series of well-defined subproblems (modules) of a far lower computational complexity than the original one. The form of information granulation (IG) themselves becomes an important design feature of the fuzzy model, which are geared toward capturing relationships between information granules.

It is worth emphasizing that the HCM clustering has been used extensively not only to organize and categorize data, but it becomes useful in data compression and model identification. For the sake of completeness of the entire discussion, let us briefly recall the essence of the HCM algorithm [15].

We obtain the matrix representation for hard  $c$ -partition, defined as follows.

**[Step 1].** Fix the number of clusters  $c$  ( $2 \leq c < m$ ) and initialize the partition matrix.

**[Step 2].** Calculate the center vectors  $\mathbf{v}_g$  of each cluster.

**[Step 3].** Update the partition matrix  $\mathbf{U}^{(r)}$ ; these modifications are based on the standard Euclidean distance function between the data points and the prototypes.

**[Step 4].** Check a termination criterion.

## 3 Fuzzy Inference Systems (FIS) with the Aid of Information Granulation

The identification procedure for fuzzy models is usually split into the identification activities dealing with the premise and consequence parts of the rules. The identification completed at the premise level consists of two main steps. First, we select the input variables  $x_1, x_2, \dots, x_k$  of the rules. Second, we form fuzzy partitions of the spaces over which these individual variables are defined. The identification of the consequence part of the rules embraces two phases, namely 1) a selection of the consequence variables of the fuzzy rules, and 2) determination of the parameters of the consequence (conclusion part). And the least square error (LSE) method used at the parametric optimization of the consequence parts of the successive rules.

In this study, we carry out the modeling using characteristics of input-output data set. Therefore, it is important to characteristics of data. To find this we use HCM clustering. By classifying data as characteristics through HCM clustering, we design the fuzzy model by means of center of classified clusters. Information granulation with the aid of HCM clustering help determine the initial parameters of fuzzy model such as the initial apexes of the membership functions and the initial values of polynomial function being used in the premise and consequence part of the fuzzy rules.

### 3.1 Premise Identification

In the premise part of the rules, we confine ourselves to a triangular type of membership function. For the triangular membership functions we have parameters to optimize. The HCM clustering helps us organize the data into cluster, and in this way we take into account the characteristics of the experimental data. In the regions where some clusters of data have occurred, we end up with some fuzzy sets that help represent the specificity of the data set. In the sequel, the modal values of the clusters are refined (optimized) using genetic optimization (GAs).

The premise identification according to information granulation with the aid of HCM clustering is as follows.

Suppose that we are given a set of data  $U = \{x_1, x_2, \dots, x_l; y\}$ , where  $x_k = [x_{1k}, \dots, x_{mk}]^T$ ,  $y = [y_1, \dots, y_m]^T$ ,  $l$  is the number of variables and  $m$  is the number of data.

**[Step 1].** Arrange a set of data  $U$  into data set  $X_k$  composed of respective input data and output data.

$$X_k = [x_k; y] \tag{1}$$

$X_k$  is data set of  $k$ -th input data and output data, where,  $x_k = [x_{1k}, \dots, x_{mk}]^T$ ,  $y = [y_1, \dots, y_m]^T$ , and  $k=1, 2, \dots, l$ .

**[Step 2].** Perform the HCM clustering to find the center vector  $v_{kg}$  with data set  $X_k$ .

**[Step 2-1].** Classify data set  $X_k$  into  $c$ -clusters, which granulate the information.

**[Step 2-2].** Calculate the center vectors  $v_{kg}$  of each cluster.

$$v_{kg} = \{v_{k1}, v_{k2}, \dots, v_{kc}\} \tag{2}$$

Where,  $k=1, 2, \dots, l$ ,  $g = 1, 2, \dots, c$ .

**[Step 3].** Partition the corresponding input space by the center vectors  $v_{kg}$  and assign a comprehensive meaning to each cluster such as Small, Big.

**[Step 4].** Set the initial apexes of the membership functions with the center vectors  $v_{kg}$ .

### 3.2 Consequence Identification

The characteristics of input-output data is also involved in the conclusion parts as follows:

**[Step 1].** Find a set of data included in the fuzzy space of the  $j$ -th rule.

**[Step 1-1].** Find the input data included in each cluster (information granule) from the partition matrix  $u_{gi}$ ,  $g = 1, 2, \dots, c$ ,  $i=1, 2, \dots, m$  of each input variable.

**[Step 1-2].** Find the input data pairs included in the fuzzy space of the  $j$ -th rule.

**[Step 1-3].** Find the corresponding output data from above input data pairs.



**[Step 2].** Compute the center vectors  $\mathbf{V}_j$  of the data set by the arithmetic mean in each rule.

$$\mathbf{V}_j = \{V_{1j}, V_{2j}, \dots, V_{kj}, M_j\} \quad (3)$$

where,  $k=1, 2, \dots, l$ ,  $j=1, 2, \dots, n$ .  $V_{kj}$  and  $M_j$  are center values of input and output data, respectively.

**[Step 3].** Set the initial values of polynomial functions with the center vectors  $\mathbf{V}_j$ .

## 4 Optimization of IG-Based FIS

The need to solve optimization problems arises in many fields and is especially dominant in the engineering environment. There are several analytic and numerical optimization techniques, but there are still large classes of functions that are fully addressed by these techniques. Especially, the standard gradient-based optimization techniques that are being used mostly at the present time are augmented by a differential method of solving search problems for optimization processes. Therefore, the optimization of fuzzy models may not be fully supported by the standard gradient-based optimization techniques, because of the nonlinearity of fuzzy models represented by rules based on linguistic levels. This forces us to explore other optimization techniques such as genetic algorithms. First of all, to identify the fuzzy model we determine such an initial structure as the number of input variables, input variables being selected and the number of membership functions in premise part and the order of polynomial (Type) in conclusion. And then the membership parameters of the premise are optimally tuned by GAs. In what follows, we briefly review the underlying ideas of GAs, and then discuss a form of the performance index used in this identification problem.

Genetic algorithms [16] have proven to be useful in optimization of such problems because of their ability to efficiently use historical information to obtain new solutions with enhanced performance and a global nature of search supported there.

In this study, for the optimization of the fuzzy model, genetic algorithms use the serial method of binary type, roulette-wheel in the selection operator, one-point crossover in the crossover operator, and invert in the mutation operator. Here, we use 1000 generations, 60 populations, 10 bits per string, crossover rate equal to 0.6, and mutation probability equal to 0.1 (the choice of these specific values of the parameters is a result of intensive experimentation).

## 5 Experimental Studies

We illustrate the performance of the model and elaborate on its development by experimenting with data coming from the gas furnace process. The time series data (296 input-output pairs) resulting from the gas furnace process has been intensively studied in the previous literatures [11], [12], [17], [18]. The delayed terms of methane gas flow rate  $u(t)$  and carbon dioxide density  $y(t)$  are used as six input variables with vector formats such as  $[u(t-3), u(t-2), u(t-1), y(t-3), y(t-2), y(t-1)]$ . And as output variable  $y(t)$  is used. The first one (consisting of 148 pairs) was used for training. The remain-

ing part of the series serves as a testing set. We consider the MSE being regarded here as a performance index.

We carried out the structure identification on a basis of the experimental data using GAs to design Max\_Min-based and IG-based fuzzy model. The maximal number of input variables was set to be equal to 2 from above the type of vector format. The corresponding input variables were picked up to be  $y(t-2)$ ,  $y(t-1)$  for Max\_Min-based fuzzy and  $u(t-3)$ ,  $y(t-1)$  for IG-based fuzzy model. To evaluate the proposed model we designed the Max\_Min-based and IG-based fuzzy model for each model. In case of input variables of  $y(t-2)$ ,  $y(t-1)$ , the number of membership functions assigned to each input was set up to be 3, 2 and the other was set up to be 3 for each input. At the conclusion part, each system comes with the consequence type 2 and 4, respectively. For each fuzzy model, we conducted the optimization of the parameters of the premise membership functions.

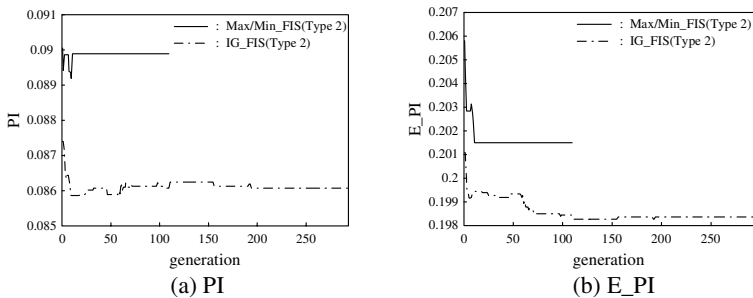
Table 1 summarizes the performance index for Max\_Min-based and IG-based fuzzy model.

**Table 1.** Performance index of Max\_Min-based and IG-based fuzzy model ( $\theta=0.5$ )

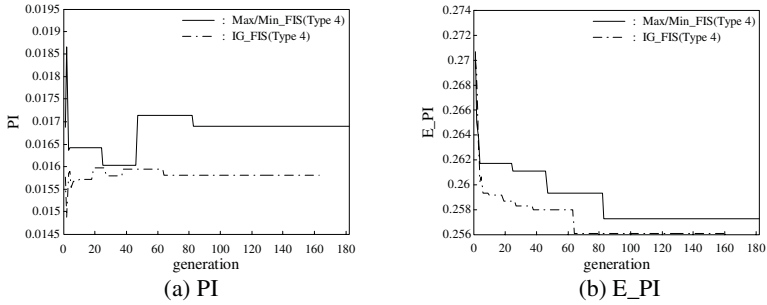
| Model       | Identification | Input variable | No. Of MFs | Type | PI    | E_PI  |
|-------------|----------------|----------------|------------|------|-------|-------|
| Max/Min_FIS | Structure*     |                |            |      | 0.092 | 0.215 |
|             | Parameters*    | $y(t-2)$       | 3x2        | Type | 0.089 | 0.201 |
| IG_FIS      | Structure      | $y(t-1)$       |            | 2    |       | 0.085 |
|             | Parameters     |                |            |      | 0.086 | 0.198 |
| Max/Min_FIS | Structure      |                |            |      | 0.018 | 0.315 |
|             | Parameters     | $u(t-3)$       | 3x3        | Type | 0.016 | 0.257 |
| IG_FIS      | Structure*     | $y(t-1)$       |            | 4    |       | 0.015 |
|             | Parameters*    |                |            |      | 0.015 | 0.256 |

\* identified structure using GAs

From the table 1 it is clear that the performance of a IG-based fuzzy model is better than that of a Max\_Min-based fuzzy model not only after identifying the structure but also after identifying optimally the parameters.



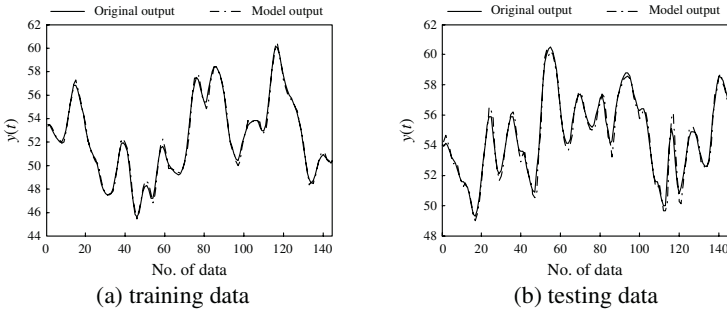
**Fig. 1.** Optimal convergence process of performance index for Max\_Min-based and IG-based fuzzy model ( $y(t-2)$ ,  $y(t-1)$ )



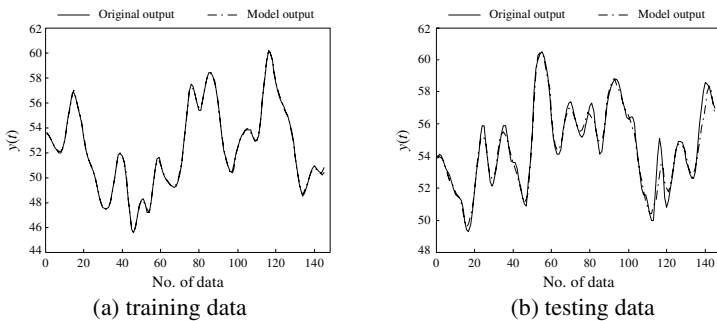
**Fig. 2.** Optimal convergence process of performance index for Max\_Min-based and IG-based fuzzy model ( $u(t-3)$ ,  $y(t-1)$ )

Figure 1 and 2 depict the values of the performance index produced in successive generation of the GAs. It is obvious that the performance of an IG-based fuzzy model is good from initial generation due to the characteristics of input-output data. Model output and predicting error of training and testing data for IG-based fuzzy model is presented in figure 3 and 4.

The identification error (performance index) of the proposed model is also compared to the performance of some other models in table 2.



**Fig. 3.** The comparison of original and output data for IG-based fuzzy model ( $y(t-2)$ ,  $y(t-1)$ )



**Fig. 4.** The comparison of original and output data for IG-based fuzzy model ( $u(t-3)$ ,  $y(t-1)$ )

**Table 2.** Comparison of identification error with previous models

| Model                          | $PI_t$           | PI    | E_PI  | No. of rules |
|--------------------------------|------------------|-------|-------|--------------|
| Tong's model[3]                | 0.469            |       |       | 19           |
| Pedrycz's model[4]             | 0.776            |       |       | 20           |
| Xu's model[7]                  | 0.328            |       |       | 25           |
| Sugeno's model[9]              | 0.355            |       |       | 6            |
| Oh et al.'s<br>model[11], [12] | Simplified       | 0.024 | 0.328 | 4            |
|                                | Linear           | 0.022 | 0.326 | 4            |
|                                |                  | 0.021 | 0.364 | 6            |
| HCM+GA<br>[18]                 | Simplified       | 0.035 | 0.289 | 4            |
|                                | ( $\theta=0.5$ ) | 0.022 | 0.333 | 6            |
|                                | Linear           | 0.026 | 0.272 | 4            |
| ( $\theta=0.5$ )               |                  | 0.020 | 0.264 | 6            |
| Our model<br>( $\theta=0.5$ )  | $y(t-2), y(t-1)$ | 0.086 | 0.198 | 6            |
|                                | $u(t-3), y(t-1)$ | 0.015 | 0.256 | 9            |

## 6 Conclusions

In this paper, we have developed a comprehensive identification framework for fuzzy model with the aid of information granulation. The underlying idea deals with an optimization of information granules by exploiting techniques of clustering and genetic algorithms. The experimental studies showed that the model is compact (realized through a small number of rules), and their performance is superb in comparison to other models. The proposed model is effective for nonlinear complex systems, so we can construct a well-organized model.

While the detailed discussion was focused on triangular fuzzy sets, the developed methodology applies equally well to any other class of fuzzy sets as well as a type of nonlinear local model. Moreover, the models scale up quite easily and do not suffer from the curse of dimensionality encountered in other identification techniques of rule-based systems.

## Acknowledgement

This work has been supported by KESRI(R-2004-B-133-01), which is funded by MOCIE(Ministry of commerce, industry and energy).

## References

1. Zadeh, L.A.: Fuzzy sets. *Information and Control*. **8** (1965) 338-353
2. Tong, R.M.: Synthesis of fuzzy models for industrial processes. *Int. J Gen Syst*. **4** (1978) 143-162
3. Tong, R.M.: The evaluation of fuzzy models derived from experimental data. *Fuzzy Sets Syst*. **13** (1980) 1-12
4. Pedrycz, W.: An identification algorithm in fuzzy relational system. *Fuzzy Sets Syst*. **13** (1984) 153-167

5. Pedrycz, W.: Numerical and application aspects of fuzzy relational equations. *Fuzzy Sets Syst.* **11** (1983) 1-18
6. Czogola, E., Pedrycz, W.: On identification in fuzzy systems and its applications in control problems. *Fuzzy Sets Syst.* **6** (1981) 73-83
7. Xu, C. W., Zailu, Y.: Fuzzy model identification self-learning for dynamic system. *IEEE Trans. on Syst. Man, Cybern. SMC-17*(4) (1987) 683-689
8. Takagi, T., Sugeno, M.: Fuzzy identification of systems and its applications to modeling and control. *IEEE Trans Syst, Cybern. SMC-15*(1) (1985) 116-132
9. Sugeno, M., Yasukawa, T.: Linguistic modeling based on numerical data. In: *IFSA'91 Brussels, Computer, Management & System Science.* (1991) 264-267
10. Ismail, M.A.: Soft Clustering Algorithm and Validity of Solutions. In: Gupta MM, editor. *Fuzzy Computing Theory, Hardware and Application.* Amsterdam. North Holland. (1988) 445-471
11. Oh, S.K., Pedrycz, W.: Identification of fuzzy systems by means of an auto-tuning algorithm and its application to nonlinear systems. *Fuzzy Sets and Syst.* **115**(2) (2000) 205-230
12. Park, C. S., Oh, S. K., Pedrycz, W.: Fuzzy Identification by means of Auto-Tuning Algorithm and Weighting Factor. *The Third Asian Fuzzy Systems Symposium(AFSS).* (1998) 701-706
13. Zadeh, L.A.: Toward a theory of fuzzy information granulation and its centrality in human reasoning and fuzzy logic. *Fuzzy Sets and Syst.* **90** (1997) 111-117
14. Pedrycz, W., Vukovich, G.: Granular neural networks. *Neurocomputing.* **36** (2001) 205-224
15. Krishnaiah, P.R., Kanal, L.N., editors.: Classification, pattern recognition, and reduction of dimensionality, volume 2 of *Handbook of Statistics.* North-Holland Amsterdam (1982)
16. Goldberg, D.E.: *Genetic Algorithm in search, Optimization & Machine Learning,* Addison Wesley (1989)
17. Box, G. E. P., Jenkins, G. M.: *Time Series Analysis: Forecasting and Control,* 2nd ed. Holden-Day. San Francisco CA (1976)
18. Park, B. J., Pedrycz, W., Oh, S.-K.: Identification of Fuzzy Models with the Aid of Evolutionary Data Granulation. *IEE Proc.-Control Theory and Applications.* 148(05) (2001) 406-418
19. Park, H.S., Oh, S.K.: Rule-based Fuzzy-Neural Networks Using the Identification Algorithm of the GA Hybrid Scheme. *International journal of Control, Automations, and Systems.* **1**(1) (2003) 101-110
20. Oh, S.K., Rho, S.B., Kim, H.K.: Fuzzy Controller Design by Means of Genetic Optimization and NFN-Based Estimation Technique. *International journal of Control, Automations, and Systems.* **2**(3) (2004) 362-373

# Clustering-Based TSK Neuro-fuzzy Model for Function Approximation with Interpretable Sub-models

Luis Javier Herrera, Héctor Pomares, Ignacio Rojas, Alberto Guilén,  
Jesús González, and Mohammed Awad

University of Granada, Department of Computer Architecture and Technology,  
E.T.S. Computer Engineering, 18071 Granada, Spain  
<http://atc.ugr.es>

**Abstract.** TSK models are a very powerful tool for function approximation problems given a dataset of input/output data. Given a global error function to approximate, there are several methodologies for training (adjust the parameters and find the optimal structure) the TSK model. Nevertheless, this strategy implies that the interpretability of the rules comprising the neuro-fuzzy TSK system as linearizations of the nonlinear subjacent system can be lost. Several recent works have addressed this problem with partial success, sometimes performing a tradeoff between global accuracy and local models interpretability. In this paper we propose an accurate modified TSK neuro-fuzzy model for function approximation that solves the cited problem, and that furthermore allows us to interpret the output of the rules as the Taylor Series Expansion of the system output around the rule centres.

## 1 Introduction

Typically, the structure of a multiple-input single-output (MISO) Takagi-Sugeno-Kang (TSK) [1] fuzzy system and its associated fuzzy inference method comprises a set of  $K$  IF-THEN rules in the form

$$Rule^k : \text{IF } x_1 \text{ is } \mu_1^k \text{ AND } \dots \text{ AND } x_n \text{ is } \mu_n^k \text{ THEN } y = R^k \quad (1)$$

where the  $\mu_i^k$  are fuzzy sets characterized by membership functions (MF)  $\mu_i^k(x_i)$  in universes of discourse  $U_i$  (in which variables  $x_i$  take their values), and  $R_k$  are the consequents of the rules.

The output of a fuzzy system with rules in the form shown in Eq. 1 can be expressed (using weighted average aggregation) as

$$F(\mathbf{x}) = \frac{\sum_{k=1}^K \mu^k(\mathbf{x}) R^k}{\sum_{k=1}^K \mu^k(\mathbf{x})} \quad (2)$$

provided that  $\mu^k(\mathbf{x})$  is the activation value for the antecedent of the rule  $k$ , which can be expressed as

$$\mu^k(\mathbf{x}) = \mu_1^k(x_1)\mu_2^k(x_2)\dots\mu_n^k(x_n). \quad (3)$$

As we have mentioned before, TSK fuzzy systems are often used to deal with function approximation problems due to its ability to explain non-linear relations using a relatively low number of simple rules. We must recall that the problem of function approximation is about to estimate an unknown function  $f$  from samples of the form  $\{\mathbf{x}^m; z^m\}; m = 1, 2, \dots, M$ ; with  $z^m = f(\mathbf{x}^m) \in \mathbb{R}$ , and  $\mathbf{x}^m \in \mathbb{R}^m$  and is a crucial problem for a number of scientific and engineering areas. The main goal is thus to learn an unknown functional mapping between the input vectors and their corresponding continuous output values, using a set of known training samples. Later, this generated mapping will be used to obtain the expected output given any new input data.

Specifically, for function approximation problems, two main techniques appear in the literature using neuro-fuzzy TSK systems: Grid-Based Fuzzy Systems (GBFSs) [8] and *Clustering-Based Fuzzy Systems* (CBFSs) [9] (though there are also several hybrid and modified techniques), whose main difference is the type of partitioning of the input space. GBFSs have the advantage that they perform a thorough coverage of the input space, but at the expense of suffering from the curse of dimensionality problem that makes them inapplicable in cases of moderate complexity. In contrast, Clustering-Based Fuzzy System (CBFSs) techniques locate the rules in the zones of the input space in which they are needed, being more suitable thus, for example, for Time Series Prediction problems in which the input data is more centralized in some regions of the input space, or for problems with moderate complexity and a higher number of input variables.

In this paper we deal with CBFSs, though the method proposed is also applicable to GBFSs. In both cases, as we have said, the interpretability of the local sub-models is a must [2, 3]. Some methodologies are centred in the interpretability of the local sub-models [4], but these systems barely represent a good approximation to the I/O dataset. On the other side, most of the neuro-fuzzy approaches (including RBF networks [7]) are centred on reducing the global error function

$$J = \sum_{m \in D} (f(\mathbf{x}^m) - z^m)^2. \quad (4)$$

In relation to GBFSs, there are some recent approaches [3, 6] that solve both the accuracy and local interpretability problems with, in principle, no need of trade-off between both objectives. In this paper we present a novel modified TSK neuro-fuzzy model that will allow us to use learning algorithms that minimize  $J$  (see *Eq. 4*), while providing full interpretability to the local models, in the line of the TaSe model [6], using the Taylor Series Expansion concept.

## 2 The TaSe-C Model

In this section we introduce our modified TSK model for function approximation, which we will call the TaSe-C model (Taylor Series-Based TSK model for CBFSS) that allows the interpretability of the local models. First we will review the characteristics that a TSK fuzzy model has to comply in order to provide this interpretability.

### 2.1 Requirements for the Interpretability of the TSK Rules

First we will recall the Taylor Theorem that states that if a function  $f(\mathbf{x})$  defined in an interval has derivatives of all orders, it can be approximated near a point  $\mathbf{x} = \mathbf{a}$ , as its Taylor Series Expansion around that point:

$$f(\mathbf{x}) = f(\mathbf{a}) + (\mathbf{x} - \mathbf{a})^T \left[ \frac{\partial f}{\partial \mathbf{x}_i}(\mathbf{a}) \right]_{i=1 \dots n} + \frac{1}{2}(\mathbf{x} - \mathbf{a})^T W(\mathbf{x} - \mathbf{a}) + \dots + \frac{1}{(n+1)!} f^{(n+1)}(\mathbf{c})(\mathbf{x} - \mathbf{a})^{n+1} \quad (5)$$

where in each case,  $\mathbf{c}$  is a point between  $\mathbf{x}$  and  $\mathbf{a}$ , and where  $W$  is a triangular matrix of dimensions  $n \times n$  having the respective second order partial derivatives.

The Taylor Theorem opens a door for the approximation of any function through polynomials, that is, through the addition of a number of simple functions. It is therefore a fundamental key in the field of Function Approximation Theory and Mathematical Analysis. Thus, we can use rule consequents  $R^k$  (in Eq. 1) that have the truncated form of Eq. 5. Having a system with a fixed MF configuration, the rule consequents coefficients can be optimally calculated using the Least Squares approach (from Eq. 4) as explained in [5].

But in order to obtain a model in which each rule consequent  $R^k$  in Eq. 1 is the Taylor Series Expansion of the model output around each rule centre, we will require the model output to be continuous and differentiable (at least as many times as the order of the polynomial rule consequent  $R^k$ , so that the Taylor Theorem could be applied properly) and the activation of all the rules in each rule centre  $\mathbf{c}^k$  to be 0 but for the respective rule  $k$ . That is, the activation function of every rule must be continuous and differentiable and must vanish as it approaches to every other rule centre.

For GBFSS, Bikhdash in [4] presented a modified membership function type that complied with such interpretability, and in [5] and [6], this approach was expanded to perform function approximation from a I/O data set and a complete GBFS learning algorithm. Nevertheless, for CBFSS this problem is trickier. It's not straightforward to obtain a fuzzy clustering partition that covers the whole input space and that accomplishes the previous requirements, neither exists a Membership Function type that could provide such interpretability in CBFSS.

**Aggregation Operator.** As we have mentioned, for CBFSS it is impossible to have MFs that comply with such desired properties. The non-grid input space



partition organization avoids this possibility. Suppose the simplest case in which we have a one-dimensional input space with domain [0,1] and two gaussian-type MFs (thus two rules) centred in  $c^1 = 0.3$  and  $c^2 = 0.7$  with  $\sigma = 0.3$ . In this case there is a strong overlap between both MFs in both rule centres. To avoid this overlap we will allow the domain of the first MF  $\mu^1(x)$  to be limited by the function  $1 - \mu^2(x)$ , i.e., when the activation value of the other rule is 1, the activation value of the first rule will be forced to be 0. More specifically the activation value for the first rule  $\mu^1(x)$  will be limited by

$$\mu^2(x) = \begin{cases} 1 - \mu^2(x) & \text{if } x < c^2 \\ 0 & \text{if } x \geq c^2 \end{cases} \tag{6}$$

and on the other hand, the activation value for the first rule  $\mu^1(x)$  will be limited by

$$\mu^1(x) = \begin{cases} 1 - \mu^1(x) & \text{if } x > c^1 \\ 0 & \text{if } x \leq c^1 \end{cases} \tag{7}$$

that is to say, the final activation value of any point for the each rule will be

$$\mu^{k*}(x) = \mu^k(x) \left( 1 - \prod_{\substack{j=1 \\ j \neq k}}^{j=n} \mu'^j(x) \right). \tag{8}$$

Thus, generalizing to the n-dimensional case with any number of rules  $K$ , the general expression for the output of the TSK system, using weighted average (to force each activation value of each rule at the same rule centre to be 1) can be calculated as

$$F(\mathbf{x}) = \frac{\sum_{k=1}^K \mu^{k*}(\mathbf{x}) R^k}{\sum_{k=1}^K \mu^{k*}(\mathbf{x})} = \frac{\sum_{k=1}^K \left( \mu^k(\mathbf{x}) \left( 1 - \prod_{\substack{j=1 \\ j \neq k}}^{j=n} \mu'^j(\mathbf{x}) \right) \right) R^k}{\sum_{k=1}^K \left( \mu^k(\mathbf{x}) \left( 1 - \prod_{\substack{j=1 \\ j \neq k}}^{j=n} \mu'^j(\mathbf{x}) \right) \right)} \tag{9}$$

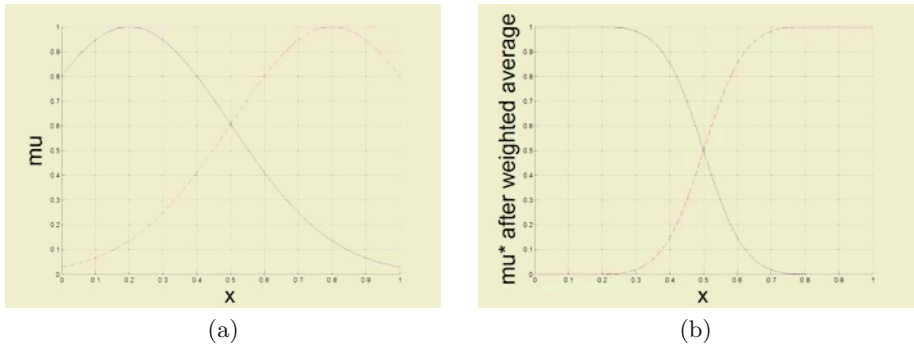
where the  $\mu'^j(\mathbf{x})$  has the form

$$\mu'^j(\mathbf{x}) = \mu_1'^j(x_1) \mu_2'^j(x_2) \dots \mu_n'^j(x_n) \tag{10}$$

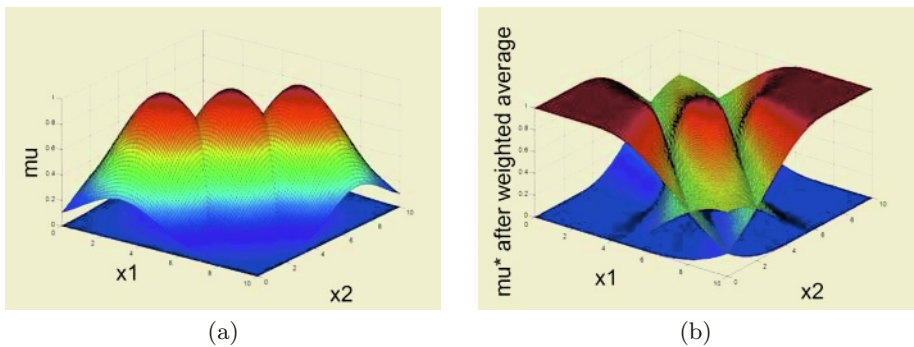
in which each  $\mu_i'^j(x_i)$  has the form shown in Eq. 6 or Eq. 7 depending on the relative position of the centres  $c_i^j$  and  $c_i^k$ .

This new formulation of the system output in Eq. 9 is simply a modified aggregation operator with weighted averaging behaviour. Fig. 1 shows the uni-dimensional toy example with the original MFs  $\mu^1(x)$  and  $\mu^2(x)$  and the final MFs  $\mu^{1*}(x)$  and  $\mu^{2*}(x)$ .

For higher dimensional cases, Fig. 2 shows an example of a two dimensional case with domain [0,10] with 3 gaussian-type MFs having centres in [3,3], [5,5]



**Fig. 1.** a) Original MFs for a one-dimensional example. b) Activations using the modified Aggregation Operator



**Fig. 2.** a) Original MFs for a two-dimensional example. b) Activations using the modified Aggregation Operator

and [7,7] with  $\sigma = 2$ . Again, the activation values of the rule centres is 0 except for the respective rule. Additionally, we have to mention that although CBFSSs have been seen as low interpretable systems due to the overlap of the MFs, and since each rule has its own MF in each input variable [10], this model performs a pseudo-partitioning of the input space that is very intuitive as we can see from *Fig. 2*; no rule has effect in the rest of the rule centres, and besides, the activation value is limited according to the location of the rest of the centres in the input space. Every rule defines a region of the input space where it has a positive activation value around its respective rule centre, that is limited by the rest of rule centres positions.

Translating this new approach to the RBF networks, the traditional three layered structure of the RBF network could be modified so that a new layer is introduced between the hidden layer and the output layer that performs the calculations needed as seen in *Eq. 9*. Also for the Taylor Series-based rule consequent coefficients, the well-known regression weights (see [11]) can be expanded here to obtain higher order polynomial weights, for example of order 2 [5].

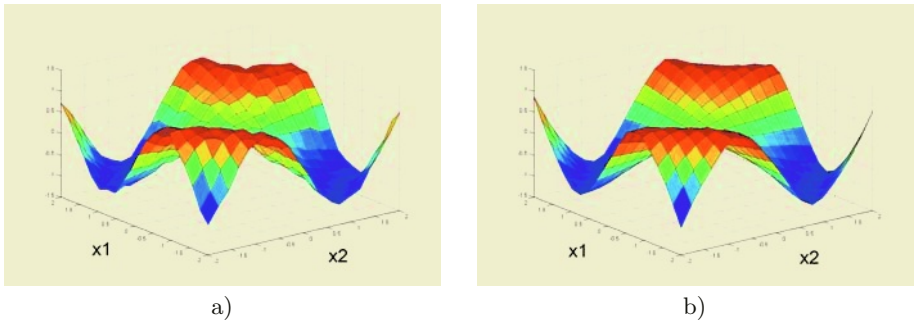
### 3 Simulations

In this section we present an application of our TaSe-C for function approximation using a significative example. Given the function  $f_1$  that is expressed as

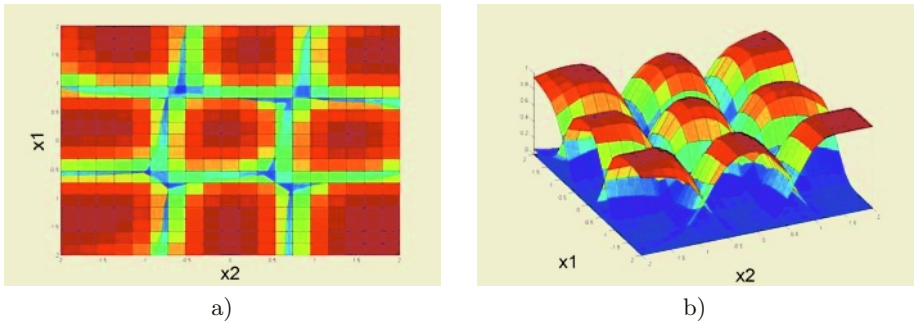
$$f_1(x_1, x_2) = \sin(x_1 * x_2) + \text{noise} \quad x \in [-2, 2] \tag{11}$$

we will generate and evaluate 400 equally distributed points as our I/O data set using a normally distributed noise term with  $\sigma = 0.05$ . As an example we will make use of a CBFS with second order polynomial consequent rules, and 9 centres that have been initialized using the input clustering k-means algorithm [12]. *Fig. 3* shows the original  $f_1$  function and the function approximation obtained by the TaSe-C model using the nine rules. As we can see, the approximation obtained using only 9 rules is quite precise (with a training Normalized Root-Mean Square Error NRMSE [8] = 0.082 and test error NRMSE = 0.050). *Fig. 4a* shows the distribution of the MFs  $\mu^{k*}$  in the input space centred on the nine clusters, and *Fig. 4b* shows the activation functions  $\mu^{k*}(x)$  for each rule.

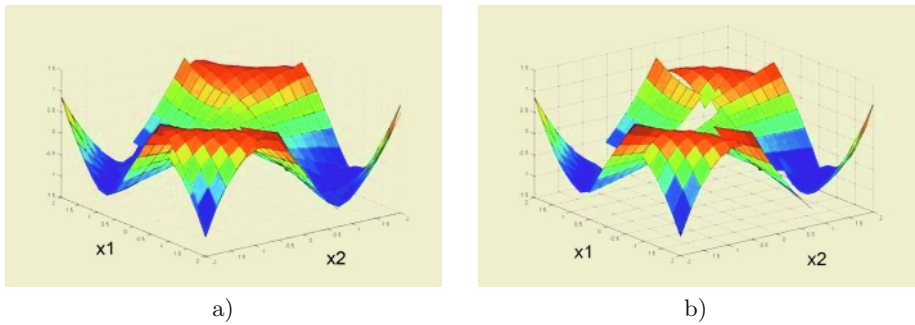
To see the interpretability properties obtained by our TaSe-C model, *Fig. 5a* shows the approximation obtained by the model and the second order polynomials of each rule consequent around each rule centre. Also in *Fig. 5b* we show the



**Fig. 3.** a) Original function  $f_1$ . b) Approximation obtained by the TaSe-C model using 9 rules located using the C-means algorithm



**Fig. 4.** a) Clusters location in the input space  $\mu^{k*}$ . b) Activation functions  $\mu^{k*}(x)$  for the nine rules



**Fig. 5.** a) Model output + polynomial representations around the centres. b) Polynomial representations around the centres

second order polynomials of each rule consequent. From both views we can see the high local interpretability obtained by the TaSe-C system for the local models. In fact, *each of the local model corresponds to the Taylor Series Expansion of the TaSe-C model output around each of the rule centres.*

## 4 Conclusions

In this paper we have presented a new neuro-fuzzy TSK model, the TaSe-C model, that brings interpretability to the local models (rules) in clustering based fuzzy systems using the Taylor Series Expansion of the model output around the rule centres. Several recent works have addressed the local models interpretability problem with partial success, sometimes performing a tradeoff between global accuracy and local models interpretability. Using our TaSe-C model, the desirable property inherent to the neuro-fuzzy TSK models is recovered, according to which TSK models can be seen as linearizations of the nonlinear subjacent system.

## Acknowledgements

This work has been partially supported by the Spanish CICYT Project TIN2004-01419.

## References

1. Takagi, T., Sugeno, M.: Fuzzy identification of systems and its applications to modelling and control. *IEEE Trans. Syst. Man and Cyber.* **15** (1985) 116–132
2. Johansen, T.A., Babuska, R.: Multiobjective Identification of Takagi-Sugeno Fuzzy Models. *IEEE Trans. Fuz. Syst.* **11(6)** (2003) 847–860
3. Zhou, S.M., Gan, J.Q.: Improving the interpretability of Takagi-Sugeno fuzzy model by using linguistic modifiers and a multiple objective learning scheme. *Int. Joint Conf. on Neural Networks IJCNN* (2004) 2385–2390

4. Bikdash, M.: A Highly Interpretable Form of Sugeno Inference Systems. *IEEE Trans. Fuz. Syst.* (1999) **7(6)** (1999) 686–696
5. Herrera, L.J., Pomares, H., Rojas, I., González, J., Valenzuela, O.: Function Approximation through Fuzzy Systems Using Taylor Series Expansion-Based Rules: Interpretability and Parameter Tuning. *Lecture Notes in Computer Science*. New York: Springer-Verlag, **2972** (2004) 252–261
6. Herrera, L.J., Pomares, H., Rojas, I., Valenzuela, O., Prieto, A.: TaSe, a Taylor Series Based Fuzzy System Model that Combines Interpretability and Accuracy. *Fuzzy Sets and Systems*. *Accepted*
7. González, J., Rojas, I., Pomares, H., Ortega, J., Prieto, A.: A new Clustering Technique for Function Aproximation. *IEEE Transactions on Neural Networks*, **13(1)**(2002) 132-142
8. Pomares, H., Rojas, I., Ortega, J., Prieto, A.: A systematic approach to a self-generating fuzzy rule-table for function approximation. *IEEE Trans. Syst., Man, Cybern.* **30** (2000) 431–447
9. Chiu, S.: Fuzzy Model Identification Based on Cluster Estimation. *Journal of Intelligent and Fuzzy Systems*, **2(3)** (1994)
10. Guillaume, S.: Designing fuzzy inference systems from data: an interpretability-oriented review. *IEEE Trans. Fuzzy Systems*, **9** (2001) 426-443.
11. Rojas, I., Rojas, F., Pomares, H., Herrera, L.J., Gonzalez, J., Valenzuela, O.: The Synergy between Classical and Soft-Computing Techniques for Time Series Prediction. *Lecture Notes in Computer Science*. New York: Springer-Verlag, **2972** (2004) 30–39
12. Duda, R. O., Hart, P. E.: *Pattern Classification and Scene Analysis*. New York: Wiley (1973)

# Genetically Optimized Hybrid Fuzzy Neural Networks with the Aid of TSK Fuzzy Inference Rules and Polynomial Neural Networks

Sung-Kwun Oh<sup>1</sup>, Witold Pedrycz<sup>2</sup>, Hyun-Ki Kim<sup>1</sup>, and Yong-Kab Kim<sup>3</sup>

<sup>1</sup> Department of Electrical Engineering, The University of Suwon, San 2-2 Wau-ri, Bongdam-eup, Hwaseong-si, Gyeonggi-do, 445-743, South Korea  
ohsk@wonkwang.ac.kr

<sup>2</sup> Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB T6G 2G6, Canada  
and Systems Research Institute, Polish Academy of Sciences, Warsaw, Poland

<sup>3</sup> Department of Electrical Electronic and Information Engineering, Wonkwang University, 344-2, Shinyong-Dong, Iksan, Chon-Buk, 570-749, South Korea

**Abstract.** We introduce an advanced architecture of genetically optimized Hybrid Fuzzy Neural Networks (gHFNN) and develop a comprehensive design methodology supporting their construction. The gHFNN architecture results from a synergistic usage of the hybrid system generated by combining Fuzzy Neural Networks (FNN) with Polynomial Neural Networks (PNN). We distinguish between two types of the linear fuzzy inference rule-based FNN structures showing how this taxonomy depends upon the type of a fuzzy partition of input variables. As to the consequence part of the gHFNN, the development of the PNN dwells on two general optimization mechanisms: the structural optimization is realized via GAs whereas in case of the parametric optimization we proceed with a standard least square method-based learning.

## 1 Introductory Remarks

The models should be able to take advantage of the existing domain knowledge and augment it by available numeric data to form a coherent data-knowledge modeling entity. The omnipresent modeling tendency is the one that exploits techniques of Computational Intelligence (CI) by embracing fuzzy modeling [1], [2], [3], [4], [5], [6], neurocomputing [7], and genetic optimization [8].

In this study, we develop a hybrid modeling architecture, called genetically optimized Hybrid Fuzzy Neural Networks (gHFNN). In a nutshell, gHFNN is composed of two main substructures driven to genetic optimization, namely a fuzzy set-based fuzzy neural network (FNN) and a polynomial neural network (PNN). From a standpoint of rule-based architectures, one can regard the FNN as an implementation of the antecedent part of the rules while the consequent is realized with the aid of a PNN. The role of the FNN is to interact with input data, granulate the corresponding input spaces. The FNNs come with two kinds of network architectures, namely fuzzy-set

based FNN and fuzzy-relation based FNN. The role of the PNN is to carry out nonlinear transformation at the level of the fuzzy sets formed at the level of FNN. The PNN that exhibits a flexible and versatile structure [9] is constructed on a basis of Group Method of Data Handling (GMDH [10]) method and genetic algorithms (GAs). The design procedure applied in the construction of each layer of the PNN deals with its structural optimization involving the selection of optimal nodes (polynomial neurons; PNs) with specific local characteristics (such as the number of input variables, the order of the polynomial, and a collection of the specific subset of input variables) and addresses specific aspects of parametric optimization. To assess the performance of the proposed model, we exploit a well-known time series data. Furthermore, the network is directly contrasted with several existing intelligent models.

## 2 Conventional Hybrid Fuzzy Neural Networks (HFNN)

The architectures of conventional HFNN [11], [12] result as a synergy between two other general constructs such as FNN and PNN. Based on the different PNN topologies, the HFNN distinguish between two kinds of architectures, namely basic and modified architectures. Moreover, for the each architecture we identify two cases. In the connection point, if input variables to PNN used on the consequence part of HFNN are less than three (or four), the generic type of HFNN does not generate a highly versatile structure. Accordingly we identify also two types as the generic and advanced. The topologies of the HFNN depend on those of the PNN used for the consequence part of HFNN. The design of the PNN proceeds further and involves a generation of some additional layers. Each layer consists of nodes (PNs) for which the number of input variables could be the same as in the previous layers or may differ across the network. The structure of the PNN is selected on the basis of the number of input variables and the order of the polynomial occurring in each layer.

## 3 Genetically Optimized HFNN (gHFNN)

The gHFNN emerges from the genetically optimized multi-layer perceptron architecture based on fuzzy set-based FNN, GAs and GMDH. These networks result as a synergy between two other general constructs such as FNN [13] and PNN [9].

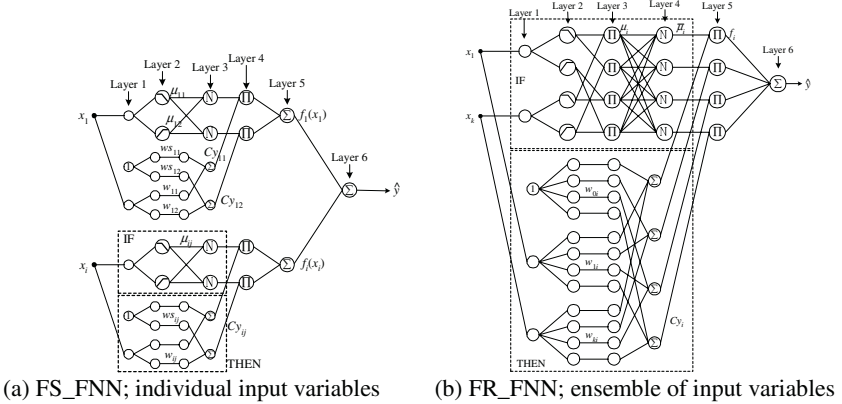
### 3.1 Fuzzy Neural Networks Based on Genetic Optimization

We consider two kinds of FNNs (viz. FS\_FNN and FR\_FNN) based on linear fuzzy inference. The FNN is designed by using space partitioning realized in terms of the individual input variables or an ensemble of all variables. The fuzzy partitions formed for each case lead us to the topologies visualized in Fig. 1. Table 1 represents the comparison of fuzzy rules, inference result and learning for two types of FNNs. In Table 1,  $R^j$  is the  $j$ -th fuzzy rule while  $A_{ij}$  denotes a fuzzy variable of the premise of the corresponding fuzzy rule and represents membership function  $\mu_{ij}$ .  $ws$  and  $w$  ex-

press a connection (weight) existing between the neurons as we have already visualized in Fig. 1.

**Table 1.** Comparison of fuzzy set with fuzzy relation-based FNNs

| Structure        | FS_FNN                                                                                                                                                                                                                                                                                          | FR_FNN                                                                                                                                                                                                                                                                 |
|------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Fuzzy rules      | $R^j: \text{If } x_i \text{ is } A_{ij} \text{ then } C_{y_{ij}} = w_{s_{ij}} + w_{ij} \cdot x_i$                                                                                                                                                                                               | $R: \text{If } x_i \text{ is } A_{i1} \dots \text{and } x_k \text{ is } A_{ik} \text{ then } C_{y_i} = w_{0i} + w_{1i} \cdot x_i + w_{ki} \cdot x_k$                                                                                                                   |
| Inference result | $\hat{y} = f_1(x_1) + f_2(x_2) + \dots + f_m(x_m) = \sum_{i=1}^m f_i(x_i)$ $f_i(x_i) = \frac{\sum_{j=1}^z (\mu_{ij}(x_i) \cdot (w_{s_{ij}} + x_i w_{ij}))}{\sum_{j=1}^z \mu_{ij}(x_i)}$ $= \mu_{ik}(x_i) \cdot (w_{s_{ik}} + x_i w_{ik}) + \mu_{ik+1}(x_i) \cdot (w_{s_{ik+1}} + x_i w_{ik+1})$ | $\hat{y} = \sum_{i=1}^n f_i$ $= \sum_{i=1}^n \bar{\mu}_i \cdot (w_{0i} + w_{1i} \cdot x_i + w_{ki} \cdot x_k)$ $= \sum_{i=1}^n \frac{\bar{\mu}_i \cdot (w_{0i} + w_{1i} \cdot x_i + w_{ki} \cdot x_k)}{\sum_{i=1}^n \bar{\mu}_i}$                                      |
| Learning         | $\begin{cases} \Delta w_{s_{ij}} = 2 \cdot \eta \cdot (y - \hat{y}) \cdot \mu_{ij} \\ \quad + \alpha (w_{s_{ij}}(t) - w_{s_{ij}}(t-1)) \\ \Delta w_{ij} = 2 \cdot \eta \cdot (y - \hat{y}) \cdot \mu_{ij} \cdot x_i \\ \quad + \alpha (w_{ij}(t) - w_{ij}(t-1)) \end{cases}$                    | $\begin{cases} \Delta w_{0i} = 2 \cdot \eta \cdot (y - \hat{y}) \cdot \bar{\mu}_i \\ \quad + \alpha (w_{0i}(t) - w_{0i}(t-1)) \\ \Delta w_{ki} = 2 \cdot \eta \cdot (y - \hat{y}) \cdot \bar{\mu}_i \cdot x_k \\ \quad + \alpha (w_{ki}(t) - w_{ki}(t-1)) \end{cases}$ |



**Fig. 1.** Topologies of FNN

The learning of FNN is realized by adjusting connections of the neurons and as such it follows a standard Back-Propagation (BP) algorithm [14]. For the linear fuzzy inference-based FNN, the update formula of a connection is as shown in Table 1. Genetic algorithms (GAs) are optimization techniques based on the principles of natural evolution. In essence, they are search algorithms that use operations found in natural genetics to guide a comprehensive search over the parameter space [8]. In order to enhance the learning of the FNN and augment its performance of a FNN, we use GAs



to adjust learning rate( $\eta$ ), momentum coefficient( $\alpha$ ) and the parameters of the membership functions of the antecedents of the rules.

### 3.2 Genetically Optimized PNN (gPNN)

When we construct PNs of each layer in the conventional PNN [9], such parameters as the number of input variables (nodes), the order of polynomial, and input variables available within a PN are fixed (selected) in advance by the designer. The overall genetically-driven optimization process of PNN is shown in Fig. 2.

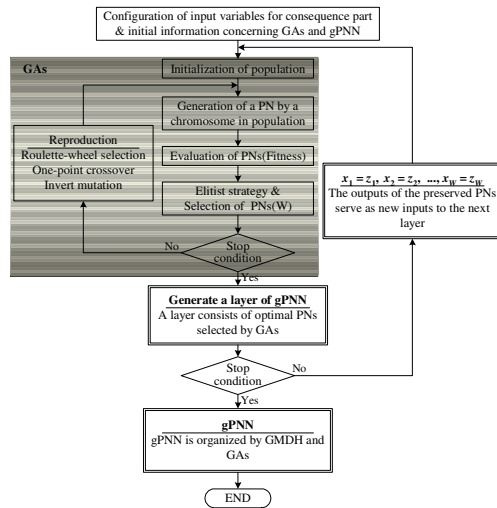


Fig. 2. Overall genetically-driven optimization process of PNN

## 4 The Algorithms and Design Procedure of gHFNN

### The premise of gHFNN: FS\_FNN (Refer to Fig. 1)

[Layer 1] Input layer.

[Layer 2] Computing activation degrees of linguistic labels.

[Layer 3] Normalization of a degree activation (firing) of the rule.

[Layer 4] Multiplying a normalized activation degree of the rule by connection. If we choose Connection point 1 for combining FNN with gPNN as shown in Fig. 1,  $a_{ij}$  is given as the input variable of the gPNN.

$$a_{ij} = \bar{\mu}_{ij} \times C_{y_{ij}} = \mu_{ij} \times C_{y_{ij}} \text{ (Here, } C_{y_{ij}} = ws_{ij} + w_{ij} \cdot x_i \text{)} \tag{1}$$

[Layer 5] Fuzzy inference for the TSK fuzzy rules. If we choose Connection point 2,  $f_i$  is the input variable of gPNN.

[Layer 6; Output layer of FNN] Computing output of a FNN.

The design procedure for each layer in FR\_FNN is carried out in a same manner as the one presented for FS\_FNN.

**The consequence of gHFNN: gPNN (Refer to Fig. 2)**

[Step 1] Configuration of input variables.

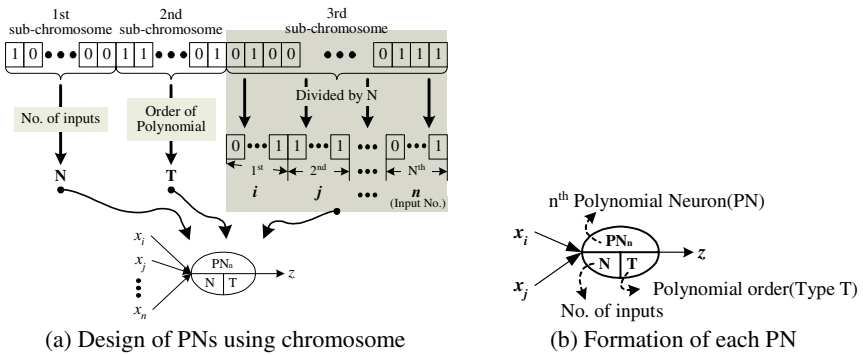
If we choose the first option (Connection point 1),  $x_1=a_{11}, x_2=a_{12}, \dots, x_n=a_{ij} (n=i \times j)$ .

For the second option (Connection point 2), we have  $x_1=f_1, x_2=f_2, \dots, x_n=f_m (n=m)$ .

[Step 2] Decision of initial information for constructing the gPNN.

[Step 3] Initialization of population.

[Step 4] Decision of PNs structure using genetic design. We divide the chromosome to be used for genetic optimization into three sub-chromosomes as shown in Fig. 3(a). In Fig. 3(b), ‘PN<sub>n</sub>’ denotes the n<sup>th</sup> PN (node) of the corresponding layer, ‘N’ denotes the number of inputs coming to the node, and ‘T’ denotes the polynomial order in the node (Refer to Table 2).



**Fig. 3.** The PN design using genetic optimization

**Table 2.** Different forms of regression polynomial forming a PN

| Number of inputs        | 2             | 3              | 4                |
|-------------------------|---------------|----------------|------------------|
| Order of the polynomial |               |                |                  |
| 1 (Type 1)              | Bilinear      | Trilinear      | Tetralinear      |
| 2 (Type 2)              | Biquadratic-1 | Triquadratic-1 | Tetraquadratic-1 |
| 2 (Type 3)              | Biquadratic-2 | Triquadratic-2 | Tetraquadratic-2 |

[Step 5] Evaluation of PNs.

[Step 6] Elitist strategy and selection of PNs with the best predictive capability.

[Step 7] Reproduction.

[Step 8] Repeating Step 4-7.

[Step 9] Construction of their corresponding layer.

[Step 10] Check the termination criterion (performance index) – MSE.

[Step 11] Determining new input variables for the next layer.

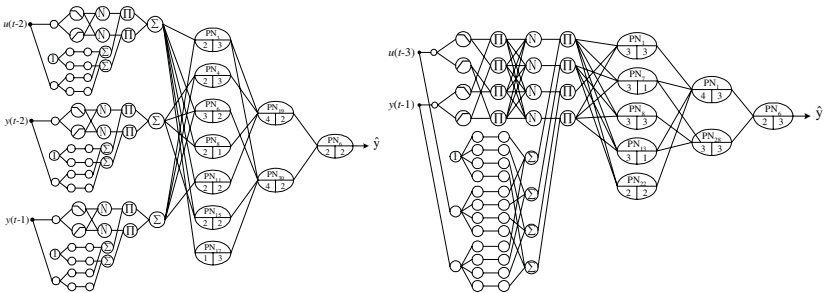
The gPNN algorithm is carried out by repeating Steps 4-11.

### 5 Experimental Studies

The performance of the gHFNN is illustrated with the aid of a time series of gas furnace [14]. We use two types of system input variables of FNN structure, Type I and Type II to design an optimal model from gas furnace data. Type I utilize two system input variables such as  $u(t-3)$  and  $y(t-1)$  and Type II utilizes 3 system input variables such as  $u(t-2)$ ,  $y(t-2)$ , and  $y(t-1)$ . The output variable is  $y(t)$ .

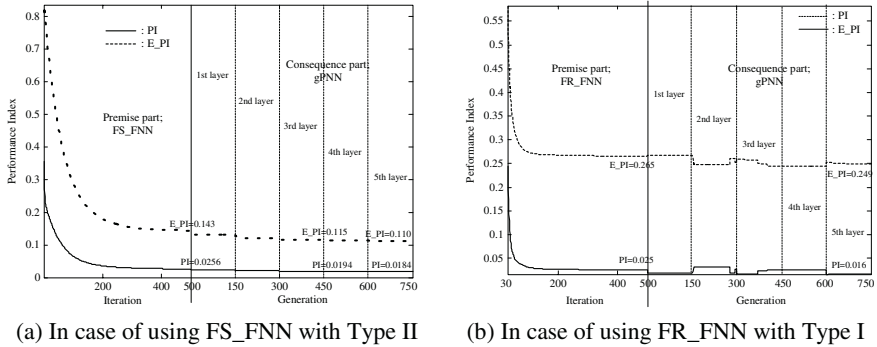
**Table 3.** Performance index of gHFNN for the gas furnace

| Structure | Premise part       |        |       | CP | Layer | Consequence part |           |       |       |    | PI | EPI    |       |
|-----------|--------------------|--------|-------|----|-------|------------------|-----------|-------|-------|----|----|--------|-------|
|           | No. of rules (MFs) | PI     | EPI   |    |       | No. of inputs    | Input No. |       |       | T  |    |        |       |
| FS_FNN    | 4<br>(2+2)         | 0.041  | 0.267 | 01 | 1     | 4                | 4         | 2     | 1     | 3  | 3  | 0.019  | 0.292 |
|           |                    |        |       |    | 2     | 4                | 7         | 12    | 2     | 10 | 2  | 0.018  | 0.271 |
|           |                    |        |       |    | 3     | 4                | 20        | 21    | 5     | 3  | 2  | 0.017  | 0.267 |
|           |                    |        |       |    | 4     | 3                | 22        | 13    | 29    | .  | 2  | 0.016  | 0.263 |
|           |                    |        |       |    | 5     | 4                | 25        | 18    | 27    | 9  | 3  | 0.015  | 0.258 |
|           | 6<br>(2+2+2)       | 0.0256 | 0.143 | 02 | 1     | 3                | 1         | 2     | 3     | .  | 3  | 0.0232 | 0.130 |
|           |                    |        |       |    | 2     | 4                | 12        | 15    | 13    | 6  | 2  | 0.0196 | 0.120 |
|           |                    |        |       |    | 3     | 2                | 19        | 30    | .     | .  | 2  | 0.0194 | 0.115 |
|           |                    |        |       |    | 4     | 4                | 2         | 21    | 11    | 5  | 1  | 0.0188 | 0.113 |
|           |                    |        |       |    | 5     | 4                | 13        | 3     | 26    | 25 | 1  | 0.0184 | 0.110 |
| FR_FNN    | 4<br>(2x2)         | 0.025  | 0.265 |    | 1     | 4                | 4         | 1     | 3     | 2  | 2  | 0.019  | 0.267 |
|           |                    |        |       |    | 2     | 4                | 7         | 1     | 13    | 22 | 3  | 0.026  | 0.251 |
|           |                    |        |       |    | 3     | 2                | 1         | 28    | .     | .  | 3  | 0.025  | 0.244 |
|           |                    |        |       |    | 4     | 2                | 7         | 6     | .     | .  | 3  | 0.025  | 0.243 |
|           | 8<br>(2x2x2)       | 0.033  | 0.119 |    | 5     | 3                | 29        | 22    | 17    | .  | 3  | 0.016  | 0.249 |
|           |                    |        |       |    | 1     | 4                | 6         | 5     | 2     | 8  | 1  | 0.083  | 0.146 |
|           |                    |        |       |    | 2     | 4                | 21        | 18    | 6     | 9  | 2  | 0.028  | 0.116 |
|           |                    |        |       |    | 3     | 4                | 4         | 24    | 5     | 6  | 2  | 0.022  | 0.110 |
|           | 4                  | 3      | 28    | 4  | 5     | .                | 2         | 0.021 | 0.106 |    |    |        |       |
|           | 5                  | 3      | 21    | 18 | 25    | .                | 1         | 0.021 | 0.104 |    |    |        |       |



(a) In case of using FS\_FNN with Type II (b) In case of using FR\_FNN with Type I

**Fig. 4.** Optimal topology of genetically optimized HFNN for the gas furnace



**Fig. 5.** Optimization procedure of gHFNN by BP learning and GAs

Table 3 includes the results of the overall network reported according to various alternatives concerning various forms of FNN architecture, format of entire system inputs and location of the connection point (CP). The optimal topology of gHFNN is shown in Fig. 4. Fig. 5 illustrates the optimization process by visualizing the performance index in successive cycles. Table 4 contrasts the performance of the genetically developed network with other fuzzy and fuzzy-neural networks studied in the literatures.

**Table 4.** Comparison of performance with other modeling methods

| Model                           |               | PI            | EPI   | No. of rules |                                         |
|---------------------------------|---------------|---------------|-------|--------------|-----------------------------------------|
| Box and Jenkin's model [14]     |               | 0.710         |       |              |                                         |
| Pedrycz's model [1]             |               | 0.320         |       |              |                                         |
| Xu and Zailu's model [2]        |               | 0.328         |       |              |                                         |
| Sugeno and Yasukawa's model [3] |               | 0.190         |       |              |                                         |
| Kim, et al.'s model [15]        |               | 0.034         | 0.244 | 2            |                                         |
| Lin and Cunningham's mode [16]  |               | 0.071         | 0.261 | 4            |                                         |
| Fuzzy                           | Complex [4]   | Simplified    | 0.024 | 0.328        | 4(2×2)                                  |
|                                 |               | Linear        | 0.023 | 0.306        | 4(2×2)                                  |
|                                 | Hybrid [6]    | Simplified    | 0.024 | 0.329        | 4(2×2)                                  |
|                                 | (GAs+Complex) | Linear        | 0.017 | 0.289        | 4(2×2)                                  |
|                                 | HCM+GAs [5]   | Simplified    | 0.022 | 0.333        | 6(3×2)                                  |
|                                 |               | Linear        | 0.020 | 0.264        | 6(3×2)                                  |
| FNN [13]                        |               | Simplified    | 0.043 | 0.264        | 6(3+3)                                  |
|                                 |               | Linear        | 0.037 | 0.273        | 6(3+3)                                  |
| SOFNN                           |               | Generic [11]  | 0.023 | 0.277        | 4 rules/5 <sup>th</sup> layer(NA)       |
|                                 |               |               | 0.020 | 0.119        | 6 rules/5 <sup>th</sup> layer(22 nodes) |
|                                 |               | Advanced [12] | 0.019 | 0.264        | 4 rules/5 <sup>th</sup> layer(NA)       |
| Proposed model<br>(gHFNN)       |               |               | 0.017 | 0.113        | 6 rules/5 <sup>th</sup> layer(26 nodes) |
|                                 |               |               | 0.017 | 0.267        | 4 rules/3 <sup>rd</sup> layer(16 nodes) |
|                                 |               | FS_FNN        | 0.019 | 0.115        | 6 rules/3 <sup>rd</sup> layer(10 nodes) |
|                                 |               |               | 0.025 | 0.244        | 4 rules/3 <sup>rd</sup> layer(8 nodes)  |
|                                 |               | FR_FNN        | 0.022 | 0.110        | 7 rules/3 <sup>rd</sup> layer(14 nodes) |

## 6 Concluding Remarks

The genetically optimized HFNNs are constructed by combining FNNs with gPNNs. The proposed model comes with two kinds of rule-based FNNs (viz. FS\_FNN and

FR\_FNN based on linear fuzzy inferences) as well as a diversity of local characteristics of PNs that are extremely useful when coping with various nonlinear characteristics of the system under consideration. In what follows, in contrast to the conventional HFNN structures and their learning, the depth (the number of layers) and the width (the number of nodes) as well as the number of entire nodes (inputs) of the proposed genetically optimized HFNN (gHFNN) can be lower.

The comprehensive design methodology comes with the parametrically as well as structurally optimized network architecture. 1) As the premise structure of the gHFNN, the optimization of the rule-based FNN hinges on GAs and BP. And 2) the gPNN that is the consequent structure of the gHFNN is based on the technologies of the PNN and GAs. The PNN is comprised of both a structural phase such as a self-organizing and evolutionary algorithm, and a parametric phase of least square estimation-based learning, moreover the PNN is driven to genetic optimization, in what follows it leads to the selection of the optimal nodes. In the sequel, a variety of architectures of the proposed gHFNN driven to genetic optimization have been discussed.

**Acknowledgement.** This work was supported by Korea Research Foundation Grant (KRF-2004-002-D00257).

## References

1. Pedrycz, W.: An identification algorithm in fuzzy relational system. *Fuzzy Sets and Systems*. **13** (1984) 153-167
2. Xu, C.W., Zailu, Y.: Fuzzy model identification self-learning for dynamic system. *IEEE Trans. on Syst. Man, Cybern. SMC*-**17**(4) (1987) 683-689
3. Sugeno, M., Yasukawa, T.: A Fuzzy-Logic-Based Approach to Qualitative Modeling. *IEEE Trans. Fuzzy Systems*. **1**(1) (1993) 7-31
4. Oh, S.K., Pedrycz, W.: Fuzzy Identification by Means of Auto-Tuning Algorithm and Its Application to Nonlinear Systems. *Fuzzy Sets and Systems*. **115**(2) (2000) 205-230
5. Park, B.J., Pedrycz, W., Oh, S.K.: Identification of Fuzzy Models with the Aid of Evolutionary Data Granulation. *IEE Proceedings-Control theory and application*. **148**(5) (2001) 406-418
6. Oh, S.K., Pedrycz, W., Park, B.J.: Hybrid Identification of Fuzzy Rule-Based Models. *International Journal of Intelligent Systems*. **17**(1) (2002) 77-103
7. Narendra, K.S., Parthasarathy, K.: Gradient Methods for the Optimization of Dynamical Systems Containing Neural Networks. *IEEE Transactions on Neural Networks*, **2** (1991) 252-262
8. Michalewicz, Z.: *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, Berlin Heidelberg (1996)
9. Oh, S.K., Pedrycz, W., Park, B.J.: Polynomial Neural Networks Architecture: Analysis and Design. *Computers and Electrical Engineering*. **29**(6) (2003) 653-725
10. Ivahnenko, A.G.: The group method of data handling: a rival of method of stochastic approximation. *Soviet Automatic Control*. **13**(3) (1968) 43-55
11. Park, B.J., Oh, S.K., Jang, S.W.: The Design of Adaptive Fuzzy Polynomial Neural Networks Architectures Based on Fuzzy Neural Networks and Self-Organizing Networks. *Journal of Control, Automation and Systems Engineering*. **8**(2) (2002) 126-135 (In Korean)

12. Park, B.J., Oh, S.K.: The Analysis and Design of Advanced Neurofuzzy Polynomial Networks. *Journal of the Institute of Electronics Engineers of Korea*. **39**-CI(3) (2002) 18-31
13. Oh, S.K., Pedrycz, W., Park, H.S.: Hybrid Identification in Fuzzy-Neural Networks. *Fuzzy Sets and Systems*. **138**(2) (2003) 399-426
14. Box, D.E.P., Jenkins, G.M.: *Time Series Analysis, Forecasting, and Control*, 2nd edition Holden-Day, SanFransisco (1976)
15. Kim, E., Lee, H., Park, M., Park, M.: A Simply Identified Sugeno-type Fuzzy Model via Double Clustering. *Information Sciences*. **110** (1998) 25-39
16. Lin, Y., Cunningham, G. A. III.: A new Approach to Fuzzy-neural Modeling. *IEEE Transaction on Fuzzy Systems*. **3**(2) 190-197
17. Park, B.J., Lee, D.Y., Oh, S.K.: Rule-based Fuzzy Polynomial Neural Networks in Modeling Software Process Data. *Int. J. of Control, Automations, and Systems*. **1**(3) (2003) 321-331

# IG-Based Genetically Optimized Fuzzy Polynomial Neural Networks

Sung-Kwon Oh<sup>1</sup>, Seok-Beom Roh<sup>2</sup>, Witold Pedrycz<sup>3</sup>, and Jong-Beom Lee<sup>2</sup>

<sup>1</sup> Department of Electrical Engineering, The University of Suwon, San 2-2 Wau-ri, Bongdam-eup, Hwaseong-si, Gyeonggi-do, 445-743, South Korea  
ohsk@wonkwang.ac.kr

<sup>2</sup> Department of Electrical Electronic and Information Engineering, Wonkwang University, 344-2, Shinyong-Dong, Iksan, Chon-Buk, 570-749, South Korea

<sup>3</sup> Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB T6G 2G6, Canada and Systems Research Institute, Polish Academy of Sciences, Warsaw, Poland

**Abstract.** In this paper, we introduce a neo scheme of fuzzy-neural networks – Fuzzy Polynomial Neural Networks (FPNN) with a new fuzzy set-based polynomial neurons (FSPNs) whose fuzzy rules include the information granules (about the real system) obtained through Information Granulation(IG). We investigate the proposed networks from two different aspects to improve the performance of the fuzzy-neural networks. First, We have developed a design methodology (genetic optimization using Genetic Algorithms) to find the optimal structure for fuzzy-neural networks that expanded from Group Method of Data Handling (GMDH). It is the number of input variables, the order of the polynomial, the number of membership functions, and a collection of the specific subset of input variables that are the parameters of FPNN fixed by aid of genetic optimization that has search capability to find the optimal solution on the solution space. Second, we have been interested in the architecture of fuzzy rules that mimic the real world, namely sub-model (node) composing the fuzzy-neural networks. We adopt fuzzy set-based fuzzy rules as substitute for fuzzy relation-based fuzzy rules and apply the concept of Information Granulation to the proposed fuzzy set-based rules. The performance of genetically optimized FPNN (gFPNN) with fuzzy set-based polynomial neurons (FSPNs) composed of fuzzy set-based rules is quantified through experimentation where we use a number of modeling benchmarks data which are already experimented with in fuzzy or neurofuzzy modeling.

## 1 Introduction

A lot of researchers on system modeling have been interested in the multitude of challenging and conflicting objectives such as compactness, approximation ability, generalization capability and so on which they wish to satisfy. Fuzzy sets emphasize the aspect of linguistic transparency of models and a role of a model designer whose prior knowledge about the system may be very helpful in facilitating all identification pursuits. In addition, to build models with substantial approximation capabilities, there should be a need for advanced tools.

As one of the representative advanced design approaches comes a family of self-organizing networks with fuzzy polynomial neuron (FPN) (called “FPNN” as a new category of neuro-fuzzy networks) [1], [4], [8]. The design procedure of the FPNNs exhibits some tendency to produce overly complex networks as well as comes with a repetitive computation load caused by the trial and error method being a part of the development process.

In this paper, in considering the above problems coming with the conventional FPNN [1], [4], [8], we introduce a new structure of fuzzy rules as well as a new genetic design approach. The new structure of fuzzy rules based on the fuzzy set-based approach changes the viewpoint of input space division. In other hand, from a point of view of a new understanding of fuzzy rules, information granules seem to melt into the fuzzy rules respectively. The determination of the optimal values of the parameters available within an individual FSPN leads to a structurally and parametrically optimized network through the genetic approach.

## 2 The Architecture and Development of Fuzzy Polynomial Neural Networks (FPNN)

The FSPN encapsulates a family of nonlinear “if-then” rules. When put together, FSPNs results in a self-organizing Fuzzy Set-based Polynomial Neural Networks (FSPNN). As visualized in Fig. 1, the FSPN consists of two basic functional modules. The first one, labeled by F, is a collection of fuzzy sets (here denoted by  $\{A_k\}$  and  $\{B_k\}$ ) that form an interface between the input numeric variables and the processing part realized by the neuron. The second module (denoted here by P) refers to the function – based nonlinear (polynomial) processing that involves some input variables. This nonlinear processing involves some input variables ( $x_i$  and  $x_j$ ), which are capable of being the input variables (Here,  $x_p$  and  $x_q$ ), or entire system input variables. Each rule reads in the form

$$\begin{aligned} &\text{if } x_p \text{ is } A_k \text{ then } z \text{ is } P_{pk}(x_i, x_j, \mathbf{a}_{pk}) \\ &\text{if } x_q \text{ is } B_k \text{ then } z \text{ is } P_{qk}(x_i, x_j, \mathbf{a}_{qk}) \end{aligned} \tag{1}$$

where  $\mathbf{a}_{qk}$  is a vector of the parameters of the conclusion part of the rule while  $P(x_i, x_j, \mathbf{a})$  denoted the regression polynomial forming the consequence part of the fuzzy rule. The activation levels of the rules contribute to the output of the FSPN being computed as a weighted average of the individual condition parts (functional transformations)  $P_K$  (note that the index of the rule, namely “K” is a shorthand notation for the two indexes of fuzzy sets used in the rule (1), that is  $K = (l, k)$ ).

$$\begin{aligned} z &= \sum_{l=1}^{\text{total\_inputs}} \left( \frac{\sum_{k=1}^{\text{total\_rules related to input } l} \mu_{(l,k)} P_{(l,k)}(x_i, x_j, \mathbf{a}_{(l,k)})}{\sum_{k=1}^{\text{total\_rules related to input } l} \mu_{(l,k)}} \right) \\ &= \sum_{l=1}^{\text{total\_inputs}} \left( \sum_{k=1}^{\text{rules related to input } l} \tilde{\mu}_{(l,k)} P_{(l,k)}(x_i, x_j, \mathbf{a}_{(l,k)}) \right) \end{aligned} \tag{2}$$



In the above expression, we use an abbreviated notation to describe an activation level of the “*K*” th rule to be in the form

$$\tilde{\mu}_{(l,k)} = \frac{\mu_{(l,k)}}{\sum_{k=1}^{\text{total\_rules related to input } l} \mu_{(l,k)}} \tag{3}$$

When developing an FSPN, we use genetic algorithms to produce the optimized network. This is realized by selecting such parameters as the number of input variables, the order of polynomial, and choosing a specific subset of input variables. Based on the genetically optimized number of the nodes (input variables) and the polynomial order, refer to Table 1, we construct the optimized self-organizing network architectures of the FSPNNs.

**Table 1.** Different forms of the regression polynomials forming the consequence part of the fuzzy rules

| No. of inputs<br>Order of<br>the polynomial | 1         | 2             | 3              |
|---------------------------------------------|-----------|---------------|----------------|
| 0 (Type 1)                                  | Constant  | Constant      | Constant       |
| 1 (Type 2)                                  | Linear    | Bilinear      | Trilinear      |
| 2 (Type 3)                                  | Quadratic | Biquadratic-1 | Triquadratic-1 |
| 2 (Type 4)                                  |           | Biquadratic-2 | Triquadratic-2 |

1: Basic type, 2: Modified type

### 3 Information Granulation Through Hard C-Means Clustering Algorithm

Information granules are defined informally as linked collections of objects (data points, in particular) drawn together by the criteria of indistinguishability, similarity or functionality [9]. Granulation of information is a procedure to extract meaningful concepts from numeric data and an inherent activity of human being carried out with intend of better understanding of the problem. We granulate information into some classes with the aid of Hard C-means clustering algorithm, which deals with the conventional crisp sets.

#### 3.1 Definition of the Premise and Consequent Part of Fuzzy Rules Using Information Granulation

We assume that given a set of data  $X=\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$  related to a certain application, there are some clusters which are capable of being found through HCM. The center point and the membership elements represent each cluster. The set of membership elements is crisp. To construct a fuzzy mode, we should transform the crisp set into the fuzzy set. The center point means the apex of the membership function of the fuzzy set. Let us consider building the consequent part of fuzzy rule. We can think of

each cluster as a sub-model composing the overall system. The fuzzy rules of Information Granulation-based FSPN are as followings.

$$\begin{aligned}
 &\text{if } x_p \text{ is } \mathbf{A}_k^* \text{ then } z\text{-}m_{pk} = P_{pk}((x_i\text{-}v_{pk}^i), (x_j\text{-}v_{pk}^j), a_{pk}) \\
 &\text{if } x_q \text{ is } \mathbf{B}_k^* \text{ then } z\text{-}m_{qk} \text{ is } P_{qk}((x_i\text{-}v_{qk}^i), (x_j\text{-}v_{qk}^j), a_{qk})
 \end{aligned} \tag{4}$$

Where,  $\mathbf{A}_k^*$  and  $\mathbf{B}_k^*$  mean the fuzzy set, the apex of which is defined as the center point of information granule (cluster) and  $m_{pk}$  is the center point related to the output variable on cluster $_{pk}$ ,  $v_{pk}^i$  is the center point related to the  $i$ -th input variable on cluster $_{pk}$  and  $\mathbf{a}_{qk}$  is a vector of the parameters of the conclusion part of the rule while  $P((x_i\text{-}v^i), (x_j\text{-}v^j), a)$  denoted the regression polynomial forming the consequence part of the fuzzy rule which uses several types of high-order polynomials (linear, quadratic, and modified quadratic) besides the constant function forming the simplest version of the consequence; refer to Table 1. If we are given  $m$  inputs and one output system and the consequent part of fuzzy rules is linear, the overall procedure of modification of the generic fuzzy rules is as followings. The given inputs are  $\mathbf{X}=[\mathbf{x}_1 \mathbf{x}_2 \dots \mathbf{x}_m]$  related to a certain application, where  $\mathbf{x}_k=[x_{k1} \dots x_{kn}]^T$ ,  $n$  is the number of data and  $m$  is the number of variables and the output is  $\mathbf{Y}=[y_1 y_2 \dots y_n]^T$ .

**Step 1)** build the universe set

Universe set  $U=\{ \{x_{11}, x_{12}, \dots, x_{1m}, y_1\}, \{x_{21}, x_{22}, \dots, x_{2m}, y_2\}, \dots, \{x_{n1}, x_{n2}, \dots, x_{nm}, y_n\} \}$

**Step 2)** build  $m$  reference data pairs composed of  $[\mathbf{x}_1; \mathbf{Y}]$ ,  $[\mathbf{x}_2; \mathbf{Y}]$ , and  $[\mathbf{x}_m; \mathbf{Y}]$ .

**Step 3)** classify the universe set  $U$  into  $l$  clusters such as  $c_{i1}, c_{i2}, \dots, c_{il}$  (subsets) by using HCM according to the reference data pair  $[\mathbf{x}_i; \mathbf{Y}]$ . Where  $c_{ij}$  means the  $j$ -th cluster (subset) according to the reference data pair  $[\mathbf{x}_i; \mathbf{Y}]$ .

**Step 4)** construct the premise part of the fuzzy rules related to the  $i$ -th input variable ( $x_i$ ) using the directly obtained center points from HCM.

**Step 5)** construct the consequent part of the fuzzy rules related to the  $i$ -th input variable ( $x_i$ ). On this step, we need the center points related to all input variables. We should obtain the other center points through the indirect method as followings.

**Sub-step1)** make a matrix as shown in (5) according to the clustered subsets

$$\mathbf{A}_j^i = \left[ \begin{array}{cccc|c} x_{21} & x_{22} & \dots & x_{2m} & y_2 \\ x_{51} & x_{52} & \dots & x_{5m} & y_5 \\ x_{k1} & x_{k2} & \dots & x_{km} & y_k \\ \vdots & \vdots & \dots & \vdots & \vdots \end{array} \right] \tag{5}$$

Where,  $\{x_{k1}, x_{k2}, \dots, x_{km}, y_k\} \in c_{ij}$  and  $\mathbf{A}_j^i$  means the membership matrix of  $j$ -th subset related to the  $i$ -th input variable.

**Sub-step2)** take an arithmetic mean of each column on  $\mathbf{A}_j^i$ . The arithmetic mean of each column is the additional center point of subset  $c_{ij}$ . The arithmetic mean of column is described as (6)

$$\text{center points} = \left[ v_{ij}^1 \quad v_{ij}^2 \quad \dots \quad v_{ij}^m \quad \middle| \quad m_{ij} \right] \tag{6}$$

**Step 6)** if  $i$  is  $m$  then terminate, otherwise, set  $i=i+1$  and return step 3.

### 4 Genetic Optimization of FPNN

GAs are aimed at the global exploration of a solution space. The main features of genetic algorithms concern individuals viewed as strings, population-based optimization and stochastic search mechanism (selection and crossover). GAs use serial method of binary type, roulette-wheel as the selection operator, one-point crossover, and an invert operation in the mutation operator [2]. In this study, for the optimization of the FPNN model, GA uses the serial method of binary type, roulette-wheel used in the selection process, one-point crossover in the crossover operation, and a binary inversion operation in the mutation operator. To retain the best individual and carry it over to the next generation, we use elitist strategy [3]. The overall genetically-driven structural optimization process of FPNN is shown in Fig. 1.

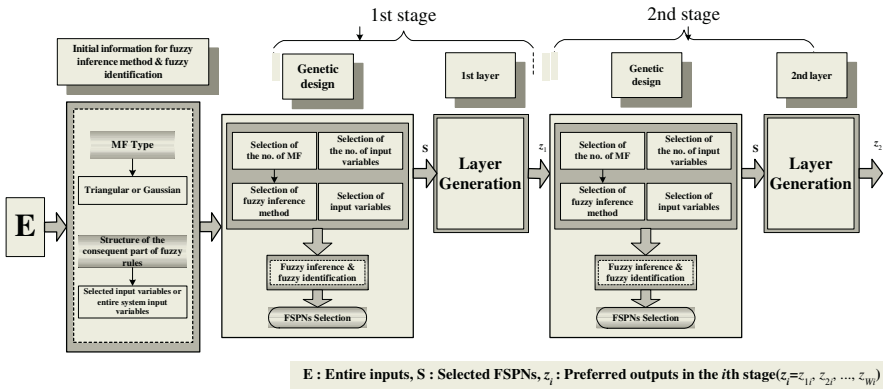


Fig. 1. Overall genetically-driven structural optimization process of FPNN

The framework of the design procedure of the genetically optimized FSPNN comprises the following steps

- [Step 1] Determine system’s input variables
- [Step 2] Form training and testing data
- [Step 3] specify initial design parameters
- [Step 4] Decide FSPN structure using genetic design
- [Step 5] Carry out fuzzy-set based fuzzy inference and coefficient parameters estimation for fuzzy identification in the selected node (FSPN)
- [Step 6] Check the termination criterion
- [Step 7] Determine new input variables for the next layer.

### 5 Experimental Studies

We demonstrate how the IG-gFSPNN can be utilized to predict future values of a chaotic Mackey-Glass time series. This time series is used as a benchmark in fuzzy

and neurofuzzy modeling. The performance of the network is also contrasted with some other models existing in the literature [5], [6], [7]. The time series is generated by the chaotic Mackey-Glass differential delay equation. To come up with a quantitative evaluation of the network, we use the standard RMSE performance index.

Fig. 2 depicts the performance index of each layer of gFPNN with Type T\* according to the increase of maximal number of inputs to be selected.

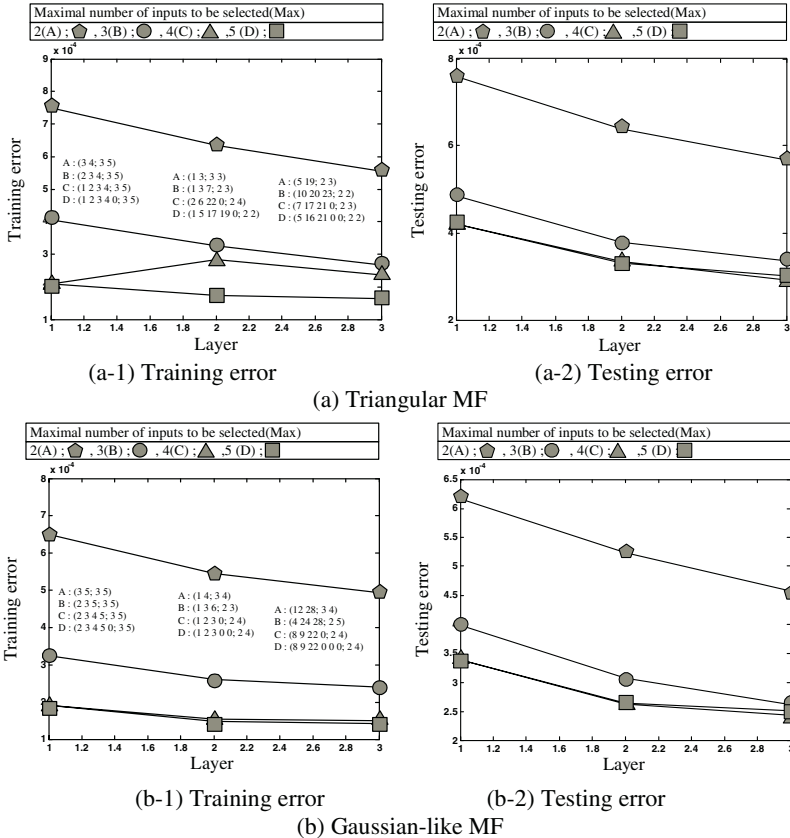
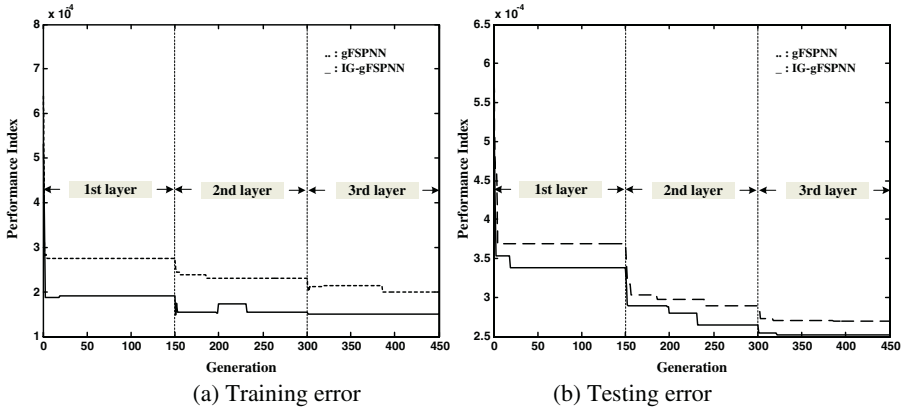


Fig. 2. Performance index of IG-gFPNN (with Type T\*) with respect to the increase of number of layers

Fig. 3 illustrates the different optimization process between gFPNN and the proposed IG-gFPNN by visualizing the values of the performance index obtained in successive generations of GA when using Type T\*.

Table 2 summarizes a comparative analysis of the performance of the network with other models.



**Fig. 3.** The optimization process quantified by the values of the performance index (in case of using Gaussian MF with Max=5 and Type T\*)

**Table 2.** Comparative analysis of the performance of the network; considered are models reported in the literature

| Model              |                                    | Performance index |                 |                  |
|--------------------|------------------------------------|-------------------|-----------------|------------------|
|                    |                                    | PI                | PI <sub>s</sub> | EPI <sub>s</sub> |
| Wang’s model[5]    |                                    | 0.044             |                 |                  |
|                    |                                    | 0.013             |                 |                  |
|                    |                                    | 0.010             |                 |                  |
| ANFIS[6]           |                                    |                   | 0.0016          | 0.0015           |
| FNN model[7]       |                                    |                   | 0.014           | 0.009            |
| Proposed IG-gFSPNN | Triangular (2 <sup>nd</sup> layer) | Max=5             | 1.72e-4         | 3.30e-4          |
|                    | Gaussian (2 <sup>nd</sup> layer)   | Max=5             | 1.48e-4         | 2.61e-4          |

## 6 Concluding Remarks

In this study, we have surveyed the new structure and meaning of fuzzy rules and investigated the GA-based design procedure of Fuzzy Polynomial Neural Networks (FPNN) along with its architectural considerations. The whole system is divided into some sub-systems that are classified according to the characteristics named information granules. Each information granule seems to be a representative of the related sub-systems. A new fuzzy rule with information granule describes a sub-system as a stand-alone system. A fuzzy system with some new fuzzy rules depicts the whole system as a combination of some stand-alone sub-system.

The GA-based design procedure applied at each stage (layer) of the FSPNN leads to the selection of the preferred nodes (or FSPNs) with optimal local characteristics (such as the number of input variables, the order of the consequent polynomial of fuzzy rules, and input variables) available within FSPNN. The comprehensive ex-

perimental studies involving well-known datasets quantify a superb performance of the network in comparison to the existing fuzzy and neuro-fuzzy models.

**Acknowledgement.** This work has been supported by KESRI(R-2004-B-274), which is funded by MOCIE (Ministry of commerce, industry and energy).

## References

1. Oh, S.K., Pedrycz, W.: Self-organizing Polynomial Neural Networks Based on PNs or FPNs : Analysis and Design, Fuzzy Sets and Systems, 2003(in press)
2. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs, Springer-Verlag, Berlin Heidelberg, 1996
3. Jong, D.K.A.: Are Genetic Algorithms Function Optimizers?. Parallel Problem Solving from Nature 2, Manner, R. and Manderick, B. eds., North-Holland, Amsterdam (1992)
4. Oh, S.K., Pedrycz, W.: Fuzzy Polynomial Neuron-Based Self-Organizing Neural Networks. *Int. J. of General Systems*. **32** (2003) 237-250
5. Wang, L.X., Mendel, J.M.: Generating fuzzy rules from numerical data with applications, *IEEE Trans. Systems, Man, Cybern.*, **22**(6) (1992) 1414-1427
6. Jang, J.S.R.: ANFIS: Adaptive-Network-Based Fuzzy Inference System. *IEEE Trans. System, Man, and Cybern.* **23** (1993) 665-685
7. Maguire, L.P., Roche, B., McGinnity, T.M., McDaid, L.J.: Predicting a chaotic time series using a fuzzy neural network. *Information Sciences*. **112** (1998) 125-136
8. Oh, S.K., Pedrycz, W., Ahn, T.C.: Self-organizing neural networks with fuzzy polynomial neurons. *Applied Soft Computing*. **2** (2002) 1-10
9. Zadeh, L.A.: Toward a theory of fuzzy information granulation and its centrality in human reasoning and fuzzy logic. *Fuzzy sets and Systems*. **90** (1997) 111-117
10. Park, B.J., Lee, D.Y., Oh, S.K.: Rule-based Fuzzy Polynomial Neural Networks in Modeling Software Process Data. *Int. J. of Control, Automations, and Systems*. **1**(3) (2003) 321-331

# Hierarchical Neuro-fuzzy Models Based on Reinforcement Learning for Intelligent Agents

Karla Figueiredo<sup>1</sup>, Marley Velasco<sup>2,3</sup>,  
and Marco Aurélio Pacheco<sup>2,3</sup>

<sup>1</sup> Department of Electronic and Telecommunication Engineering - UERJ,  
Rua São Francisco Xavier, 524, Maracanã - CEP 20550-900, Rio de Janeiro RJ – Brasil  
karlaf@uerj.br

<sup>2</sup> Applied Computational Intelligence Lab.,  
Department of Electrical Engineering – PUC-Rio,  
Rua Marquês de São Vicente, 225, Rio de Janeiro, 22453-900 RJ, Brazil  
{marley, marco}@ele.puc-rio.br

<sup>3</sup> Department of Computer and Systems Engineering - UERJ

**Abstract.** This work introduces two new neuro-fuzzy systems for intelligent agents called Reinforcement Learning - Hierarchical Neuro-Fuzzy Systems BSP (RL-HNFB) and Reinforcement Learning - Hierarchical Neuro-Fuzzy Systems Politree (RL-HNFP). By using hierarchical partitioning methods, together with the Reinforcement Learning (RL) methodology, a new class of Neuro-Fuzzy Systems (SNF) was obtained, which executes, in addition to automatically learning its structure, the autonomous learning of the actions to be taken by an agent. These characteristics have been developed in order to bypass the traditional drawbacks of neuro-fuzzy systems. The paper details the two novel RL\_HNF systems and evaluates their performance in a benchmark application – the cart-centering problem. The results obtained demonstrate the capacity of the proposed models in extracting knowledge from the agent's direct interaction with large and/or continuous environments.

## 1 Introduction

This work presents two news hybrid neuro-fuzzy models, called *Reinforcement Learning Hierarchical Neuro-Fuzzy BSP* (RL-HNFB) [1] and *Reinforcement Learning Hierarchical Neuro-Fuzzy Politree* (RL-HNFP), which are based on a Reinforcement Learning algorithm to provide an agent with intelligence, making it capable, by interacting with its environment, to acquire and retain knowledge for reasoning (infer an action).

The proposed models were devised based on an analysis of the limitations in the existing RL models [2-3] and on the desirable characteristics for RL-based learning systems, particularly in applications involving continuous environments and/or environments considered to be highly dimensional.

When the environment is large and/or continuous, the application of traditional Reinforcement Learning methods based on lookup tables (a table that stores value functions for a small or discrete state space) is no longer possible, since the state space

becomes too large. This problem is known as the *curse of dimensionality* [3]. In order to bypass it, some form of generalization must be incorporated into how the states are represented.

On the other hand, most neuro-fuzzy systems [4-6] present limitations with regard to the number of inputs allowed and/or to the limited (or nonexistent) form to create their own structure and rules [7-8].

Thus, the new Reinforcement Learning-Hierarchical Neuro-Fuzzy BSP and Politree models apply a recursive partitioning methodology (already explored with excellent results in Souza [7-8]), which significantly reduces the limitations of the existing neuro-fuzzy systems.

The use of this partitioning method combined with Reinforcement Learning resulted in a new class of Neuro-Fuzzy System (NFS), called *Reinforcement Learning-Hierarchical Neuro-Fuzzy System* (RL-HNF). In addition to the ability to create its own structure, this model autonomously learns the actions that must be taken by an agent.

In this manner, the new RL-HNF model presents the following important characteristics: it automatically learns its structure; it performs self-adjustment of the parameters associated with the structure; it is capable of learning an action to be taken when the agent is in a given state of the environment; it is able to deal with a greater number of inputs when compared with the traditional neuro-fuzzy systems; and it automatically generates linguistic rules. These characteristics represent an important differential in relation to the existing intelligent agent learning systems.

The paper is organized as follows: section 2 contains a brief description of BSP and Politree partitionings. Section 3 introduces the RL-HNFB and RL-HNFP models, describing their basic cells, architectures and learning methods. Section 4 presents the results obtained with the case study: cart-centering problem [9]. The main objective of using this case study was to evaluate how well the proposed models would adapt to changes in the domain of the input variables without having to retrain the systems. Lastly, section 5 presents the conclusions.

## 2 Hierarchical Partitioning

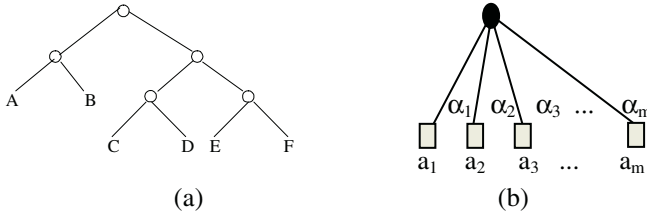
The partitioning process of the input/output space has great influence on the neuro-fuzzy system performance in relation to its desirable features (accuracy, generalization, automatic generation of rules, etc).

The BSP – Binary Space Partitioning [7-8] divides the space successively, in a recursive way, in two regions. Figure 1a shows an example of a two dimensional input partitioned using the BSP method.

The Politree partitioning is a generalization of the quadtree [8]. In this partitioning the subdivision of the n-dimensional space is accomplished in  $m=2^n$  subdivision. The Politree partitioning can be represented by a tree structure. Figure 1b presents the generic politree partitioning (with n inputs).

Hierarchical partitioning is flexible and minimizes the exponential rule growth problem, since it creates new rules locally according to the learning process. This type of partitioning is considered recursive because it makes use of a recursive process to generate partitions. In this manner, the resulting models have a hierarchy in their structure and consequently, hierarchical rules.





**Fig. 1.** (a) BSP Tree representing the BSP partitioning (b) generic tree representation of the politree partitioning with  $n$  inputs, where  $m = 2^n$

### 3 Reinforcement Learning – Hierarchical Neuro-fuzzy BSP (RL-HNFB) and Politree (RL-HNFP) Models

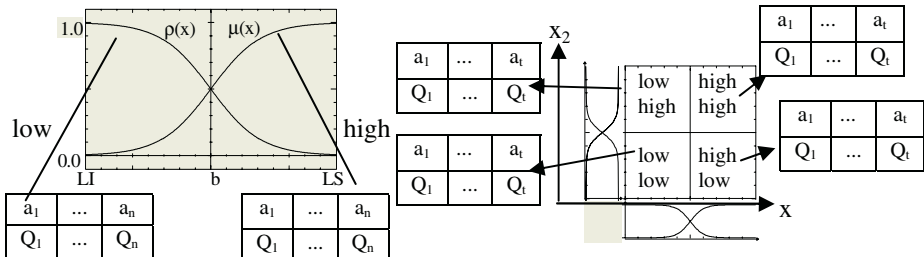
The RL-HNFB and RL-HNFP models are composed of one or various standard cells called RL-neuro-fuzzy BSP (RL-NFB) and RL-neuro-fuzzy-Politree (RLNFP), respectively. These cells are laid out in a hierarchical structure, according to the partition being used. The following sections describe the basic cells, the hierarchical structures and the learning algorithm.

#### 3.1 Reinforcement Learning Neuro-fuzzy BSP and Politree Cells

An RL-NFB cell is a mini-neuro-fuzzy system that performs binary partitioning of a given space in accordance with the membership functions described in figure 2(a). In the same way, an RL-NFP cell is a mini-neuro-fuzzy system that performs  $2^n$  partitioning of a given input space, also using complementary membership functions in each input dimension. The RL-NFB and RL-NFP cells generate a precise (crisp) output after the defuzzification process [1][10].

The RL-NFB cell has **only one** input ( $x$ ) associated with it. For further details about RL-NFB cell, see [1].

The RL-NFP cell receives **all** the inputs that are being considered in the problem. For illustration purpose, figure 2(b) depicts a cell with two inputs –  $x_1$  and  $x_2$  - (Quadtree partitioning), providing a simpler representation than the  $n$ -dimensional



**Fig. 2.** Internal representation of the (a) RL-NFB and (b) RL-NFP cells

form of Politree. In figure 2(b) each partitioning is generated by the combination of two membership functions -  $\rho$  (*low*) and  $\mu$  (*high*) of each input variable.

The consequents of the cell's poli-partitions may be of the *singleton* type or the output of a stage of a previous level. Although the *singleton* consequent is simple, this consequent is not previously known because each *singleton* consequent is associated with an action that has not been defined a priori. Each poli-partition has a set of possible actions ( $a_1, a_2, \dots a_n$ ), as shown in figure 2, and each action is associated with a Q-value function. The Q-value is defined as being the sum of the expected values of the rewards obtained by the execution of action  $a$  in state  $s$ , in accordance with a policy  $\pi$ . For further details about RL theory, see [3].

The linguistic interpretation of the mapping implemented by the RL-NFP cell depicted in Figure 2(b) is given by the following set of rules:

- rule<sub>1</sub>: If  $x_1 \in \rho_1$  and  $x_2 \in \rho_2$  then  $y = a_i$
- rule<sub>2</sub>: If  $x_1 \in \rho_1$  and  $x_2 \in \mu_2$  then  $y = a_j$
- rule<sub>3</sub>: If  $x_1 \in \mu_1$  and  $x_2 \in \rho_2$  then  $y = a_p$
- rule<sub>4</sub>: If  $x_1 \in \mu_1$  and  $x_2 \in \mu_2$  then  $y = a_q$

where  $\alpha_i$  is the firing level of rule  $i$  (poli-partition  $i$ ) and consequent  $a_i$  corresponds to one of the two possible consequents below:

**A *singleton*** (fuzzy singleton consequent, or zero-order Sugeno): the case where  $a_i = \text{constant}$ ;

**The output of a stage of a previous level:** the case where  $a_i = y_m$ , where  $y_m$  represents the output of a generic cell 'm', whose value is calculated by equation (1) below. Since the *high* ( $\mu$ ) and *low* ( $\rho$ ) membership functions are complementary, the defuzzification process is simplified, with the denominator of the equation on the left being equal to 1 for any values of inputs 'x', where  $\alpha_i$  corresponds to the firing level of partition  $i$ .

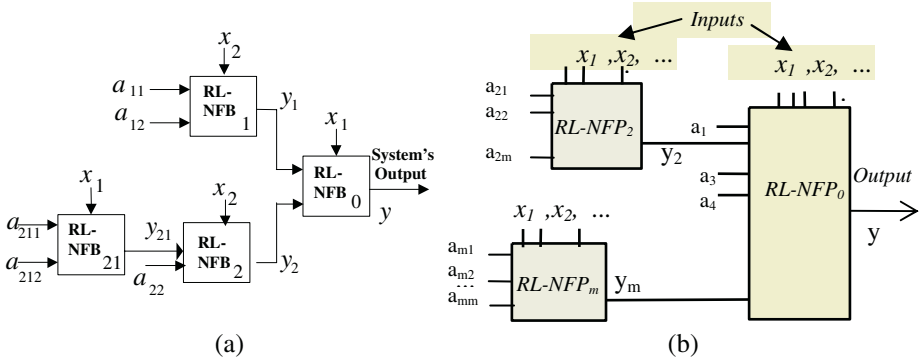
$$y = \left( \frac{\sum_{i=1}^{2^n} \alpha_i \times a_i}{\sum_{i=1}^{2^n} \alpha_i} \right) \qquad y = \sum_{i=1}^{2^n} \alpha_i \cdot a_i \qquad (1)$$

### 3.2 RL-HNFB and RL-HNFP Architectures

RL-HNFB models can be created based on the interconnection of the basic cells. The cells form a hierarchical structure that results in the rules that compose the agent's reasoning. Figure 3 exemplifies the RL-HNFB and RL-HNFP architectures. The output of the system in figure 3(a) (defuzzification) is given by equation (2). For further details about RL-NFB architecture, see [1].

In the architecture presented in figure 3(b), the poli-partitions 1, 3, 4, ...  $m-1$  have not been subdivided, having as consequents of its rules the values  $a_1, a_3, a_4, \dots a_{m-1}$ , respectively. On the other hand, poli-partitions 2 and  $m$  have been subdivided; so the consequents of its rules are the outputs ( $y_2$  and  $y_m$ ) of subsystems 2 and  $m$ , respectively. On its turn, these subsystems have, as consequent, the values  $a_{21}, a_{22}, \dots, a_{2m}$ , and  $a_{m1}, a_{m2}, \dots, a_{mm}$ , respectively. Each ' $a_i$ ' corresponds to a consequent of zero-order Sugeno (*singleton*), representing the action that will be identified (between the possible actions), through reinforcement learning, as being the most favorable for a

certain state of the environment. The output of the system depicted in figure 3(b) (defuzzification) is given by equation (3). Again, in these equations,  $\alpha_i$  corresponds to the firing level of partition  $i$  and  $a_i$  is the singleton consequent of the rule associated with partition  $i$ .



**Fig. 3.** Example of an architecture of the (a) RL-HNFB and (b) RL-HNFP models

$$y = \rho_0(x_1) \cdot [\rho_1(x_2) \cdot a_{11} + \mu_1(x_2) \cdot a_{12}] + \mu_0(x_1) \cdot \{ \rho_2(x_2) \cdot [\rho_{21}(x_1) \cdot a_{211} + \mu_{21}(x_1) \cdot a_{212}] + \mu_2(x_2) \cdot a_{22} \} \quad (2)$$

$$y = \alpha_1 \cdot a_1 + \alpha_2 \sum_{i=1}^{2^n} \alpha_{2i} \cdot a_{2i} + \alpha_3 \cdot a_3 + \alpha_4 \cdot a_4 + \dots + \alpha_m \sum_{i=1}^{2^n} \alpha_{mi} \cdot a_{mi} \quad (3)$$

### 3.3 RL-HNFB and RL-HNFP Learning Algorithm

The learning process starts with the definition of the relevant inputs for the system/environment where the agent is and the sets of actions it may use in order to achieve its objectives. The agent must run many cycles to ensure learning in the system/environment where it is. A cycle is defined as the number of steps the agent takes in the environment, which extends from the point he is initiated to the target point.

The RL-HNFB and RL-HNFP models employ the same learning algorithm. Each partition chooses an action from its set of actions; the resultant action is calculated by the defuzzification process (see equation (1) and represents the action that will be executed by the agents' actuators. After the resultant action is carried out, the environment is read once again. This reading enables calculation of the environment reinforcement value that will be used to evaluate the action taken by the agent. The reinforcement is calculated for each partition of all active cells, by means of its participation in the resulting action. Thus, the environment reinforcement calculated by the evaluation function is backpropagated from the root-cell to the leaf-cells. Next, the Q-values associated to the actions that have contributed to the resulting action are updated, based on the SARSA algorithm [3]. More details can be found in [10].

The RL\_HNFB and RL\_HNFP models have been evaluated in a control application. This case study is presented in the next section.

### 4 Case Studies

The case studied, the *cart-centering problem* (parking the car) [9], is generally used as a benchmark of the area of evolutionary programming, where the force that is applied to the car is of the “bang bang” type [9]. This problem was used mainly for the purpose of evaluating how well the RL-HNFB and RL-HNFP models would adapt to changes in the input variable domain without having to undergo a new training phase.

The problem consists of parking, in the centre of a one-dimensional environment, a car with mass  $m$  that moves along this environment due to an applied force  $F$ . The input variables are the position ( $x$ ) of the car, and its velocity ( $v$ ), where the  $\tau$  parameter represents the time unit. The objective is to park the car in position  $x = 0$  with velocity  $v = 0$ . The equations of motion are:

$$x_{t+\tau} = x_t + \tau.v_t \quad \text{and} \quad v_{t+\tau} = v_t + \tau.F_t / m \tag{4}$$

The global reinforcement is calculated by equation 5 below:

If ( $x > 0$  and  $v < 0$ ) or ( $x < 0$  and  $v > 0$ )

$$R_{global} = k_1 e^{-(\text{distance\_objective})} + k_2 e^{(\text{velocity}^2)} \tag{5}$$

Else  $R_{global} = 0$

The evaluation function increases as the car gets closer to the centre of the environment with velocity zero. The  $k_1$  and  $k_2$  coefficients are constants greater than 1 used for adapting the reinforcement values to the model’s structure. The values used for time unit and mass were  $\tau = 0.02$  and  $m = 2.0$ .

The stopping criterion is achieved when the difference between the velocity and the position value in relation to the objective ( $x=0$  and  $v=0$ ) is smaller than 5% of the universe of discourse of the position and velocity inputs. Table 1 shows the average of the results obtained in 5 experiments for each configuration.

**Table 1.** Configurations and results of the RL-HNFB and RL-HNFP models applied to the cart-centering problem

| No.                   | Position Limits | Velocity Limits | Size of the Structure | Average steps learning phase |
|-----------------------|-----------------|-----------------|-----------------------|------------------------------|
| RL-HNFB <sub>1</sub>  | 10              | 10              | 195 cells             | 424                          |
| RL- HNFB <sub>2</sub> | 3               | 3               | 340 cells             | 166                          |
| RL-HNFP <sub>1</sub>  | 10              | 10              | 140 cells             | 221                          |
| RL-HNFP <sub>2</sub>  | 3               | 3               | 251 cells             | 145                          |

The columns position and velocity limits refer to the limits imposed to the (position and velocity) state variables during learning and testing. The actions used in these experiments are:  $F1 = \{-150, -75, -50, -30, -20, -10, -5, 0, 5, 10, 20, 30, 50, 75, 150\}$ . The size of the structure column shows the average of the number of cells at the

end of each experiment and the last column shows the average steps during the learning phase. The number of cycles was fixed at 1000. At each cycle, the car's starting points were  $x = -3$  or  $x = 3$ .

As can be observed from Table 1, the RL-HNFP structure is smaller because each cell receives both input variables, while in the case of the RL-HNFB model, a different input variable is applied at each level of the BSP tree.

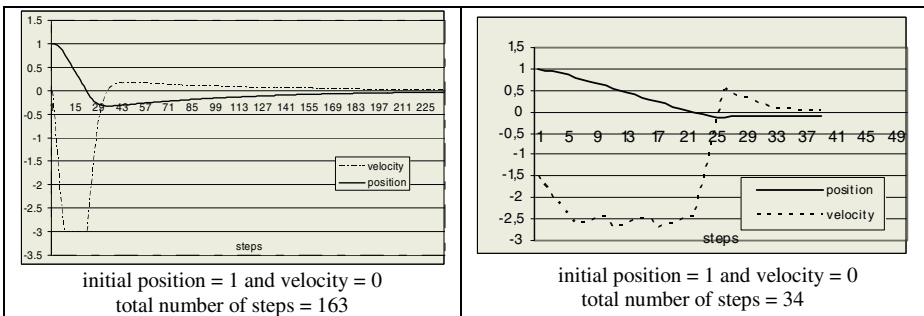
Table 2 below presents the results obtained for one of the 5 experiments carried out at each configuration shown in Table 1 when the car starts out at points (-2, -1, 1, 2) which were not used in the learning phase.

**Table 2.** Testing results of the proposed models applied to the cart-centering problem

| Configuration        | Initial Position        |     |     |
|----------------------|-------------------------|-----|-----|
|                      | 3                       | 2   | 1   |
|                      | Average number of steps |     |     |
| RL-HNFB <sub>1</sub> | 387                     | 198 | 141 |
| RL-HNFB <sub>2</sub> | 122                     | 80  | 110 |
| RL-NFHP <sub>1</sub> | 190                     | 166 | 99  |
| RL-NFHP <sub>2</sub> | 96                      | 124 | 68  |

In the first configuration of each model, the results show that the broader the position and velocity limits are (in this case equal to |10|), the more difficult it is to learn. In these cases a small oscillation occurs around the central point. What actually happens is that the final velocity is very small but, after some time, it tends to move the car out of the convergence area, resulting in a peak of velocity in the opposite direction to correct the car's position. In the second configurations, the results show that there were fewer oscillations when the position and velocity limits were lowered to |3|.

A different test was carried out with configuration 2 of RL-HNFB and RL-HNFP in which the environment limits were expanded to 10 and the velocity limits were



**Fig. 4.** Test results with a larger environment (-10 to 10) for the cart centering problem for the (a) RL-HNFB<sub>2</sub> configuration and (b) RL-HNFP<sub>2</sub> configuration

maintained at 3. The results obtained are presented in the graphs of figure 4. In this new test, although the environment is three times larger than the environment used for learning, the car is able to achieve its objective when it starts out from the following points: the environment limit (10 and -10); halfway in the environment (-5 and 5) and points that are close to the objective (-1 and 1) (Figure 4). This behaviour was made possible because the input variables were normalized, enabling the autonomous learning process to adapt to changes in the environment without requiring a new learning process.

## 5 Conclusions

The objective of this paper was to introduce a new class of neuro-fuzzy models which aims to improve the weak points of conventional neuro-fuzzy systems and reinforcement learning models. The models RL-HNFB and RL-NFHP belong to a new class of neuro-fuzzy systems called Reinforcement Learning Hierarchical Neuro-Fuzzy Systems.

These models are able to create their own structures and allow the extraction of knowledge in a fuzzy rule-base format.

The RL-NFHB and RL-HNFP models were able to create and expand the structure of rules without any prior knowledge (fuzzy rules or sets); extract knowledge from the agent's direct interaction with large and/or continuous environments (through reinforcement learning), in order to learn which actions are to be carried out; and produce interpretable fuzzy rules, which compose the agent's intelligence to achieve his goal(s). The agent was able to generalize its actions, showing adequate behaviour when the agent was in states whose actions had not been specifically learned. This capacity increases the agent's autonomy. Normalizing the inputs also enabled the model to adequately respond to changes in the limits of input variables. This characteristic was also very interesting, because it further increases the autonomy desired for this agent.

## References

1. Vellasco, M., Pacheco, M., Figueiredo, K.: Hierarchical Neuro-Fuzzy Systems, 7th Int. Work Conf. on Art. and Natural Neural Networks. IWANN2003 Part I. (2003) 126-135
2. Jouffe, L.: Fuzzy Inference System Learning by Reinforcement Methods, IEEE Trans. on SMCs-C 28/3 (1998) 338-355
3. Sutton, R.S., Barto, A.G: Reinforcement Learning: An Introduction, (1998) MIT Press.
4. Jang, J.S.R.:ANFIS: Adaptive-network-based fuzzy inference system. IEEE Trans. on SMC 23 No 3 (1993) 665-685
5. Kruse, R., Nauck, D.: NEFCLASS-A neuro-fuzzy approach for the classification of data. Proc. of the 1995 ACM Symposium on Applied Computing, (1995) 26-28.
6. Vuorimaa, P.: Fuzzy self-organizing map. Fuzzy Sets and Systems, No.66 (1994) 223-231.
7. Vellasco, M., Pacheco M., Ribeiro Neto, L.S., Souza, F.J.: Electric load forecasting: evaluating the novel hierarchical neuro-fuzzy BSP model. Electrical Power and Energy Systems Elsevier Ltd 26 (2004) 131-142

8. Souza, F., Vellasco, M., Pacheco, M.: Hierarchical Neuro-Fuzzy QuadTree Models. *Fuzzy Sets & Systems* vol 130/2 (2002) 189-205.
9. Koza, J.R.: *Genetic Programming: On the programming of computers by means of natural selection*, Cambridge, MA, MIT Press, (1992)
10. Figueiredo, K., Vellasco, M., Pacheco, M., Souza, F.: Reinforcement Learning Hierarchical Neuro-Fuzzy Polintree Model for Control of Autonomous Agents. *Fourth Int. Conf. on Hybrid Intelligent Systems*, Japan (2004).

# Interfacing with Patterned in Vitro Neural Networks by Means of Hybrid Glass-Elastomer Neurovectors: Progress on Neuron Placement, Neurite Outgrowth and Biopotential Measurements

Enric Claverol-Tinturé, Xavier Rosell, and Joan Cabestany

Dept. of Electronics, Technical University of Catalonia, Jordi Girona 1-3,  
08034 Barcelona, Spain  
{claverol, jrosell, cabestan}@eel.upc.es

**Abstract.** In order to extract learning algorithms from living neural aggregates it would be advantageous to achieve one-to-one neuron-electrode interfacing with in vitro networks. Towards this goal, we have developed a hybrid glass-elastomer technology, which allows topology specification in small networks (of the order of 10 neurons) and recording of extracellular potentials from individual neurites grown through microfluidic channels. Here we report on progress towards adhesion-free placement of cells within microwells, promotion of neurite growth and recording of intra-channel extracellular spikes.

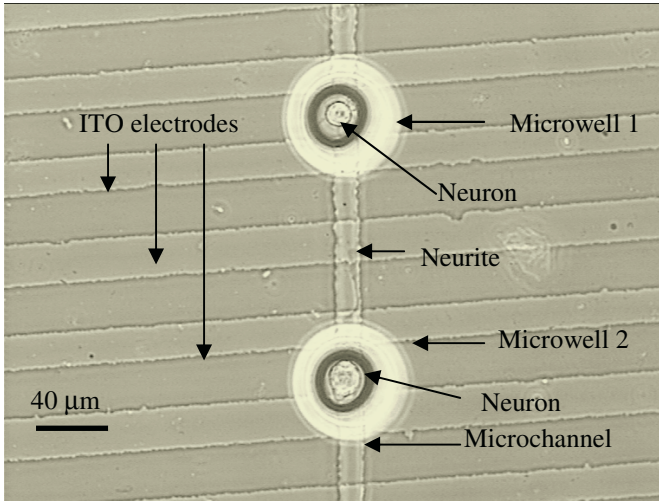
## 1 Introduction

We have recently reported on the development of hybrid glass-elastomer multielectrode devices for in vitro neural network patterning and recording of extracellular potentials [1]. The goal is the achievement of one-to-one neuron-electrode interfacing with small networks (approx. 10 neurons) in order to extract neural information processing algorithms.

In our devices, network topology specification is realized by microfluidic channels capable of guiding neurite growth. Extracellular potentials are recorded by substrate embedded ITO electrodes over which neurites extend. As transmembrane channels (mostly at the axon-hillock) sink current to generate an action potential, the extracellular flow is confined within microchannels and the associated potentials recorded by intra-channel electrodes.

The fabrication process has been reported elsewhere [1]. Briefly, the approach is based on the assembly of a sandwich structure; polymeric (PDMS) films (100  $\mu\text{m}$  in thickness) including microchannels for neurite guidance and through holes for neuronal soma confinement are fabricated with a combination of soft-lithography techniques [2] and micro-hole punching [1] and overlaid on standard multielectrode arrays [3]. Microchannels with a cross-section of 15x5  $\mu\text{m}$  allow neurite outgrowth while confining cell bodies within microwells. Inter-well distances of 100-200  $\mu\text{m}$  have proved adequate to obtain spikes in the order of 60-80  $\mu\text{V}$ , resulting in signal-to-noise ratios of 3-4, for typical electrode/electronics noise levels.





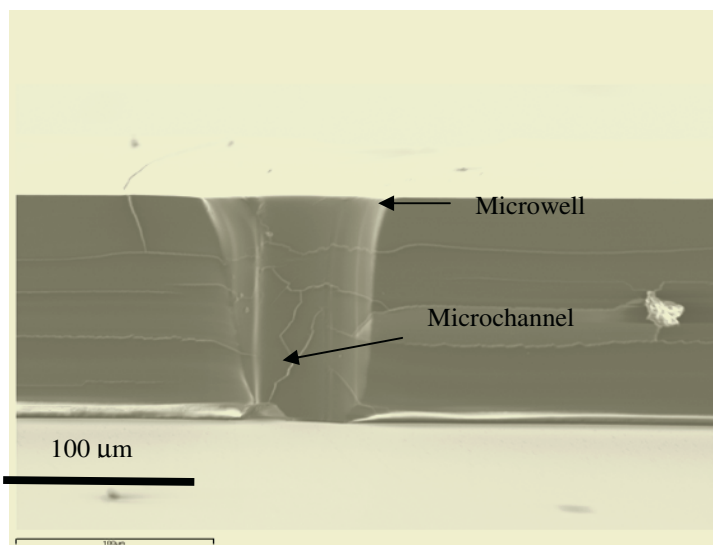
**Fig. 1.** Top view of two *microwells* in a neurovector with one *neuron* plated per *microwell*. The transparent 100  $\mu\text{m}$ -thick PDMS film, containing through holes interconnected by a neurite guiding microchannel (cross-section 15x5  $\mu\text{m}$ ), is overlaid on a glass substrate with embedded 40  $\mu\text{m}$ -wide ITO electrodes

Fig. 1 shows, in top view, a two-well neurovector, a device for interfacing with uni-dimensional aggregates, i.e. a set of neurons patterned to form in vitro chains. Fig. 2 shows a SEM image of a microwell. The conic geometry of the microwell facilitates neuron placement. The entrance of a microchannel is also visible at the bottom of the well.

Placement of individual *Helix aspersa* neurons in microwells and sprouting of neurites of sufficient length to transverse the microchannels are necessary conditions for network formation. The mollusk neurons used for our work are typically in the range 20-50  $\mu\text{m}$  in diameter. Selection, transport and placement within the neurovector must be compatible with cell viability. Here we report on progress towards improved neuron placement and network formation within neurovector structures.

## 2 Cell Placement and Neurite Sprouting

Placement of individual neurons within microwells is achieved by selection of healthy cells from dispersed brains and subsequent transport by suction/expel sequences with fire-polished borosilicate capillaries. Sonication of the capillaries in acetone/isopropanol for 10 minutes and thorough rinsing in Milli-Q water greatly reduces the probability of damage to neurons during transport. The neurovectors were designed for easy cell placement. Neurons were to be dropped on the PDMS film and rolled on the surface, by sequential puffs of culture medium, to fall within wells. The conic shape of the wells facilitates the latest phases of the process.



**Fig. 2.** SEM image of a cross-section through a *microwell* and *microchannel*

For this procedure to be successful, an anti-fouling coating must be used, since PDMS is known to facilitate cell adhesion, possibly due to hydrophobic-hydrophobic interactions. We have experimented with a number of film treatments.

Incubation of the PDMS structures in 1% bovine serum albumin (BSA) in 1% PBS for up to 2 hours markedly decreased adhesion but was insufficient to avoid adhesion to the walls of the microwells. Interestingly, on occasions, neurons adhered to the walls of the microwells and grew neurites that would extend to the upper end of the well and over the surface of the PDMS film for hundreds of microns (see Fig. 3).

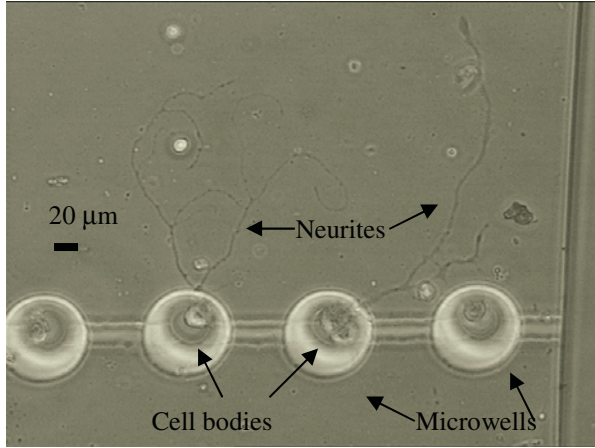
Adhesion was successfully minimized with 24h BSA treatment in which case nearly 100% of the cells reached the substrate at the lower end of the microwells. Fig.1 shows two cells growing in BSA-coated PDMS structures. However, the use of BSA as a coating on the PDMS film had a negative effect on the probability of sprouting. While 90% of the cells sprouted when cultured in controls without BSA, only 30% grew neurites when plated within BSA-treated structures.

To a first approximation, and assuming independence, chains with  $N$  adjacent sprouted neurons would occur with a probability of approximately  $0.3^N$  which renders small size networks feasible but larger aggregates time consuming.

It is hypothesized that the effect of BSA on cell growth can relate to its leach to the culture after cell placement. Since cell adhesion and growth requires charge-rich substrates, poly-l-lysine is often used to coat surfaces intended to promote cell adhesion and neurite growth. BSA is commonly used as a blocker of unspecific binding. Therefore, BSA could act as a blocker of neurite growth through binding to substrate-bound poly-l-lysine.

To address this problem, PDMS was used uncoated whereas brain dissociation was carried out in 1% BSA to promote shielding of the cells by a BSA rich, hydrophilic and anti-fouling shell. After a 30 minutes incubation, BSA coated neurons were

transferred to PDMS. Culture medium was changed to remove traces of BSA and avoid putative blocking of poly-L-lysine.



**Fig. 3.** Phase-contrast image showing *neurite* outgrowth over the surface of the PDMS film upon adhesion of neurons on the walls of *microwells*

Preliminary data show that adhesion is reduced, as observed with overnight BSA treatment of PDMS. Experiments on the probability of neurite outgrowth are underway.

We have also experimented with Pluronic P105 (BASF, Germany) an amphiphilic polymer with one hydrophobic polypropylene oxide (PPO) and two hydrophilic polyethylene oxide (PEO) blocks. The hydrophobic segment is adsorbed onto the hydrophobic surface of untreated PDMS while the hydrophilic ends remain accessible on the surface and decrease protein and cell adhesion. Treatments with 0.1% w/v Pluronic of up to 24h decreased adhesion on the surface but were not sufficient to avoid adhesion on the microwell walls in 70% of the cells plated. Moreover, Pluronic P105 appeared toxic to mollusk neurons after 96 h in culture.

Pluronic F68, a PEO-PPO-PEO polymer with higher hydrophilic to hydrophobic chain ratios as compared to P105, has been proven non-toxic with insect cell cultures and is currently under test as a PDMS anti-fouling treatment.



**Fig. 4.** Extracellular potentials recorded from the lumen a microchannel threaded by a neurite. Electrodes at different locations record spikes correlated in time but with different shapes/signs

### 3 Extracellular Potentials

Fig. 4 shows typical extracellular potentials recorded from a single microchannel-guided neuron by parallel electrodes as shown in Fig. 1. The location of the electrodes and the grounds of the circuit result in correlated spikes recorded with various characteristic shapes by different electrodes. Signal magnitudes with respect to distant grounds of 60  $\mu\text{V}$  are typical.

The axon-hillock and the passive neuronal segments result in a current dipole, i.e. hillocks sink current while dendrites source it during the rising phase of the action potential whereas the pattern is reversed during the repolarization phase. The voltage drop along the channel associated with this dipole can be measured at different points along the microchannel. Details on the instrumentation will be described in detail in a future report.

### 4 Conclusions

Preliminary data suggest that BSA anti-fouling coatings facilitate cell placement within microwells. However, if PDMS is treated overnight, the probability of sprouting is markedly lower within neurovectors, possibly due to BSA leach and subsequent block of growth cone adhesion and extension. Coating of cells with BSA and subsequent elimination of remaining BSA traces, as opposite to treatment of PDMS structures amenable to leach, is being explored.

Moreover, the use of conditioned medium and co-culture with entire brains has been shown to promote sprouting [4]. Observation of occasional extensive sprouting in proximal neurons suggest that sufficient neurite growth could be promoted by adequate partners. Random selection of neurons from dispersed tissue may, therefore, be inadequate to support extensive outgrowth. Research will focus on the use of cells from individual ganglia or, preferably, from identified neurons with confirmed connections in vivo in order to increase the probability of sprouting and connection between adjacent cells. Overall these strategies could compensate for the effects of BSA on neurite outgrowth.

Despite decreased sprouting probabilities due to BSA treatments, neurite extension along microchannels already makes possible the recording of extracellular spikes with the current version of the experimental setup.

### References

1. Claverol-Tinturé, E., Ghirardi M., Fiumara F., Rosell X., Cabestany J. Multielectrode arrays with elastomeric microstructures for neuronal patterning towards interfacing with uni-dimensional neuronal networks. Submitted to *J. Neural Eng.* (2005)
2. Xia, Y., Whitesides, G.M. Soft lithography. *Ann. Rev. of Materials Science* (1998) 28(1):153-184
3. Pine, J. Recording action potentials from cultured neurons with extracellular microcircuit electrodes. *J Neurosci Methods*. (1980) Feb;2(1):19-31.
4. Tiwari, S.K., Woodruff M.L. Helix aspersa neurons maintain vigorous electrical activity when co-cultured with intact *H. aspersa* ganglia. *Comp. Biochem. Physiol.* (1992) 101C(1):163-174.

# Using Kolmogorov Inspired Gates for Low Power Nanoelectronics

Valeriu Beiu<sup>1</sup>, Artur Zawadski<sup>1</sup>, Răzvan Andonie<sup>2</sup>, and Snorre Aunet<sup>3</sup>

<sup>1</sup> School of Electrical Engineer and Computer Science, Washington State University,  
Pullman, Washington 99164-2752, USA  
vbeiu@eecs.wsu.edu

<sup>2</sup> Department of Computer Science, Central Washington University,  
Ellensburg, Washington 98926, USA  
andonie@cwu.edu

<sup>3</sup> Department of Informatics, University of Oslo, Oslo, Norway  
sa@ifi.uio.no

**Abstract.** Based on explicit numerical constructions for Kolmogorov's superpositions (KS) linear *size* circuits are possible. Because classical Boolean as well as threshold logic implementations require exponential *size* in the worst case, it follows that *size*-optimal solutions for arbitrary Boolean functions (BFs) should rely (at least partly) on KS. In this paper, we will present previous theoretical results while examining the particular case of 3-input BFs in detail. This shows that there is still room for improvement on the synthesis of BFs. Such *size* reductions (which can be achieved systematically) could help alleviate the challenging power consumption problem, and advocate for the design of Kolmogorov-inspired gates, as well as for the development of the theory, the algorithms, and the CAD tools that would allow taking advantage of such optimal combinations of different logic styles.

## 1 Introduction

The problem we are going to discuss in this paper is that of the *size* of circuits implementing arbitrary Boolean functions (BFs)  $f: \{0,1\}^n \rightarrow \{0,1\}$ . This is a well-studied problem, and its solution translates into optimal Boolean circuits—leading to minimum *size*, *area*, and most importantly *power* dissipation. The optimal solution depends on the set of gates used to implement BFs. In particular it is well known that implementing arbitrary BFs using classical Boolean gates (BGs), requires  $O(2^n)$  *size* circuits (for PARITY). By using threshold logic gates (TLGs) the bound is smaller  $O(2^{n/2})$  [1, 2, 3], revealing an exponential gap. The bounds for TLGs have been obtained for (small) constant *depths*, as TLGs are perceptrons, hence approximating shallow neural structures. It is possible to reduce the *size* of TL circuits for non-constant *depths*, but results are known only for particular BFs [4]. The difficult problems when trying to implement TLGs are: the large *fan-ins* and the precision of the *weights* [5]. These are theoretically related [3, 6, 7] as:

$$2^{(\Delta-1)/2} < \text{weight} < (\Delta+1)^{(\Delta+1)/2} / 2^\Delta \quad (1)$$

for any *fan-in*  $= \Delta > 3$ , i.e., precision can be as high as  $O(n \log n)$  bits per *weight*.

Better results with respect to *size* can be obtained only if the elementary gates are more powerful, e.g., multi-threshold TLGs [8, 9] or multiple valued logic gates [10, 11]. In this paper we will take such an approach, and advocate for using Kolmogorov's superpositions (KS) theorem. This shows that there is a solution requiring only  $2n + 1$  functions of one variable which can approximate any  $f: [0,1]^n \rightarrow \mathbb{R}$ . Such a linear *size* solution appears very promising and could help alleviate the *challenging power consumption problem* [12, 13].

The paper will start by presenting Hilbert's thirteenth problem and KS in Section 2. Motivations for using Kolmogorov's theorem for BFs will be given in Section 3, while the particular case of three input BFs will be analyzed and discussed in Section 4. Conclusions and future directions of research are ending the paper.

## 2 Kolmogorov's Superposition

In 1900 the German mathematician David Hilbert challenged mathematicians to solve 23 fundamental problems [14]. In the thirteenth of these problems he conjectured that *there exist continuous multivariate functions that cannot be decomposed as a finite superposition of continuous functions of fewer variables*.

Hilbert's thirteenth problem was refuted by Kolmogorov [15] and Arnol'd [16]. In 1957 Kolmogorov also proved a general theorem where the functions in the decomposition are one dimensional, known as *Kolmogorov's superpositions (KS) theorem* [17]. Formally, for each integer  $n \geq 2$ , there exist  $2n + 1$  functions  $\psi_q$ ,  $q = 0, \dots, 2n$  in  $C([0, 1]^n)$  of the form  $\psi_q(\mathbf{x}) = \sum_{p=1}^n \psi_{pq}(x_p)$ ,  $\mathbf{x} = (x_1, \dots, x_n) \in [0, 1]^n$ ,  $\psi_q \in C([0, 1]^n)$ ,  $p = 1, \dots, n$ , such that each  $f$  in  $C([0, 1]^n)$  is representable as  $f(\mathbf{x}) = \sum_{q=0}^{2n} \phi_q(\psi_q(\mathbf{x}))$ ,  $\phi_q \in C(\mathbb{R})$ . Here  $C(X)$  denotes the space of continuous real-valued functions on the topological space  $X$ . This means that for every real-valued continuous function  $f: [0, 1]^n \rightarrow \mathbb{R}$  there are continuous functions  $\phi_q$  (depending on  $f$ ) such that

$$f(x_1, \dots, x_n) = \sum_{q=0}^{2n} \phi_q \left[ \sum_{p=1}^n \psi_{pq}(x_p) \right], \quad (2)$$

where  $\psi_{pq}$  are universal for the given dimension  $n$ , i.e., independent of  $f$  (see Fig. 1.a).

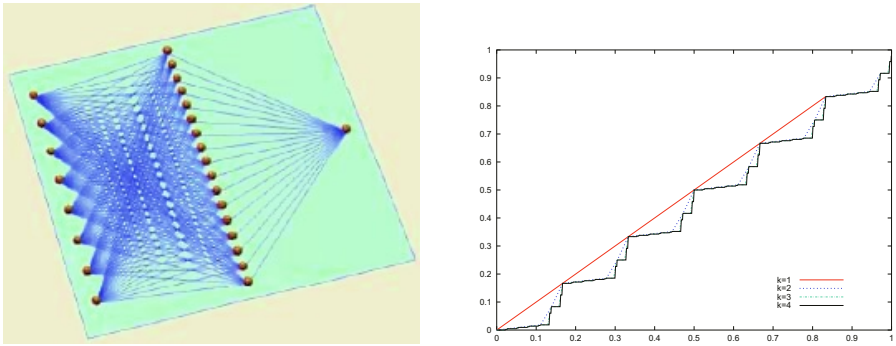
Such results were refined by Sprecher [18], and by Lorenz [19]. Sprecher ([18, 20]) showed that for a given  $\delta > 0$  there is a rational  $a$ ,  $0 < a < \delta$ , such as the unknown approximation mapping could be generated by replacing the set of functions  $\psi_{pq}$  by  $\lambda_p \psi_q$ , where  $\lambda_p$  are independent constants, and  $\psi_q$  are monotonic increasing functions:

$$f(x_1, \dots, x_n) = \sum_{q=0}^{2n} \phi_q \left[ \sum_{p=1}^n \lambda_p \psi(x_p + aq) \right]. \quad (3)$$

The function  $\psi$  and the constants  $\lambda_p$  are independent both of the function  $f$ , and of the dimension  $n$ . Another improvement is due to Lorenz in 1966 [19], namely that the function  $\phi_q$  could be replaced by only one function  $\phi$ .

Probably, the first suggestion of applying KS to circuits, together with a construction approximating  $\psi$ , was presented by de Figueiredo [21]. The construction of  $\psi$  was obtained as the uniform limit of a sequence  $\psi_r$ ,  $r \rightarrow \infty$ , where  $\psi_r$  was a continuous nondecreasing piecewise linear function. This provided a procedure for the syn-

thesis of (nonlinear) circuits. Figueiredo’s experience revealed that KS representation “is fragile and sensitive to perturbations.”



**Fig. 1.** (a) Direct implementation of  $f: [0,1]^n \rightarrow \mathbb{R}$  based on KS ( $n$  inputs and  $2n+1$  hidden nodes). (b) Sprecher’s function  $\psi$  for  $n = 2$  and  $\gamma = 10$  ( $k$  is the precision in number of digits in base  $\gamma$ )

Hecht-Nielsen realized that Sprecher’s version of KS could be interpreted as a multiplayer feed forward neural network (NN), showing that NNs are universal approximators [22]. As the functions  $\psi$  and  $\phi$  are highly non-smooth, the first hypothesis was that KS was not relevant for NNs [23]. It was later shown that it is possible to adapt KS to NNs [24, 25]: the functions  $\psi$  and  $\phi$  can be approximated by *staircase-like functions*. The learning problem was reduced to the learning of the *weights* from the second hidden layer to the output, which can be solved by linear regression [26, 27].

Constructive aspects of KS were detailed in [28–33]. For given integers  $n \geq 2$  and  $\gamma \geq 2n+2$ , the function  $\psi$  is defined such that for each integer  $k$

$$\psi(\sum_{r=1}^k i_r \gamma^{-r}) = \sum_{r=1}^k \tilde{i}_r 2^{-m_r} \gamma^{-(n^{r-m_r}-1)/(n-1)} \tag{4}$$

where  $\tilde{i}_r := i_r - (\gamma - 2)\langle i_r \rangle$ ,  $m_r := \langle i_r \rangle(1 + \sum_{s=1}^{r-1} \prod_{t=s}^{r-1} [i_t])$ , for all  $r = 1, \dots, k$ ;  $\langle i_1 \rangle = 0$ ,  $[i_1] = 0$ ;  $\langle i_r \rangle = 0$  if  $i_r = 0, \dots, \gamma - 2$  and  $\langle i_r \rangle = 1$  if  $i_r = \gamma - 1$ ;  $[i_r] = 0$  if  $i_r = 0, \dots, \gamma - 3$  and  $[i_r] = 1$  if  $i_r = \gamma - 2, \gamma - 1$ ,  $r \geq 2$ . Taking  $\lambda_1 = 1$ ,  $\lambda_p = \sum_{r=1}^{\infty} \gamma^{-(p-1)(n^{r-1})/(n-1)}$  for  $p = 2, \dots, n$ , and  $a = 1/\gamma/(\gamma-1)$ , satisfies eq. (3). The fact that  $\gamma$  depends on  $n$ , makes it that the function  $\psi$  and the constants  $\lambda_p$  will also depend on  $n$ .

All recent simulations of KS [34–36] rely on staircase-like approximations (see Fig. 1.b), similar to [21] and advocated in [24, 25].

### 3 On Kolmogorov’s Superposition and Boolean Functions

KS was intended for approximating real functions, but it can be used for BFs also [37–39]. Due to technological limitations, one digit of precision ( $k = 1$ ) should be enough (remember that  $\gamma \geq 6$ ). This gives  $\psi(0.i_1) = 0.i_1$ , i.e. the identity function. This is an analog comparison  $x_p + qa$ . Unfortunately, KS has a large *fan-in*  $= 2n + 1$  and

double exponential *weights* (see eq. 4). For BFs, the precision is at most  $(2n + 2)^{-n}$  bits per *weight*. This is the same as the precision required by TLGs (see eq. 1). Because simple analog implementations are limited by technology to a few bits of precision, a solution would be to decompose a BF into simpler BFs of fewer variables [Remark: Previous theoretical results [40] have also shown that VLSI-optimal implementations of BFs using TLGs are obtained for small *fan-ins*  $\leq 6$ ]. With nanoelectronics raising many implementation concerns, it is to be expected that only very low-precision analog gates could be implemented reliably.

One aspect when considering nanoelectronics is the very challenging power consumption problem [12, 13]. In this context, the results from [41, 42], which show that *hybrid analog/digital implementations appear to be the most energy efficient ones*, should be of interest. Informally, the fact that analog circuits can do better than digital ones could be understood as the *size* is reduced, and the switching activity is also lower. The *power* and *area* tradeoffs utilization as a function of the output signal-to-noise ratio, show *area* and *power* advantages of analog at low signal-to-noise ratios (see Fig. 7(b) in [41], which shows total current reductions of 4–6x). Therefore, low precision analog might have an edge, and when optimally mixed with digital should lead to *power-efficient* solutions.

Based on such potential advantages with respect to *power*, we propose extending the existing compendium of gates by the addition of a few analog gates of low precision (hence small *fan-ins*): Kolmogorov-inspired gates (KGs).

## 4 On Boolean Functions of Three Inputs

We start with the 3D view of the probability of being in one of the four states of a Karnaugh map for two inputs BFs (Fig. 2.a). When an input switches, the associated voltage will have to go through analog values. The probabilities associated to such voltages are lower, the lowest one being that of having both inputs at  $V_{DD}/2$ . Between the highest and the lowest probabilities, separating contours have been drawn (Fig. 2.a.). From a probabilistic point of view, the lines forming a square seem to define the optimal separation. These are of the form  $x \pm y = c$ .

We have analyzed all the 256 BFs of three inputs, but shall present here only the 70 BFs having four “ones” and four “zeros” in their truth table. Fig. 3.a presents the optimized results when using only KGs for synthesizing the 70 BFs.

The next step was to analyze if an optimal solution can be obtained using BGs in combination with KGs. The results are presented in Fig. 3.b. It can be seen that the use of KGs lead to a simpler solution in only 8 out of the 70 cases (11%), while in 4 cases (6%) a combination of KGs and BGs is minimal.

The final step was to allow majority gates of three inputs (MAJ-3) in these optimizations. The results are presented in Fig. 4.a. Only 8 BFs require KGs, while MAJ-3 gates improve on 8 other BFs.

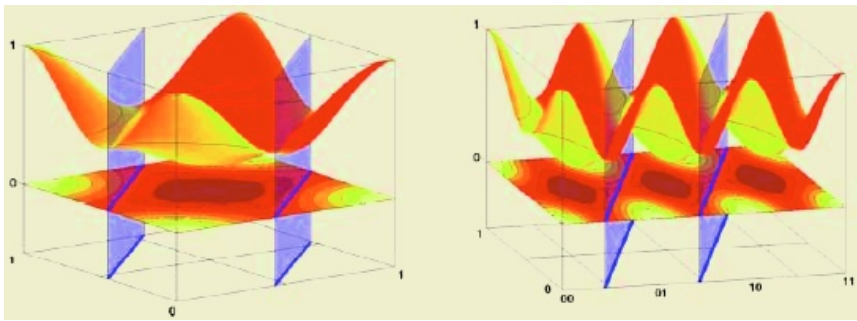
To get a better understanding, we have plotted in Fig. 4.b. the number of gates (*size*) in the hidden layer when using the three different types of logic: BGs, KGs, and MAJ (symmetric). Note that we did not properly use KS theorem, so in this sense



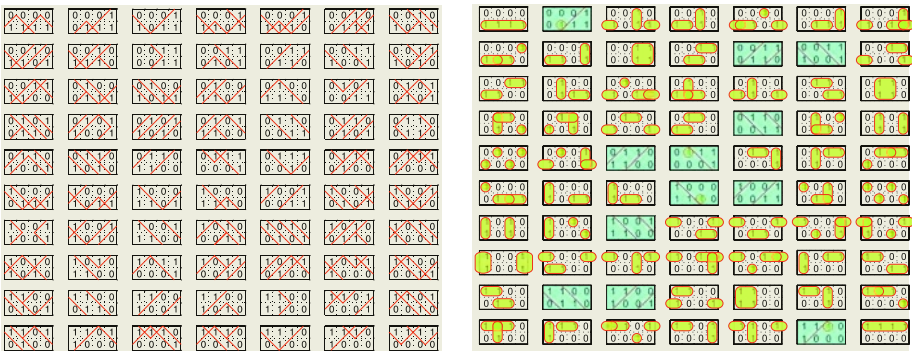
$2n+1$  is incorrect here. The results presented in Fig. 4.a. show that at most three gates are needed in the hidden layer, which is four gates less than KS ( $2*3+1=7$ ), and one less than the worst BF case ( $2^3/2=4$ ).

The results from this optimized first layer can be combined (in subsequent layers) by using: KGs, BGs, TLGs, leading to various mixtures of analog and digital solutions. Such an implementation will certainly reduce *size*, but our expectations are that the solution will also reduce *area*, while it is very likely that it will also reduce *power*.

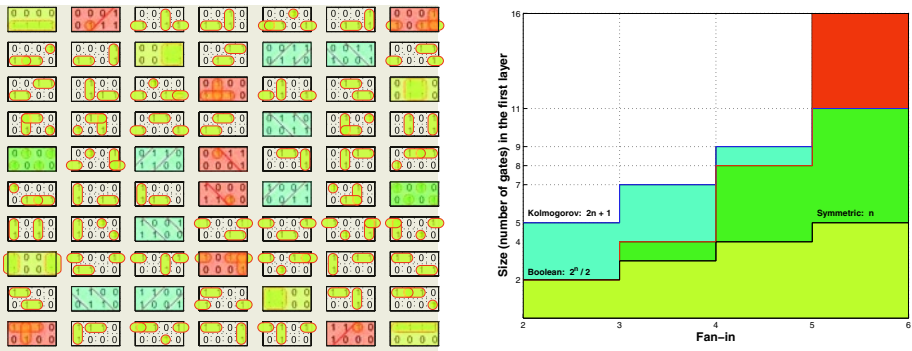
The results presented here suggest that the new low precision analog gates could be used in advanced library of gates. The main problem in this case would be that the synthesis CAD tools should be able to take advantage of them. The other immediate



**Fig. 2.** (a) 3D probability view on top of a generic Karnaugh map for BFs of two inputs. (b) 3D probability view on top of a generic Karnaugh map for BFs of three inputs



**Fig. 3.** (a) Using a KGs (with  $k = 1$ ) for implementing BFs of three inputs (for the 70 cases having four “ones” and four “zeros” in their truth table). (b) Optimization using both BGs and KGs for BFs of three inputs (for the 70 cases having four “ones” and four “zeros” in their truth table). KGs are able to improve in 12 out of the 70 cases



**Fig. 4.** (a) Combined synthesis for BFs of three inputs for all the cases having four “ones” and four “zeros” in their truth table (worst cases). The six yellow shaded Karnaugh maps are the trivial ones ( $x$ ,  $y$ ,  $z$ ,  $x_{\text{bar}}$ ,  $y_{\text{bar}}$ ,  $z_{\text{bar}}$ ), the eight red shaded Karnaugh maps are variations of MAJ-3, the eight blue shaded Karnaugh maps use KGs, while the two green-shaded Karnaugh maps are the XOR functions. (b) Size of the first layer when using KS ( $2n+1$ ), Boolean gates ( $2^n/2$ ), and TLGs for implementing symmetric functions ( $n$ )

application would be in the design of (novel) building blocks for (future CMOS, or beyond CMOS) FPGAs. These blocks should be custom designed using low precision analog gates like the ones suggested in this paper. This will reduce the power consumption. An advantage in this case is that, after the block(s) have been custom designed, the CAD tools do not need to be changed.

## 5 Conclusions

The *size* of circuits implementing BFs can be reduced (theoretically, from exponential down to linear) by allowing KGs. These are analog gates that raise implementation and reliability concerns [43], and limit the applicability of the method to very small *fan-ins*, hence KGs should be used in advanced libraries of gates, and in the design of novel building blocks for future (even beyond CMOS) FPGAs.

Allowing combinations of BGs, KGs, and TLGs, can reduced the *size* significantly. Our expectation is that both *power* and *area* could be reduced (as compared to classical Boolean implementations).

The optimal combination of different types of gates has the potential of minimizing circuit *size*, but a lot of work is needed on: the design and implementation of KGs, the theory for optimally combining BGs, TLGs and KGs, and the development of CAD tools able to take into account different logic styles. Obviously, novel high-level logic synthesis tools are needed considering that emerging nano devices such as resonant tunneling devices (RTD), single electron tunneling (SET) devices, or molecular ones are implementing TLGs [44, 45].

## References

1. Nechiporuk, E.I.: The synthesis of networks from threshold elements. *Prob. Kiber.* **11** (1964) 49–62 [*Autom. Express* **7** (1964) 27–32 and 35–39]
2. Lupanov, O.B.: On circuits of threshold elements. *Dokl. Akad. Nauk SSSR* **202** (1971) 1288–1291 [*Sov. Phys. Dokl.* **17** (1972) 91–93]
3. Muroga, S.: *Threshold Logic and Its Applications*. John Wiley, New York (1971)
4. Beiu, V.: Constructive threshold logic addition: A synopsis of the last decade. *Proc. Intl. Conf. Artif. Neural Networks (LNCS 2714)*. Springer, Berlin (2003) 745–752
5. Beiu, V., Quintana, J.M., Avedillo, M.J.: VLSI implementation of threshold logic: A comprehensive survey. *IEEE Trans. Neural Networks* **14** (2003) 1217–1243
6. Myhill, J., Kautz, W.H.: On the size of weights required for linear-input switching functions. *IRE Trans. Electr. Comp.* **EC-10** (1961) 288–290
7. Håstad, J.: On the size of weights for threshold gates. *SIAM J. Discr. Math.* **7** (1994) 484–492
8. Haring, D.R.: Multi-threshold threshold elements. *IEEE Trans. Electr. Comp.* **EC-15** (1966) 45–65
9. Diep, T.A.: Capacity of multilevel threshold devices. *IEEE Trans. Inform. Th.* **44** (1998) 241–255
10. Hurst, S.L.: Multiple-valued logic: Its status and its future. *IEEE Trans. Comp.* **33** (1984) 1160–1179
11. Smith, K.C.: Multiple valued logic: A tutorial and appreciation. *IEEE Comp.* **21** (1988) 17–27
12. The International Technology Roadmap for Semiconductors (2004) <http://public.itrs.net/>
13. Beiu, V., Rückert, U., Roy, S., Nyathi, J.: On nanoelectronic architectural challenges and solutions. *Proc. IEEE Conf. Nanotech. Munich Germany* (2004) 628–631
14. Hilbert, D.: Mathematische probleme. *Nachr. Akad. Wiss. Göttingen* (1900) 253–297 [*Bull. Amer. Math. Soc.* **8** (1902) 437–479]
15. Kolmogorov, A.N.: On the representation of continuous functions of several variables by superposition of continuous functions of fewer variables. *Dokl. Akad. Nauk SSSR* **108** (1956) 179–182 [*Amer. Math. Soc. Transl.* **17** (1961) 369–373]
16. Arnol'd, V.I.: On functions of three variables. *Dokl. Akad. Nauk SSSR* **114** (1957) 679–681 [*Amer. Math. Soc. Transl.* **28** (1963) 51–54]
17. Kolmogorov, A.N.: On the representation of continuous functions of many variables by superpositions of continuous functions of one variable and addition. *Dokl. Akad. Nauk SSSR* **114** (1957) 953–956 [*Amer. Math. Soc. Transl.* **28** (1963) 55–59]
18. Sprecher, D.A.: On the structure of continuous function of several variables. *Trans. Amer. Math. Soc.* **115** (1965) 340–355
19. Lorenz, G.G.: Representation of functions of several variables by function of one variable. Chp. 11 in *Approximations of Functions*. Holt, Rinehart, and Winston, New York (1966)
20. Sprecher, D.A.: On the structure of representations of continuous functions of several variables as finite sums of continuous functions of one variable. *Proc. Amer. Math. Soc.* **17** (1966) 98–105
21. de Figueiredo, R.J.P.: Implications and applications of Kolmogorov's superposition theorem. *IEEE Trans. Autom. Control* **25** (1980) 1227–1231
22. Hecht-Nielsen, R.: Kolmogorov's mapping neural network existence theorem. *Proc. Intl. Conf. Neural Networks. San Diego CA* (1987) vol. 3 11–14
23. Girosi, F., Poggio, T.: Representation properties of networks: Kolmogorov's theorem is irrelevant. *Neural Computation* **1** (1989) 465–469
24. Kůrková, V.: Kolmogorov's theorem is relevant. *Neural Computation* **3** (1991) 617–622
25. Kůrková, V.: Kolmogorov's theorem and multilayer neural networks. *Neural Networks* **5** (1992) 501–506

26. Lin, J.N., Unbehauen, R.: On realization of a Kolmogorov network. *Neural Computation* **5** (1993) 18–20
27. Nakamura, M., Mines, R., Kreinovich, V.: Guaranteed intervals and Kolmogorov's theorem (and their possible relation to neural networks). *Interval Computations* **3** (1993) 183–199
28. Sprecher, D.A.: A universal mapping for Kolmogorov's superposition theorem. *Neural Networks* **6** (1993) 1089–1094
29. Katsuura, H., Sprecher, D.A.: Computational aspects of Kolmogorov's superposition theorem. *Neural Networks* **7** (1994) 455–461
30. Nees, M.: Approximative versions of Kolmogorov's superposition theorem, proved constructively. *J. Comp. Appl. Math.* **54** (1994) 239–250
31. Sprecher, D.A.: A numerical implementation of Kolmogorov's superpositions. *Neural Networks* **9** (1996) 765–772
32. Sprecher, D.A.: A numerical construction of a universal function for Kolmogorov's superpositions. *Neural Network World* **6** (1996) 711–718
33. Sprecher, D.A.: A numerical implementation of Kolmogorov's superpositions II. *Neural Networks* **10** (1997) 447–457
34. Brattka, V.: A computable Kolmogorov superposition theorem. In Blanck, J., Brattka, V., Hertling, P., Weihrauch, K. (Eds.): *Computability and Complexity in Analysis*. Informatik Berichte vol. 272 FernUniversität Hagen (2000) 7–22
35. Neruda, R., Štědý, A., Drkošová, J.: Towards feasible learning algorithm based on Kolmogorov theorem. *Proc. Intl. Conf. AI. Las Vegas USA* (2000) 915–920
36. Köppen, M.: On the training of a Kolmogorov network. *Proc. Intl. Conf. Artif. Neural Networks (LNCS 2415)*. Springer, Berlin (2002) 474–479
37. Beiu, V.: Optimization of circuits using a constructive learning algorithm. In Bulsari, A.B., Kallio, S. (Eds.): *Neural Networks in Engineering Systems*. Åbo Akademis Tryckeri (1997) 291–294
38. Beiu, V.: Neural inspired parallel computations require analog processors. *Proc. Intl. Parallel Comp. and Electr. Eng. Conf. Bialystok Poland* (1998) 39–53
39. Beiu, V.: On Kolmogorov's superposition and Boolean function. *Proc. Brazilian Symp. Neural Networks. Belo Horizonte Brazil* (1998) 55–60
40. Beiu, V., Makaruk, H.E.: Deeper sparser nets can be optimal. *Neural Proc. Lett.* **8** (1998) 201–210
41. Sarpeshkar, R.: Analog versus digital: Extrapolating from electronics to neurobiology. *Neural Computation* **10** (1998) 1601–1638
42. Sarpeshkar, R., O'Halloran, M.: Scalable hybrid computation with spikes. *Neural Computations* **14** (2002) 2003–2038
43. Beiu, V.: A novel highly reliable low-power nano architecture: When von Neumann augments Kolmogorov. *Proc. Appl.-specific Sys., Arch. and Proc. Galveston USA* (2004) 167–178
44. Avedillo, M.J., Quintana, J.M.: A threshold logic synthesis tool for RTD circuits. *Proc. Euromicro Symp. Digital Sys. Design. Rennes, France* (2004) 624–627
45. Zhang, R., Gupta, P., Zhong, L., Jha, N.K.: Threshold network synthesis and optimization and its application to nanotechnologies. *IEEE Trans. CAD* **24** (2005) 107–118

# CMOL CrossNets as Pattern Classifiers

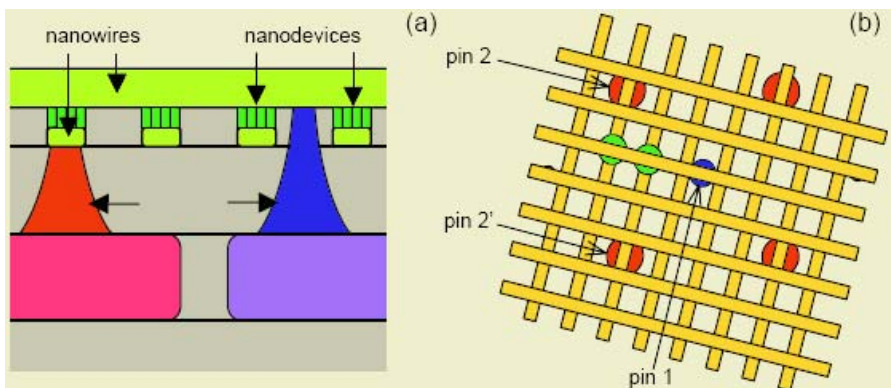
Jung Hoon Lee and Konstantin K. Likharev

Stony Brook University, Stony Brook, NY 11794-3800, U.S.A  
{jlee@grad.physics, klicharev@notes.cc}sunysb.edu

**Abstract.** This presentation has two goals: (i) to review the recently suggested concept of bio-inspired CrossNet architectures for future hybrid CMOL VLSI circuits and (ii) to present new results concerning the prospects and problems of using these neuromorphic networks as classifiers of very large patterns, in particular of high-resolution optical images. We show that the unparalleled density and speed of CMOL circuits may enable to perform such important and challenging tasks as, for example, online recognition of a face in a high-resolution image of a large crowd.

## 1 CrossNets

There is a growing consensus that the forthcoming problems of the Moore Law [1] may be only resolved by the transfer from a purely semiconductor-transistor (CMOS) technology to hybrid (“CMOL”) integrated circuits [2, 3]. Such circuit would complement a CMOS chip with a nanowire crossbar (Fig. 1) with nanodevices (e.g., specially designed functional molecules) formed between the nanowires at each crosspoint.



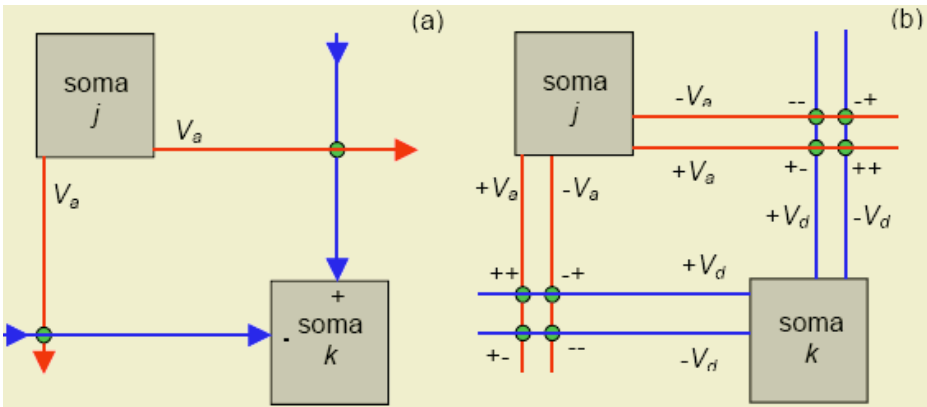
**Fig. 1.** CMOL circuit (schematically): (a) side view and (b) top view showing several adjacent pins. The latter view shows that the specific angle between the interface pin lattice and nanodevice crossbar allows each nanodevice to be addressed via the appropriate pin pair (e.g., pins 1 and 2 for the left of the two shown devices, and pins 1 and 2' for the right device)

The basic idea behind such hybrid circuits is that its minimum features are not defined by lithography (which below  $\sim 10$  nm will become prohibitively expensive) but by a Nature-given standard such as the size of a certain molecule. Estimates show [3] that CMOL circuits may feature unprecedented density (up to  $\sim 10^{12}$  active devices per  $\text{cm}^2$ ), at acceptable fabrication costs.

Realistically, nanodevices will hardly ever be formed (e.g., chemically self-assembled) with 100% yield. This is why CMOL circuit architectures should ensure their high defect tolerance. Recently we have shown that such high tolerance may be achieved in cell-based FPGA-type reconfigurable logic circuits [3, 4] and (to a less extent) in hybrid memories [3, 5]. However, the most natural application of the CMOL technology is in bio-inspired neuromorphic networks which are generically defect-tolerant.

We have proposed [6-8] a family of such architectures, called Distributed Crossbar Networks (“CrossNets”), whose topology uniquely maps on CMOL circuits. Figure 2a shows the generic architecture of our CrossNets. Relatively sparse neural cell bodies (“somas”) are implemented in the CMOS subsystem. In the simplest firing rate model, each soma is just a differential amplifier with a nonlinear saturation (“activation”) function  $V_{\text{out}} = f(V_{\text{in}})$ . Axons and dendrites are implemented as physically similar, straight segments of nanowires, while nanodevices (latching switches), formed at the nanowire crosspoints, play the role of elementary synapses. Axonic voltage  $V_a$ , developed by the somatic amplifier, is transferred by each axonic nanowire to synapses, so that if the synaptic latch of a particular synapse is in the ON state, a current proportional to  $V_a$  is flowing into the corresponding dendritic nanowire, contributing to the input signal of the post-synaptic cell.

In the generic CrossNets (Fig. 2a), any pair of cells is interconnected by two synapses leading to the opposite inputs of the somatic amplifier, so that the net synaptic weight  $w_{jk}$  may take any of three values which may be normalized to -1, 0, and +1. In other CrossNets versions the number of synapses is larger. In particular, in



**Fig. 2.** Cell coupling is the (a) generic feedforward and (b) Hebbian feedforward CrossNets. Here and below, axonic nanowires are shown in red, synaptic nanowires in blue. Each gray square show the interface pin area of a somatic cell. (The cells as such may be much larger, since they are implemented in the underlying CMOS subsystem)

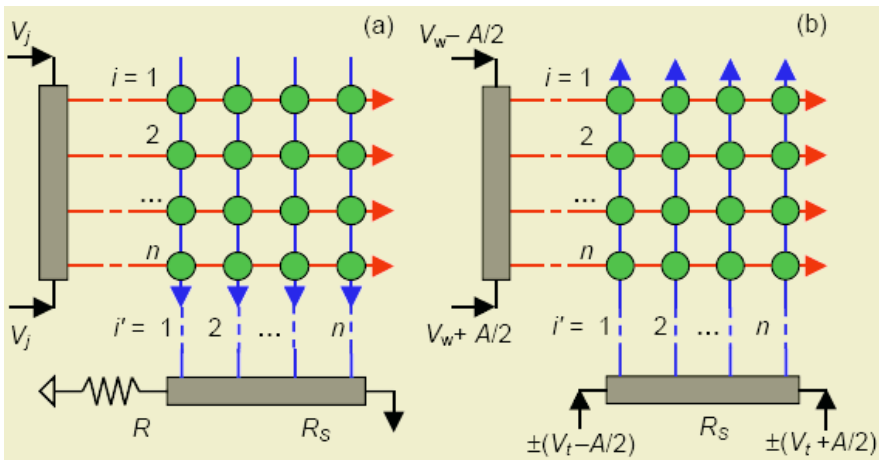
recurrent CrossNets, the number of nanowires and synapses per cell is doubled to carry feedback signals. (Generally, CrossNets are asymmetric:  $w_{kj} \neq w_{jk}$ .) In order to enable quasi-Hebbian learning rule, the number of nanowires and synapses per cell may be increased even further (Fig. 2b).

In the simplest cases (e.g., quasi-Hopfield networks with finite connectivity),<sup>1</sup> the tri-level synaptic weights of the generic CrossNets are quite satisfactory, leading to just a very modest network capacity loss [7]. However, some applications (in particular, pattern classification) may require a larger number of weight quantization levels  $L$  (e.g.,  $L \sim 30$  for a 1% fidelity [8]). This can be achieved by using compact square arrays (e.g.,  $4 \times 4$ ) of latching switches (Fig. 3).

In CrossNets the CMOS-implemented somatic cells are large and sparse. Because of this, the each axonic signal is passed to dendritic wires of many ( $M$ ) other cells, and each dendritic signal is contributed by  $M$  pre-synaptic cells. The distribution of somatic cells may vary, creating several CrossNet species (Fig. 4).

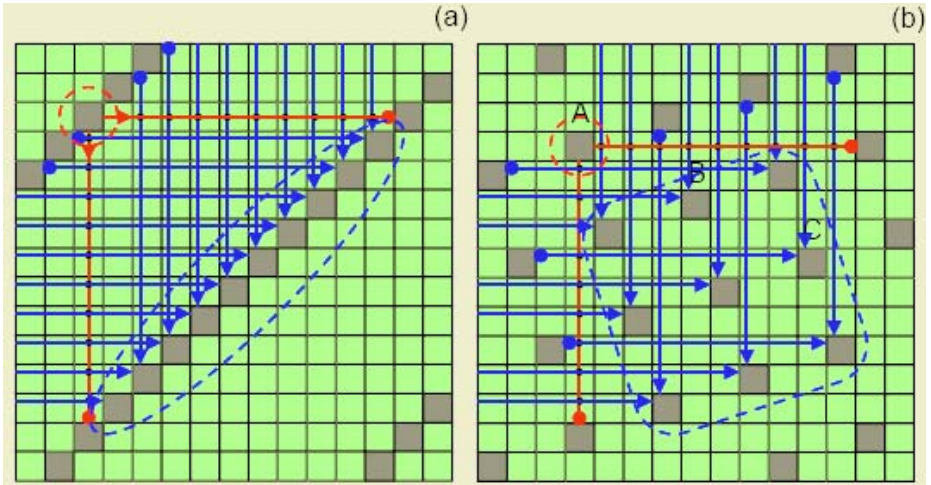
CrossNet training faces several hardware-imposed challenges:

- (i) The synaptic weight contribution provided by the elementary latching switch is binary.
- (ii) The only way to adjust any particular synaptic weight is to turn ON or OFF the corresponding latching switch(es). This is only possible to do by applying certain voltage  $V = V_a - V_d$  between the two corresponding nanowires. At this procedure, other nanodevices attached to the same wires should not be disturbed.



**Fig. 3.** A half of a composite synapse for providing  $L = 2n^2 + 1$  discrete levels of the weight in (a) operation and (b) weight import modes. The dark-gray rectangles are resistive metallic strips at soma/nanowire interfaces

<sup>1</sup> So far, this operation mode is the only one for which the CrossNet defect tolerance has been analyzed in detail [8]. The results are very encouraging: for example, the network may have a 99% fidelity, with a 50% capacity loss, at a fraction of bad nanodevices above 80% (!).



**Fig. 4.** Two main CrossBar species: (a) FlossBar and (b) InBar, in the generic (feedforward, non-Hebbian, ternary-weight) case for the connectivity parameter  $M = 9$ . Only the nanowires and nanodevices coupling one cell (indicated with red dashed lines) to  $M$  post-synaptic cells (blue dashed lines) are shown; actually all the cells (e.g., B and C on panel (b)) are similarly coupled. Bold points show open-circuit terminations of the axonic and dendritic nanowires, which prevent cell interaction in bypass of synapses

- (iii) Processes of turning single-electron latches ON and OFF are statistical rather than dynamical [2], so that the applied voltage  $V$  can only control probability rates of these, generally random events. (This problem is least significant, because the randomness may be confined by an appropriate design of the nanodevices [6].)

We have shown that these challenges may be met using (at least) the following training methods [8]:

(i) *Synaptic weight import.* This procedure is started with training of a homomorphic “precursor” artificial neural network with continuous synaptic weights  $w_{jk}$ , implemented in software, using one of established methods (e.g., error backpropagation). Then the synaptic weights  $w_{jk}$  are transferred to the CrossNet, with some “clipping” (rounding) due to the binary nature of elementary synaptic weights. To accomplish the transfer, pairs of somatic cells are sequentially selected via CMOS-level wiring. Using the flexibility of CMOS circuitry, these cells are reconfigured to apply external voltages  $\pm V_W$  to the axonic and dendritic nanowires leading to a particular synapse, while all other nanowires are grounded. The voltage level  $V_W$  is selected so that it does not switch the synapses attached to only one of the selected nanowires, while voltage  $2V_W$  applied to the synapse at the crosspoint of the selected wires is sufficient for its reliable switching. (In the composite synapses with quasi-continuous weights (Fig. 4), only a part of the corresponding switches is turned ON or OFF.)

(ii) *Error backpropagation.* The synaptic weight import procedure is straightforward when  $w_{jk}$  may be simply calculated, e.g., for the Hopfield networks.



However, for very large CrossNets used as classifiers the precursor network training may take an impracticably long time. In this case the direct training of a CrossNet may become necessary. We have developed two methods of such training, both based on “Hebbian” synapses (Fig. 2b). In CrossNets with such synapses, each axonic and dendritic signal is now passed, in the dual-rail format, through two nanowires to groups of four latching switches. Starting from the Arrhenius law describing the probability of switching of each single-electron latch, it is straightforward to show [8] that the average synaptic weight of the 4-latch group obeys the following equation:

$$\frac{d}{dt}\langle w \rangle = -4\Gamma_0 \sinh(\gamma S) \sinh(\gamma V_a) \sinh(\gamma V_d), \quad (1)$$

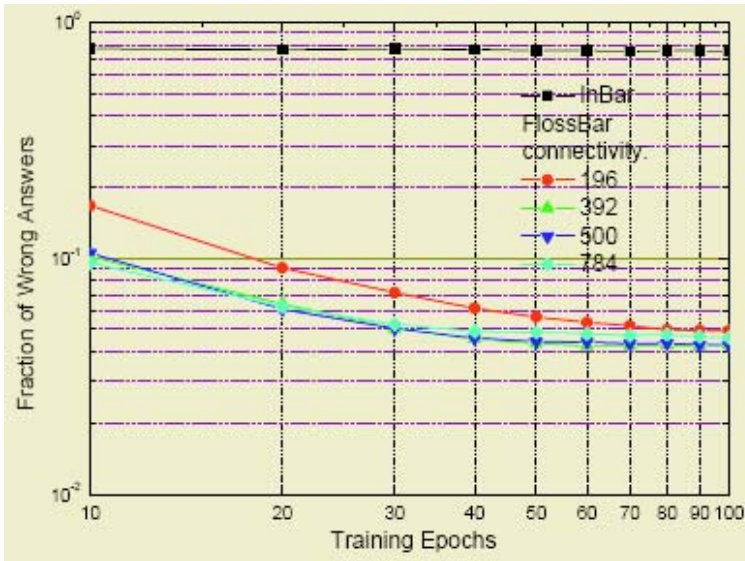
where  $\Gamma_0$  and  $\gamma$  are parameters of the latching switch, while  $S$  is a global, externally-controllable shift voltage which may be applied to all switches via a special gate. The quasi-Hebbian rule (1) may be used to implement the backpropagation algorithm either using a periodic time-multiplexing [8] or in a continuous fashion, using the simultaneous propagation of signals and errors along the same dual-rail channels

## 2 Pattern Classification

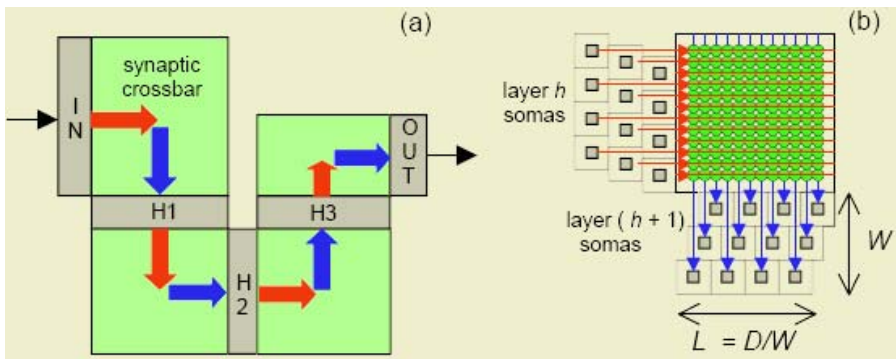
It could seem that both feedforward CrossNet species shown in Fig. 4 may be used as multilayered perceptrons (MLPs) for pattern classification [9]. However, in both cases there are some problems. FlossBars (Fig. 4a) are indeed the MLPs, but with limited connectivity between the layers, while InBars (Fig. 4b) do not have a layered structure at all. (For example, cell C gets input not only from the “first-layer” cell A, but also from the “second-layer” cell B.)

Figure 5 shows typical results of our study of an impact of these features on the performance of these networks as classifiers. The results show that, unfortunately, InBars cannot be trained to work well as classifiers, at least by error backpropagation. (At this stage, we still cannot offer a simple explanation for this fact.) On the other hand, finite connectivity of FlossBars does not affect their fidelity too substantially. Moreover, some connectivity restriction turns out to be useful to achieve the best training results. Unfortunately, Fig. 4a shows that FlossBars are not very convenient for CMOS implementation: interconnect pin areas are too close (at the distance of the order of nanowire pitch), so that CMOS layout of somatic cells may be difficult. (This would not be a problem for InBar – Fig. 4b.)

Figure 6 shows a possible way to overcome these difficulties: an MLP with a global synaptic crossbars providing full connectivity between layers implemented in InBar style. The only evident negative feature of this network is a certain loss of chip real estate, since nanodevices over the CMOS somatic cell areas are not used. (In contrast, CMOS circuits under the synaptic crossbars may be utilized – see below.) This loss, however, may be very limited, because the somatic area width  $W$  (in terms of the cell number) is only determined by the pitch ratio of CMOS and nanowiring, and may be of the order of 10, i.e. substantially smaller than the data vector length  $D \sim 10^3$  for the most challenging classifier applications. In this case the relative area loss is of the order of  $(LW)/L^2 = W^2/D \sim 10\%$ , i.e. negligible.



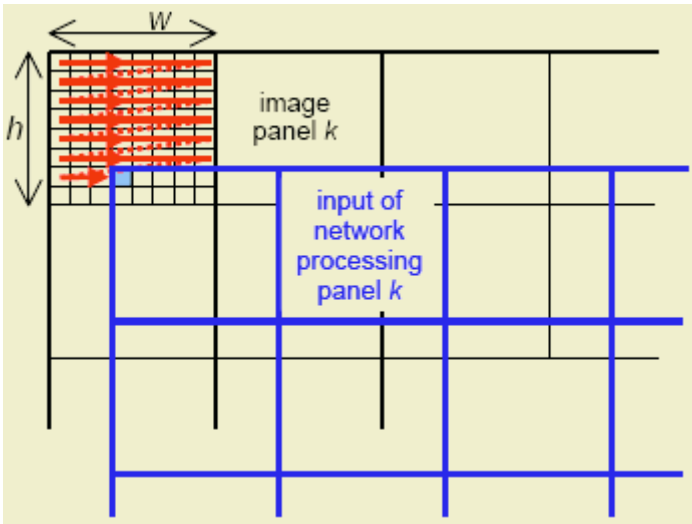
**Fig. 5.** Test set fidelity of an InBar (with  $N = 784+784+10$  cells and connectivity  $M = 784$ ) and FlossBar MLPs (also with  $784+784+10$  cells with limited connectivity between the input and first hidden layer) after their training as classifiers of the MNIST data set of handwritten characters [10]. In both cases, the training set size was 60,000 images, test set 10,000 patterns, with  $28 \times 28 = 784$  pixels (with 256 shades of gray) each



**Fig. 6.** Multi-layered perceptron based on InBar somatic areas and global synaptic crossbars: (a) general structure and (b) CMOL implementation of two adjacent layers (schematically). In this case, each green circle denotes a composite synapse consisting of two arrays shown in Fig. 3, with quasi-continuous synaptic weight

Let us give an approximate estimate of performance of such CMOL classifiers for such a challenging and important task as a search of a particular person’s face on a high-resolution picture of a large crowd. The most difficult feature of such recognition is the search for face location, i.e. optimal placement of a face on the

image relative to the panel providing input for the processing network. The enormous density and speed of CMOL hardware gives a possibility to time-and-space multiplex this task (Fig. 7). In this approach, the full image (say, formed by CMOS photodetectors on the same chip) is divided into  $P$  rectangular panels of  $h \times w$  pixels, corresponding to the expected size and approximate shape of a single face. A CMOS-implemented communication channel passes input data from each panel to the corresponding CMOL neural network, providing its shift in time, say using the TV scanning pattern (red line in Fig. 7). The standard methods of image classification require the network to have just a few hidden layers, so that the time interval  $\Delta t$  necessary for each mapping position may be so short that the total pattern recognition time  $T = hw\Delta t$  may be acceptable even for online face recognition.



**Fig. 7.** Scan mapping of the input image on CMOL neural network inputs. Red lines show the possible time sequence of image pixels sent to a certain input of the network processing image from the upper-left panel of the pattern

Indeed, let us consider a 4-Megapixel image partitioned into 4K 32×32-pixel panels ( $h = w = 32$ ). This panel will require a neural network with several (say, four) layers with 1K cells each in order to compare the panel image with  $\sim 10^3$  stored faces. With the feasible 45-nm CMOS technology [1], 4-nm nanowire half-pitch [3, 4], and 65-level synapses (sufficient for better than 99% fidelity [8]), each interlayer crossbar would require chip area about  $(4K \times 64 \text{ nm})^2 = 64 \times 64 \mu\text{m}^2$ , fitting 4×4K of them on a  $\sim 0.6 \text{ cm}^2$  chip.<sup>2</sup> With the typical nanowire capacitance of  $2 \times 10^{-10} \text{ F/m}$  and latching

<sup>2</sup> The CMOS somatic-layer and communication-system overheads are negligible. Also small is the chip area required for the output signal pins, because the search result may be demultiplexed into just a few bits (say, the recognized face’s ID number and its position on the image).

switch ON resistance  $\sim 10^{10} \Omega$  (providing acceptable power consumption of the order of  $10 \text{ W/cm}^2$  [8]), the input-to-output signal propagation in such a network will take only  $\sim 4 \times (8 \times 10^{-9} \text{ m}) \times (2 \times 10^{-10} \text{ F/m}) \times (10^{10} \Omega) \approx 50 \text{ ns}$ , so that  $\Delta t$  may be of the order of 100 ns and the total time  $T = hw\Delta t$  of processing one frame of the order of 100 microseconds, much shorter than the typical TV frame time of  $\sim 10$  milliseconds. The remaining two-order-of magnitude gap may be used, for example, for double-checking the results via stopping the scan mapping (Fig. 7) at the most promising position. (For this, a simple feedback from the recognition output to the mapping communication system is necessary.)<sup>3</sup>

It is instructive to compare the estimated CMOL chip speed with that of the implementation of a similar parallel network ensemble on a CMOS signal processor (say, also combined on the same chip with an array of CMOS photodetectors). Even assuming an extremely high performance of 30 billion additions/multiplications per second, we would need  $\sim 4 \times 4\text{K} \times 1\text{K} \times (4\text{K})^2 / (30 \times 10^9) \approx 10^4$  seconds  $\sim 3$  hours per frame, evidently incompatible with the online image stream processing.

### 3 Conclusions

Our preliminary estimates show that even such a brute-force approach as a parallel use of a few thousand similar neural networks on a single silicon chip, with time-multiplexing analysis of each of  $\sim 10^3$  possible mappings of a high-resolution image on the network input, may enable CMOL CrossNets to perform tasks clearly impossible for the usual implementation of neural networks on serial digital computers. Our nearest goals are to verify and quantify these predictions via explicit numerical simulation of such systems, and to work on the development of more elaborate schemes for CMOL CrossNet applications as classifiers of complex patterns.

Useful discussions with X. Ma and Ö. Türel are gratefully acknowledged. The work has been supported by AFOSR, NSF, and MARCO FENA Center.

### References

1. International Technology Roadmap for Semiconductors. 2003 Edition, 2004 Update, available online at <http://public.itrs.net/>
2. Likharev, K.K.: Electronics Below 10 nm. In: Greer, J. *et al.* (eds.) Nano and Giga Challenges in Microelectronics. Elsevier, Amsterdam (2003) 27-68
3. Likharev, K.K., Strukov, D.B.: CMOL: Devices, Circuits, and Architectures. In: Cuniberti, G. *et al.* (eds.) Introduction to Molecular Electronics. Springer, Berlin (2005)
4. Strukov D.B., Likharev, K.K.: A Reconfigurable Architecture for Hybrid CMOS/Nanoscale Circuits" (with D. B. Strukov). Report at FCCM'05, to be published (2005)

<sup>3</sup> Reportedly, hierarchical networks, e.g., the so-called LeNets which have been successfully used for handwriting recognition [10], may be more defect tolerant. Our estimates have shown that a similar set of parallel networks of this type would require just a little bit larger chips – about  $1 \text{ cm}^2$  for the parameters cited above.

5. Strukov D.B., Likharev, K.K.: Prospects for Terabit-scale Nanoelectronic Memories. *Nanotechnology* 16 (2005) 137-148
6. Fölling, S., Türel, Ö, Likharev, K.K.: Single-Electron Latching Switches as Nanoscale Synapses. In: *Proceedings of the IJCNN (2001)* 216-221
7. Türel, Ö., Likharev, K.K.: CrossNets: Possible Neuromorphic Networks Based on Nanoscale Components. *Int. J. of Circ. Theor. Appl.* 31 (2003) 37-53
8. Türel, Ö., Lee, J.H., Ma, X., Likharev, K.K.: Neuromorphic Architectures for Nanoelectronic Circuits. *Int. J. of Circ. Theor. Appl.* 32 (2004) 277-302
9. See, e.g., Hertz, J., Krogh, A. and Palmer, R.G.: *Introduction to the Theory of Neural Computation*. Perseus, Cambridge, MA (1991)
10. LeCun, Y. *et al.*: Gradient-based Learning Applied to Document Recognition. *Proc. IEEE* 86 (1998) 2278-2324

# Analog VLSI Implementation of Adaptive Synapses in Pulsed Neural Networks

Tim Kaulmann, Markus Ferber, Ulf Witkowski, and Ulrich Rückert

System and Circuit Technology, Heinz Nixdorf Institute,  
University of Paderborn, Fürstenallee 11, 33102 Paderborn, Germany  
kaulmann@hni.upb.de  
<http://www.hni.upb.de/sct>

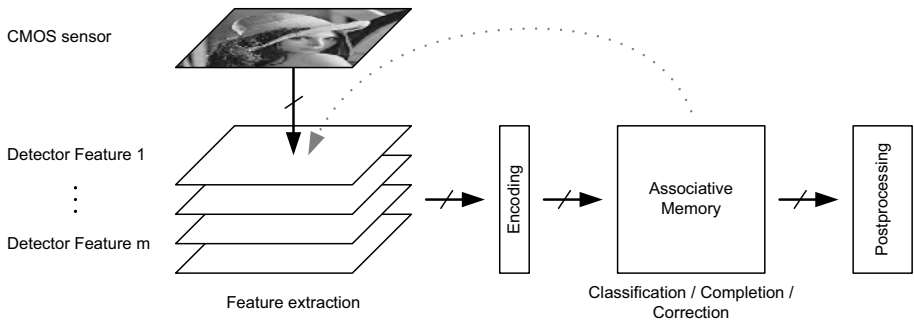
**Abstract.** An analog VLSI implementation of adaptive synapses being part of an associative memory realised with pulsed neurons is presented. VLSI implementations of dynamic synapses and pulsed neurons are expected to provide robustness and low energy consumption like observed in the human brain. We have developed a VLSI implementation of synaptic connections for an associative memory which is used in a biological inspired image processing system using pulse coded neural networks. The system consists of different layers for feature extraction to decompose the image in several features. The pulsed associative memory is used for completing or binding features. In this paper, we focus on the dynamics and the analog implementation of adaptive synapses. The discussed circuits were designed in a 130 nm CMOS process.

## 1 Introduction

A biological inspired approach for image processing is a system that uses principles observed in the visual system of the human brain. It is expected that a bio-inspired system is more robust and has a wide field of application as well as low power consumption. We are developing a bio-inspired image processing system that consists of different layers like the human visual system<sup>1</sup>. A schematic of the system is depicted in fig. 1. The first layer is a CMOS sensor with a resolution of  $128 \times 128$  pixel. Every pixel-cell generates a current depending on the brightness of the captured image. The current of all cells are fed in parallel to the underlying layers. Each following layer forms a detector for a specific feature. The feature detectors are built by groups of neurons that are connected in various ways to deliver the desired functionality. Every layer detects a specific feature, for example spots, edges or pattern identified by Gabor-filter [1, 2].

---

<sup>1</sup> The development of the image processing system with pulsed neurons is a cooperation project with the project leader Infineon Technologies AG, U. Ramacher, A. Heitmann, Munich, and the project partners: University of Berlin, Dresden, Paderborn, Ulm. The Project is funded by the Federal Ministry of Education and Research (BMBF).



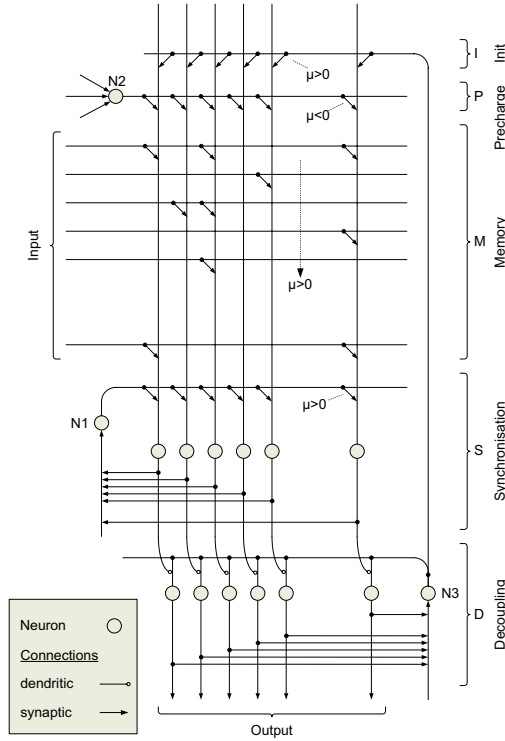
**Fig. 1.** Image processing system with feature detectors and AM

To achieve object recognition in the image processing system the information from the different feature detectors has to be bound to a symbolic representation of an object. This can be done by applying an associative memory (AM) after the detector layers. An associative memory can also be used to correct or to complete information from the detectors when the information is partially available only. In our approach, the size of the input vector and the output vector of the AM is the number of outputs of the previous feature detector.

Synapse implementations as proposed in [8] provide a constant weight function for pulsed neurons. Other implementations [9] provide a Hebbian learning rule. In our approach we adapt the weight to the actual state of neighboring neurons and membrane potentials and provide an analog, dynamic weight. The adaption of the synapse depending on the post-synaptic membrane potential does not only implement a form of Hebbian learning, but also provides a mechanism for synchronisation of neighboring neurons. The advantage of a dynamic weight is the reduced complexity of the neurons by maintaining the ability of complex signal processing.

## 2 Associative Memory

The AM consists of five layers as depicted in fig. 2. Its principle is based on a binary neural associative memory (BiNAM) as it uses a programmable matrix  $M$  to perform a mapping of the input vector (feature vector). The characteristics of binary neural associative memories are discussed in detail in [7]. A VLSI implementation of a BiNAM is proposed in [4, 5]. The additional layers around the memory-matrix are a synchronisation layer  $S$  for synchronising asynchronous pulses from the input vector expected from detectors like the spot-detector. A similar version of the synchronisation layer is discussed in [3]. Layer  $D$  is for decoupling the synchronisation layer from the output vector when the AM is initialised by layer  $I$  if necessary. At last layer  $P$  is used for precharging the synchronisation layer. The method of precharging is described later. Additionally, three surveillance neurons are used which measure the local activity in each layer and activate functions in the different layers.



**Fig. 2.** Structure of the associative memory with pulsed neurons and dynamic synapses

### 2.1 Neuron Model

The neuron model used in the image processing system is an integrate and fire (IAF) neuron without active decay of the membrane potential [1]. It also lacks a refractory period after pulse generation. All pulses from pre-synaptic neurons  $l$  from a set  $N_k$  to a post-synaptic neuron  $k$  are integrated to a membrane potential  $a_k$  according to (1), where  $x_l \in \{0, 1\}$  denotes a firing event of the pre-synaptic neuron  $l$ . The parameter  $w_{kl}$  is the weight of the synaptic connection between neuron  $l$  and  $k$ .

$$a_k(t) = \int \sum_{l \in N_k} [w_{kl}(t) \cdot x_l(t) + w_{k0} \cdot i_k(t)] dt \quad (1)$$

The weight  $w_{k0}$  is only used in neurons on the layer directly after the CMOS sensor where the gradual current  $i_k$  from the CMOS cells is integrated and the image information is transformed to a rate-based code in the pulsed neural network.



## 2.2 Dynamic Synapses

For the realisation of several neural structures we have implemented two types of connections. The first type is a synaptic weight that underlies the dynamic given in (2). The dynamics highly depends on the membrane potential  $a_k$  of a post-synaptic neuron and on the parameter  $\mu$  as well as on a constant decay rate adjusted by parameter  $\gamma$  [1].

$$\frac{d}{dt}w_{kl} = -\gamma \cdot w_{kl} + \mu \cdot \left(a_k - \frac{\Theta}{2}\right) \cdot x_l \quad (2)$$

Now we examine a neural system consisting of a pre-synaptic neuron, a synaptic weight and a post-synaptic neuron at a time when the weight is close to zero. If the parameter  $\mu$  is chosen as a negative value and a post-synaptic neuron's membrane potential is lower than  $\Theta/2$ , where  $\Theta$  defines the firing threshold of a neuron, the pulse of a pre-synaptic neuron lowers the synaptic weight even more. The weights in our model have a lower bound of zero, so further pre-synaptic pulses can not increase the membrane potential on the post-synaptic neuron anymore. If the membrane potential of the post-synaptic neuron is higher than  $\Theta/2$ , the pre-synaptic pulses raise the weight depending on the parameter  $\mu$  and increase the membrane potential of the post-synaptic neuron.

If the parameter  $\mu$  is chosen as a positive value the membrane potential of the post-synaptic neuron can only be increased when its value is lower than  $\Theta/2$ . With specific values for the parameters  $\gamma$  and  $\mu$  we can build a system of three neurons and two synaptic connections where one neuron can precharge the post-synaptic neuron, and another neuron can cause a fire event of the post-synaptic neuron only if it has been precharged. This principle is used in the synchronisation layer of the pulsed associative memory.

For the VLSI implementation of the synaptic rule described in (2) the second term of the equation has been modified. The Term  $g(a_k)$  is a nonlinear function weighting the post-synaptic membrane potential  $a_k$  of neuron  $k$ . For the function  $g(a_k)$  we use the tanh-characteristic of a differential amplifier in the VLSI implementation.

$$\frac{d}{dt}w_{kl} = -\gamma \cdot w_{kl} + g(a_k) \cdot x_l \quad (3)$$

The second type of connection is a dendritic type. The weight adaption rule between two neurons  $k$  and  $l$  is given by (4), where  $i_{pre}$  describes the current coming from a synaptic connection between a second pre-synaptic neuron and neuron  $k$ ,  $x_{pre} \in \{0, 1\}$  the fire event of the pre-synaptic neuron,  $\alpha$  an offset (here chosen as 0) and  $w_\infty$  a value the weight relaxes to when no pre-synaptic current is received.

$$\frac{d}{dt}w_{kl} = -\gamma \cdot (w_\infty - w_{kl}) + \mu \cdot (i_{pre} - \alpha) \cdot w_{kl} \cdot x_{pre} \quad (4)$$

This adaption rule is only used in the decoupling layer  $L$  of the pulsed AM and could be simplified to a switch in our implementation.

### 3 VLSI Implementation

#### 3.1 Dynamic Synapse

For the VLSI implementation of the pulsed AM using a 130 nm technology process by Infineon Technologies AG, we have developed a dynamic synapse as depicted in fig. 3. With the output  $v_w\_dyn$  we control a pulsed current source, weighting the pre-synaptic pulse. The schematic of the current source, missing from the following synapse schematic, is given in fig. 4. The transistors M1-M5 form a differential amplifier which provides a smooth tanh characteristic  $g(a_k)$  instead of high differential amplification.

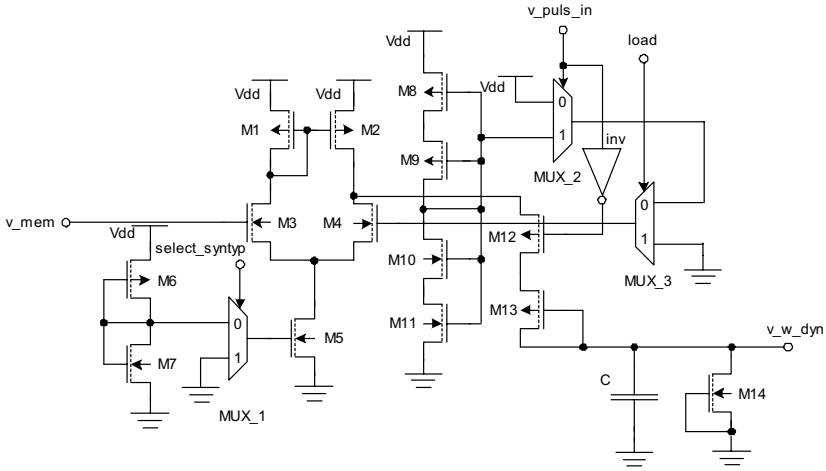


Fig. 3. Schematic of adaptive synapse with passive decay

The membrane potential  $a_k$  of the post-synaptic neuron  $k$ , present at the input  $v\_mem$ , is compared to a reference voltage generated by M8-M11. To reduce the impact of sub-threshold current and leakage current in the 130 nm process we introduce multiplexer units to cut off parts of the circuit, when not needed. Multiplexer MUX\_2 and MUX\_3 select the source of the reference voltage. If a pre-synaptic pulse is present at  $v\_puls\_in$  and the synapse is in normal operating mode ( $load=0$ ),  $v\_mem$  is compared to a voltage corresponding to the firing threshold of  $\Theta/2$ . Transistor M12 opens and the differential current, generated by M1-M5, charges the capacity C. M13 is connected as a diode to allow charging of the capacity only. Without the presynaptic pulse M12 closes and supply voltage is applied to the differential amplifier as reference voltage. This prevents further charging of the capacity by sub-threshold current. An active decay, dependend on the membrane potential of the post-synaptic neuron is not desired in this implementation. The passive decay parameter  $\gamma$  is implemented by the sub-threshold current of M14. MUX\_3 can also be used to imprint a voltage level for a constant weight mode if the load signal is set to high signal. An additional multiplexer (MUX\_1) is used to select the type of the synapse, that

we can choose between a constant or dynamic connection. We implemented the constant weight for simulations on unsynchronised input on a standard BiNAM. The bias voltage for transistor M5 is generated once by transistors M6 and M7 and provided to every adaptive synapse in the AM. Their gate-width is 200 nm and their gate-length is 1.5  $\mu\text{m}$  for M6 and 400 nm for M7. The voltage reference dissipates a static power of 6.4  $\mu\text{W}$ .

### 3.2 Pulsed Current Source

The variable pulsed current source as depicted in fig. 4 transforms voltage pulses generated by preceding neurons to corresponding current pulses by weighting supply current by transistor M3. The gate voltage of M3 is controlled by a pulse from a pre-synaptic neuron, appearing at the input  $v_{\text{pulse\_in}}$  and the voltage from a weight applied at the input  $v_{\text{ref}}$ . Transistors M4-M7 adapt the voltage  $v_{\text{ref}}$  to an appropriate gate voltage for M3. M7 limits the current flow through the voltage divider to reduce power dissipation. The voltage reference built by M8 and M9 dissipates a static power of 3.6  $\mu\text{W}$ .

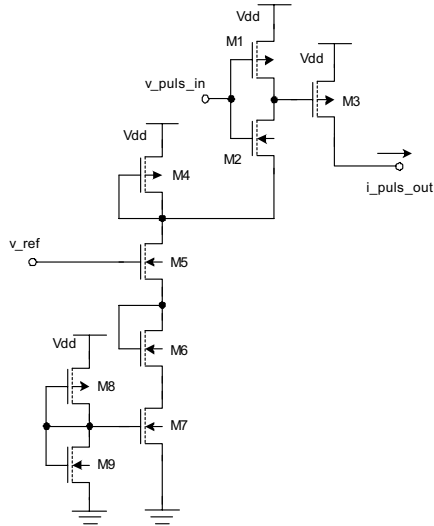


Fig. 4. Schematic of pulsed current source

## 4 Area and Power Consumption

The implemented dynamic synapse, including an additional multiplexer for switching to a constant weight as well as the presented variable pulsed current source, uses an area of 1073.03  $\mu\text{m}^2$ . The average power consumption is estimated with 24.07  $\mu\text{W}$  per active synapse. The static power consumption of unused synapses is estimated with 0.88  $\mu\text{W}$ . The implemented AM consisting of 203 neurons, 10400 dynamic synapses, 300 static synapses and control logic, uses an area of

18.04mm<sup>2</sup>. A maximal power consumption of 533mW and an average power consumption of 245mW, when only 1/8 of the dynamic synapses is active, is expected.

### 5 Simulation

Simulation has been accomplished on the schematics of the dynamic weight and the pulsed current source as well as additional neurons to generate pulses and provide the dynamics of the membrane potential. The membrane potential of a post-synaptic neuron is fed back to the input v\_mem of the dynamic weight. At the start of the simulation it is reset to 0. From 0ms to 10ms the weight is charged to a specific value. From 0ms to 400ms the pre-synaptic neuron generates pulses with a pulse-width of 10µs and a period 500µs at the input v\_pulse\_in. The weight, represented by v\_cap increases when the pre-synaptic neuron is firing and decreases slightly between the firing events by passive decay as depicted in fig. 5.

With the high constant weight, the membrane potential v\_mem of the post-synaptic neuron is uniformly charged until it reaches the firing threshold. After 400ms the pre-synaptic neuron is cut off from the weight. Without pre-synaptic pulses the weight is not charged anymore according to (2). Therefore, it decreases only by passive decay realised by transistor M14. The effect of the decreasing weight is a slower charging of the membrane potential of the post-synaptic neuron. The parameter v\_theta characterises the reference voltage of the differential

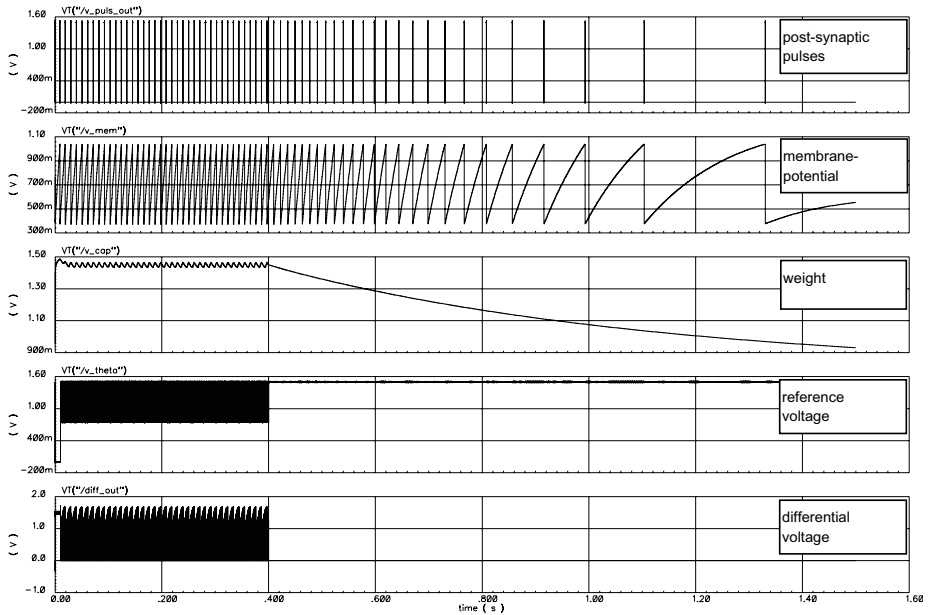


Fig. 5. Simulation: Rise and decay of the synaptic weight

amplifier. From 10 ms to 400 ms it changes between supply voltage and a voltage level representing a half of the firing threshold as described in section 3.1. The differential voltage of the differential amplifier is depicted as `diff_out`.

## 6 Conclusion

The image processing system in fig. 1 is capable of extracting different features from a captured image. The associative memory, necessary for binding sub-symbolic features to an object representation, or completing missing information was presented and the function of its different layers was described. The AM layers need different types of synapses to perform the desired function. At first the neuron model used in this paper was presented and two types of synapses with different dynamics were introduced. Further we have implemented the dynamic synapse in a 130 nm technology process provided by Infineon Technologies AG. The increased leakage and sub-threshold currents of the 130 nm technology process have been considered for the design of the dynamic synapse. Therefore, we introduced multiplexer units to cut off unused parts of the circuit from the capacity that stores the weight. As a next step the AM is fabricated and tested on our rapid prototyping platform RAPTOR2000 [6].

## References

1. Schreiter, J., Döge, J., Matolin, D., Schüffny, R., Heittmann, A., Ramacher, U.: Image segmentation by a PCNN with adaptive weights. Visualization, Imaging and Image Processing - VIIP 2003, Benalmádena, Spain (2003)
2. Heittmann, A., Ramacher, U.: Correlation-based feature detection using pulsed neural networks. IEEE International Workshop on Neural Networks for Signal Processing, Toulouse, France (2003)
3. Witkowski, U., Rückert U.: Pattern Synchronization for Associative Memory in pulse coded Neural Networks. Proceedings of the IEEE 47th Midwest Symposium on Circuits and Systems **2**. Hiroshima, Japan (2004) 381–384
4. Heittmann, A., Rückert, U.: Mixed Mode VLSI Implementation of a Neural Associative Memory. Analog Integrated Circuits and Signal Processing **30** (2002) 159–172
5. Rückert, U.: An Associative Memory with Neural Architecture and its VLSI Implementation. Proceedings of the Twenty-Fourth Annual Hawaii International Conference on System Sciences **1**. Los Alamitos: IEEE Computer Society Press (1991)
6. Kalte, H., Porrmann, M. and Rückert, U.: A Prototyping Platform for Dynamically Reconfigurable System on Chip Designs. In Proc. of the IEEE Workshop Heterogeneous reconfigurable Systems on Chip. Germany (2002)
7. Palm, G.: On associative memory. Biol. Cybernetics **36** (1980) 19–31
8. Shih-Chii L., Kramer, J., Indiveri, G., Delbrück, T., Douglas, R.: Orientation-Selective aVLSI Spiking Neurons. Adv. in Neural Inf. Proc. Systems **14** (2002)
9. Chicca E., Badoni D., Dante V., D'Andreagiovanni M., Gateano, Carota, L., Fusi, S., Del Giudice, P.: A VLSI Recurrent Network of Integrate-and-Fire Neurons Connected by Plastic Synapses With Long-Term Memory. IEEE Transactions On Neural Networks **14**, No. 5 (2003)

# Smart Sensing with Adaptive Analog Circuits

Guillermo Zatorre<sup>1</sup>, Nicolás Medrano<sup>1</sup>, Santiago Celma<sup>1</sup>,  
Bonifacio Martín-del-Brío<sup>2</sup>, and Antonio Bono<sup>2</sup>

<sup>1</sup> Electronic Design Group, Facultad de Ciencias, Pedro Cerbuna 12,  
50009 Zaragoza, Spain  
{gzatorre, nmedrano, 465249, scelma}@unizar.es

<sup>2</sup> Escuela Universitaria de Ingeniería Técnica Industrial,  
María de Luna 1, edif. Ada Byron,  
50018 Zaragoza, Spain  
{bmb, antoniob}@unizar.es

**Abstract.** This work shows the design and application of a mixed-mode analog-digital neural network circuit for sensor conditioning applications. The proposed architecture provides a high extension of the linear range for non-linear output sensors as negative temperature coefficient resistors (NTC) or giant magnetoresistive (GMR) angular position sensors, by using analog current-mode circuits with digital 8-bit weight storage. We present an analog current-based neuron model with digital weights, showing its architecture and features. By modifying the algorithm used in off-chip weight fitting, main differences of the electronic architecture, compared to the ideal model, are compensated. A small neural network based on the proposed architecture is applied to improve the output of NTC thermistors and GMR sensors, showing good results. Circuit complexity and performance make these systems suitable to be implemented as on-chip compensation modules.

## 1 Introduction

As it is well known, Artificial Neural Networks (ANNs) fit the transfer function of a system by means of a training process where input-output data pairs are iteratively presented. ANNs are implemented in several ways, depending on the application requirements. In systems where size, power consumption and speed are main constraints, electronic analog implementation is a suitable selection [1]. The present-day technology trend to shrinking bias voltages makes difficult to process high-resolution data codified in voltage-mode. In this case, current-mode processing gives better results at lower bias, reducing power consumption [2].

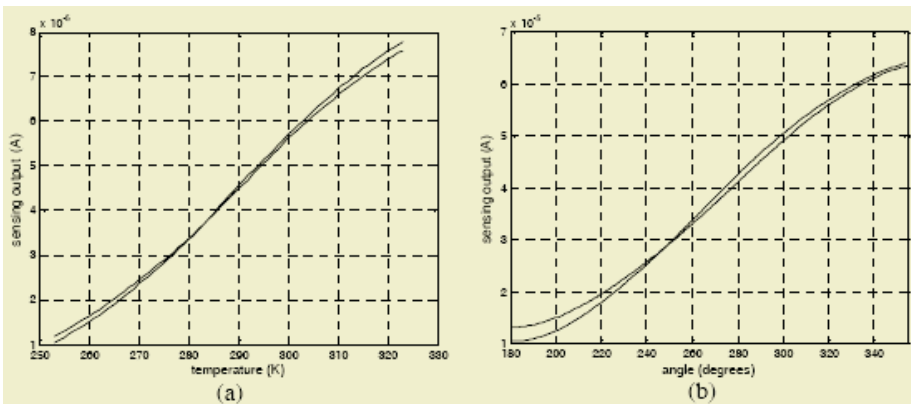
On the other hand, implementation of reliable long-term and mid-term analog programmable weights is hard because of mismatching and offsets. Due to the high accuracy of digital storage, combining both electronic technologies can improve the system performance. Previous works [3] have presented the use of mixed-mode multipliers in artificial neuron implementations, showing promising results applied to real problems.

ANNs can be used in adaptive sensor output improvement, applied to sensors that present nonlinear behavior, such as negative temperature coefficient resistors (NTC, Fig. 1a) or giant magnetoresistive sensors (GMR, Fig. 1b). Previous works show the use of these processing tools implemented in digital technology [4] or as analog circuits, but applied to classification tasks [5].

This paper shows the application of current-based mixed-mode adaptive circuits to the improvement of non-linear sensors output. In Section 2, the proposed current-based mixed-mode artificial neuron is presented and the simulated behavior is compared to the ideal one. In order to improve neuron operation, some changes in the training process are proposed. Section 3 shows an implementation of an adaptive linearization circuit based on this structure, showing the results achieved in two samples from two different sensors (NTC and GMR). Finally, some conclusions of this work are presented.

## 2 Electronic Processor

The proposed neuron architecture has been designed in current-mode using the 0.35 mm Austria Microsystems (AMS) design kit. Current-mode circuits are designed to use current as information carrier. Current-mode electronics show better performance than the corresponding voltage-mode circuits when low bias voltages and reduced power consumption are main requirements, representing a good option to be embedded as signal conditioning circuits into a sensor chip. The proposed neuron architecture consists of a current-mode four-quadrant analog-digital multiplier plus a current conveyor (CC) that performs the activation function. The main neuron building blocks are shown in Fig. 2. Fig. 2a shows the mixed-mode multiplier: Input current is multiplied by a factor in the range of  $\pm 1$  using a transistor-based R-2R current ladder [6], [7], controlled by the digital weight bits  $b_i$ . Resulting sign is selected modifying the current flow direction: if the weight is positive (sign bit is '0'), current



**Fig. 1.** (a) NTC behavior mounted on a resistive divider (two different samples); (b) GMR angular sensors behavior (two different samples)

flows directly to the multiplier; however, if the weight is negative (sign bit is ‘1’), current direction is inverted using the current follower described in [8]. The ideal multiplier output is

$$I_{out2} = I_0 \left( \sum_{n=1}^N \frac{b_n}{2^n} \right) \tag{1}$$

Real multiplier operation is described by

$$out = 0.974865wp - 0.0136726p \tag{2}$$

Where  $p$  is the current that inputs to the multiplier,  $w$  is the digital weight and  $out$  is the current output. Divergences between real and ideal multiplier operation are shown in Fig. 3.

The non-linear output is executed using a class AB CC (Fig. 2b). Fig. 4 shows circuit operation compared to the ideal hyperbolic tangent.

The most important neuron non-ideality is due to the output circuit. As Fig. 5 shows, circuit presents differences in the non-linear corners (upper and lower limits of the central range) plus an additional output offset. Both effects modify the operation of the network implemented with this processing element. In order to adjust the network weights, correcting the undesired effects, we have selected a perturbative algorithm [9] executed off-line: Network training is simulated on a computer; next,

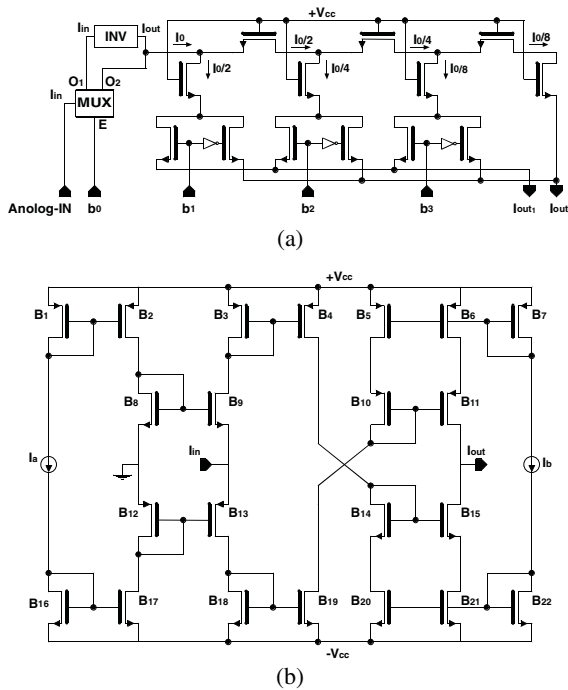
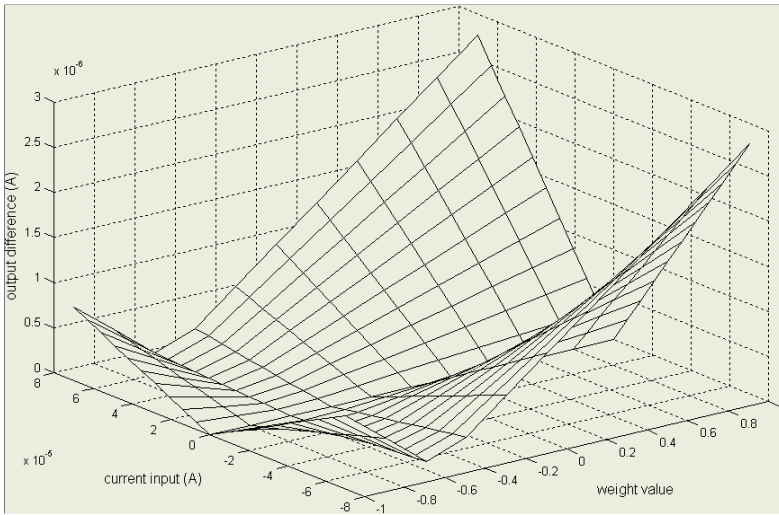
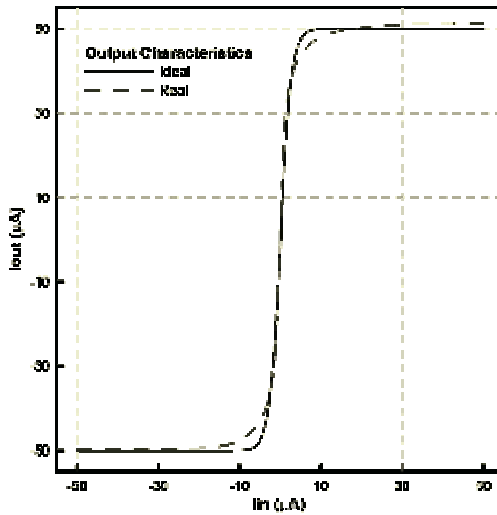


Fig. 2. Main neuron blocks: (a) analog-digital multiplier; (b) sigmoid output circuit





**Fig. 3.** Differences between the four quadrant mixed-mode multiplier operation and the ideal multiplication



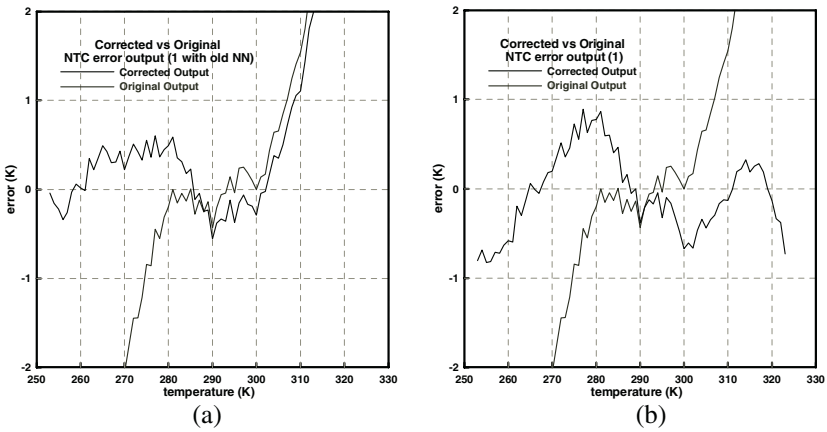
**Fig. 4.** Class AB current conveyor operation (continuous line) compared to the ideal one (dotted line)

trained weights are loaded into the digital storage blocks of the processing elements. The use of this algorithm is due to its robustness to neuron non-idealities compared to gradient descent-based techniques [10]. Moreover, we have verified that the learning algorithm improves its performance by simulating a half of the hidden layer neurons with the inverse of the real non-linear operation function.

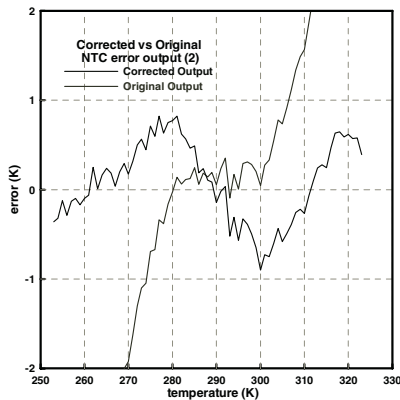
Once the desired performance is reached, the inverted output function is replaced by the real one in the corresponding simulated processors, changing the sign of the involved weights. A fast network re-training provides higher performances than using the actual output function in the training phase (Fig. 5), reducing significantly training times.

### 3 Applications

The proposed processing element has been used to extend the linear range of two different sensors: two samples of NTC resistors mounted on a resistive divider (see output in Fig. 1a) and two samples of GMR angular position sensors (see output in Fig. 1b).



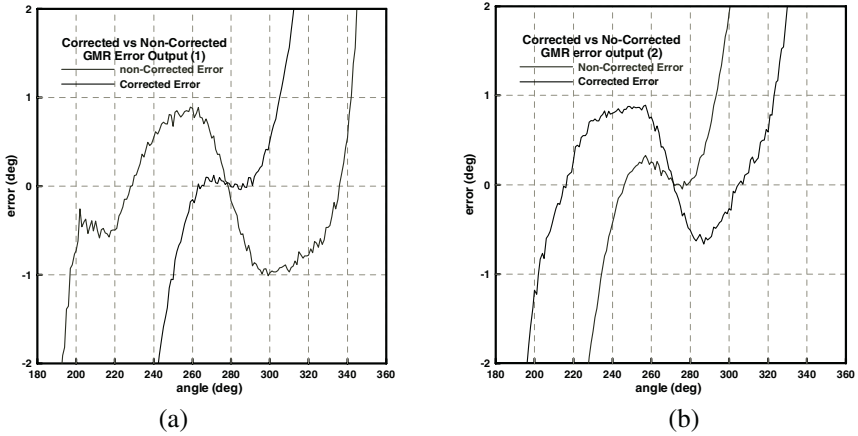
**Fig. 5.** Network output error (continuous line) compared with the sensor output error for an NTC when (a) training is developed normally and (b) a half of the hidden neurons are simulated using the inverse of the actual output



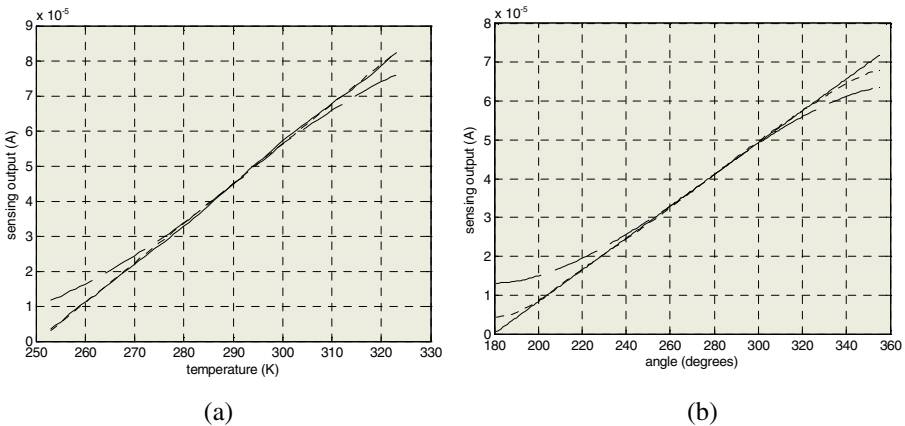
**Fig. 6.** Output network error (continuous line) compared to the second NTC sensor output. Error remains under the limits

### 3.1 Linearizing NTC Sensors Output

A neural network with one input (nonlinear sensor output), one output (correction that must be added to the sensor output) and two hidden neurons is used to extending the limited linear range of two different NTC sensors, keeping the error lower than 1K degree. Each sensor has a data set consisting of 71 temperature-voltage samples in the range of 252-323 K degrees. A set of 10 patterns are reserved for network validation purposes; the rest are applied in the training cycle. Sensor output errors are compared to the corrected output errors in Fig. 5b and 6. Fig. 8a shows the resulting corrected output of one of the NTC sensors compared to the sensor output and the ideal one.



**Fig. 7.** (a) Output network error (continuous line) compared to the first GMR sensor output (dotted line); (b) Output network error (continuous line) compared to the second GMR sensor output (dotted line)



**Fig. 8.** Output network (dotted line) compared to the sensor output (dashed line) and ideal straight output (continuous line) for (a) An NTC sensor and (b) a GMR sensor

### 3.2 Linearizing GMR Sensors Output

As in the previous case, two sensor samples are used to test the goodness of the proposed technique to extend the linear range of a sensor output. In this case, the number of patterns available along the sensor span (180-355 angular degrees) is 175; 20 of them are kept for the validation stage. Neural network consists of 1 input, 1 output and 4 processing elements in a hidden layer. Fig. 7 show the errors achieved using this technique compared to the output errors for both sensor samples. Fig. 8b show the resulting corrected output compared to the sensor output and the ideal linear one.

## 4 Conclusions

This paper shows the application of adaptive mixed-mode circuits to the improvement of non-linear sensors output. The proposed processing element, designed using the 0.35  $\mu\text{m}$  Austria Microsystems design kit, consists of a current-based artificial neuron using an analog-digital multiplier plus a current conveyor that implements the non-linear output function. Digital weight codification allows the long-term storage with the selected accuracy. Our application uses 8-bit precision, multiplying the input currents using the mixed-mode multiplier. The use of a perturbative training algorithm yields a better weight fitting than gradient descent-based techniques. Moreover, changing the sign of the non-linear output function in a half of the neurons in the simulated training phase improves dramatically the system performance.

The application of this system to the output linearization of several NTC and GMR sensors show very good results. Table 1 presents the linear output range extension achieved with this technique, assuming a maximum error of 1 K for NTC temperature sensors and 1° for angular position GMR sensors.

**Table 1.** Results achieved extending the linear range of each sensor using the proposed neuron model

| Sensor | Linear range | Extended range | %   |
|--------|--------------|----------------|-----|
| NTC #1 | 274-308 K    | 252-323 K      | 108 |
| NTC #2 | 274-308 K    | 252-323 K      | 108 |
| GMR #1 | 252-305 deg  | 198-328 deg    | 145 |
| GMR #2 | 234-292 deg  | 202-322 deg    | 107 |

## Acknowledgments

This work has been supported by CICYT (Projects TIC2002-00636 and TIC02-00636), and by the University of Zaragoza (UZ2002-TEC-05).

## References

1. M. Jabri, R. J. Coggins, B. G. Flower: Adaptive Analog VLSI Neural Systems, Chapman & Hall, (1990)
2. C. Toumazou, F. J. Lidgley, D. G. Haigh: Analogue IC Design: The Current-Mode Approach, IEE Circuits and Systems Series 2 (1990)
3. C. Lajusticia, N. Medrano, G. Zatorre, B. Martín: Applying Non-Ideal Mixed Analog-Digital Multipliers in Electronic Processing Circuits Based on Neural Networks, Proc. 2003 IEEE Conference on Emerging Technologies and Factory Automation, ETFA, vol. 2, 362-367 (2003)
4. N. Medrano, B. Martín: A sensor conditioning method for microprocessor based applications, XVIII Design of Circuits and Integrated Systems Conference, DCIS, Santander (Spain), 725-730, (2002)
5. M. Milev, M. Hristov: Analog Implementation of ANN with Inherent Quadratic Nonlinearity of the Synapses, IEEE Trans. on Neural Networks, vol. 14, n. 5, 1187-1200 (2002)
6. K. Bult, G. Geelen: An Inherentially Linear and Compact MOST-only Current Division Technique, IEEE J. Solid-State Circuits, vol. 27, n. 12, 1730-1735 (1992)
7. M. T. Sanz, B. Calvo, S. Celma, C. Morán: A Digitally Programmable VGA, IEEE Midwest Symposium on Circuits and Systems, MWSCAS, 602-605 (2001)
8. G. Zatorre, P. A. Matínez, C. Aldea: A New CMOS Class AB Current-Differential Amplifier, Seminario Annual de Automática, Electrónica Industrial e Instrumentación, SAAEI, CD-ROM (2003)
9. M. Jabri, B. Flower: Weight perturbation: An optimal architecture and learning technique for analog VLSI feedforward and recurrent multilayer networks, IEEE Transactions on Neural Networks, vol. 3 no. 1, 154-157 (1992)
10. M. Valle: Analog VLSI implementation of artificial neural networks with Supervised On-Chip Learning, Analog Integrated Circuits and Signal Processing, vol. 33, 263-287 (2002)

# Spiking Neurons Computing Platform

Eduardo Ros<sup>1</sup>, Eva M. Ortigosa<sup>1</sup>, Rodrigo Agís<sup>1</sup>, Richard Carrillo<sup>1</sup>,  
Alberto Prieto<sup>1</sup>, and Mike Arnold<sup>2</sup>

<sup>1</sup> Department of computer and Technology, University of Granada,  
E-18071, Granada, Spain

{eros, eva, ragis, rcarrillo}@atc.ugr.es

<sup>2</sup> CNL, Salk Institute of Biological Studies, La Jolla,

CA 92037, USA

mikea@salk.es

**Abstract.** A computing platform is described for simulating arbitrary networks of spiking neurons in real time. A hybrid computing scheme is adopted that uses both software and hardware components. We focus on conductance-based models for neurons that emulate the temporal dynamics of the synaptic integration process. We have designed an efficient computing architecture using reconfigurable hardware in which the different stages of the neuron model are processed in parallel (using a customized pipeline structure). Further improvements occur by computing multiple neurons in parallel using multiple processing units. The computing platform is described and its scalability and performance evaluated. The goal is to investigate biologically realistic models for the control of robots operating within closed perception-action loops.

## 1 Introduction

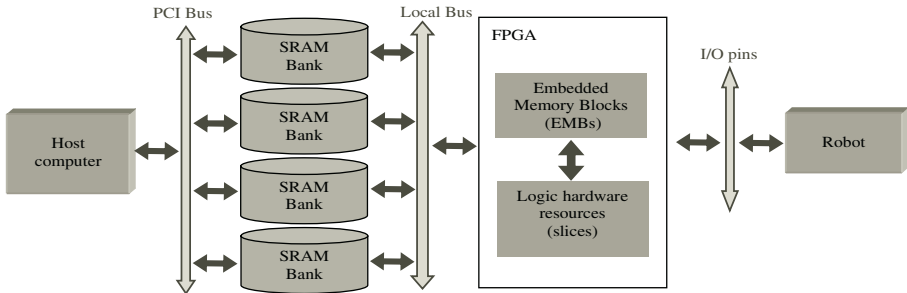
A number of studies have investigated the use of event-driven computation schemes for the efficient simulation of spiking neurons [1, 2, and 3]. The computation in these schemes occurs only at the arrival of each pre-synaptic spike packet and requires that two pieces of information, i.e. is the effect of the pre-synaptic spike on the neuron's membrane potential. Second is the probability that the neuron fires in response to the pre-synaptic spike.

We are interested in synchronization phenomena within neural populations, for which important features are the temporal dynamics of the synapses [4, 5]. The gradual injection of charge in the membrane potential due to an input spike is captured by the *Spike Response Model* (SRM) [6]. In its general form, this model is not amenable to an event-driven approach and the standard integration approach is computationally inefficient when using conventional computational architectures (single or multi processor platforms) [7]. A number of studies implement a specific type of SRM [8, 9, 10, 11, and 12] according to the neuron model proposed by Eckhorn et al. [13]. Here we present a hybrid hardware and software simulation scheme for networks of SRM neurons. On this scheme we can define the temporal dynamics of synaptic filters, the membrane capacitance, the resting potential and time constant as well as the synaptic weights. The approach is based on reconfigurable hardware (FPGA device); the easy modification of the computing modules facilitates the exploration of different neural models.

## 2 Computing Scheme

The hardware component consists of an add-on board providing memory resources and an FPGA device that works as a neuro-processor (Fig. 1). The software component runs on the host computer and the communication between the two is achieved via the PCI bus. Currently the hardware component is restricted to compute the neural integration. The software component is responsible for maintaining the network connectivity, for routing spikes between neurons, and for learning.

Communication between the hardware and software components is restricted to spike events. The software component generates pre-synaptic spikes and the hardware component generates post-synaptic spikes. The pre-synaptic event is a structure  $I(t_s, A_{neu}, w_{exc}, w_{inh})$ , where  $t_s$  is the time of the spike relative to the current epoch given as an integer between 0 and  $(N_t - 1)$ ,  $A_{neu}$  is the address of the post-synaptic or target neuron, and  $(w_{exc}, w_{inh})$  are the synaptic efficacies. The post-synaptic event is another structure  $Output(t_s, A_{neu})$ , where  $t_s$  is the time of the spike relative to the current epoch given as an integer between 1 and  $N_t$ , and  $A_{neu}$  is the address of the post-synaptic or generating neuron. With these structures each neuron can have only two types of meta-synapses, one that is excitatory and one that is inhibitory. These meta-synapses receive the input events that can accumulate contributions from several cells.



**Fig. 1.** Software-Hardware communication scheme. We use on-board memory SRAM banks to interface the software and hardware components

## 3 Neuron Model

The SRM consists of terms to calculate the membrane potential together with a spike generation mechanism [6]. For a neuron with excitatory and inhibitory synapses, the membrane potential  $V_x$  is given by:

$$V_x = V_x - \frac{1}{\tau_r} (V_x - V_{resting}) + G \cdot S_{exc} \cdot (U_{max} - V_x) - G \cdot S_{inh} \cdot (V_x - U_{min}) \quad (1)$$

Where  $\tau_r$  is the time constant of the resting term,  $S_{exc}$  and  $S_{inh}$  are the synaptic contributions and  $G$  is a gain term that modulates the refractory period modelling with ex-

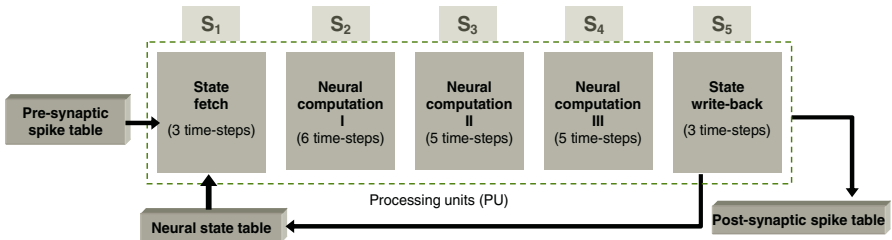
ponential functions of Gerstner [6].  $U_{max}$  and  $U_{min}$  are the maximum and minimum values of the membrane potential.  $V_{resting}$  is the membrane resting potential.

The neuron model is implemented using a processing unit with the following components; (1) two input IIR spike response filters with different time constants for GABAergic synapses and AMPA synapses. (2) an integrator register that stores the membrane potential, which is computed at each time step. (3) one output IIR filter whose time constant controls the post-firing refractory period through the gain term  $G$ . (4) one spike generation mechanism that generates spikes according to the membrane potential.

## 4 Parallel Computing Strategies

A sequential implementation of the neuron model requires 17 time steps for all the computations. This is optimised by adopting a design strategy that efficiently exploits the parallel computing resources of FPGA devices. Implementing algorithmic parallelism, or pipelining, is a frequently used technique in hardware design to reduce the number of time steps needed to perform complex operations. In order to exploit the inherent parallelism of the FPGA devices, we have designed a pipelined computing structure with  $N_{stages} = 5$  stages (S1 to S5 in Fig. 2) to calculate expression (1). These stages are:

1. *State fetch*: retrieves an input spike from the pre-synaptic spike table (stored in the embedded memory blocks, EMB) and the neural state for the corresponding neuron from the neural state table (stored in another EMB).
2. *Neural computation I*: calculate the synaptic conductance using the IIR filters ( $S_{exc}$  and  $S_{inh}$ ).
3. *Neural computation II*: calculate the membrane potential ( $V_x$ ).
4. *Neural computation III*: calculate the resting and refractory components ( $G$ ), and generate a post-synaptic spike if required.
5. *State write-back*: store the output spike and the neural state in the tables.



**Fig. 2.** Pipeline computing structure. We define 5 pipeline stages. The computing cycles of each stage are indicated in the figure

Besides the internal parallelism of each data path, the computing platform uses several processing units (PU) in the FPGA that run in parallel. We have adopted a scal-



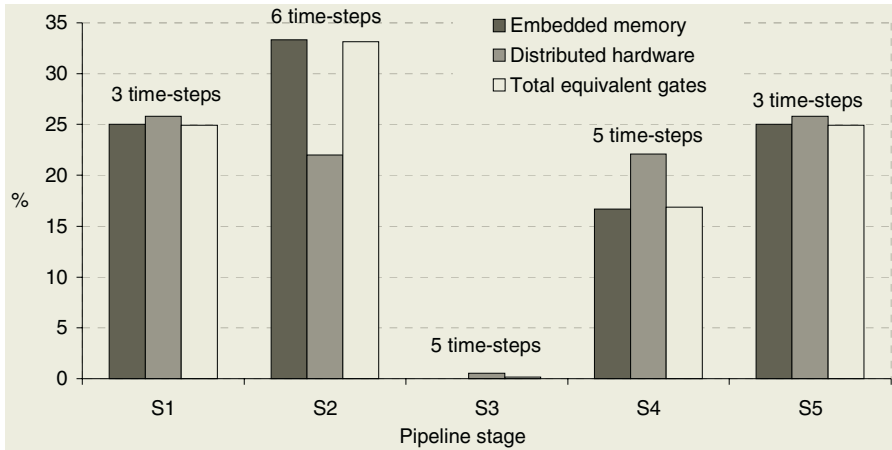
able computing scheme by splitting the embedded memory resources into customized blocks assigned to different computing units.

### 5 Computational Resource Consumption

The computational load described in section 3 has been distributed into different pipelined stages to allow parallel processing along the data path. We have used the RC1000 board of Celoxica [15] as prototyping platform to evaluate the computation scheme. This is a PCI board with 4 banks of SRAM memory on board (2MB each) and a Xilinx device with two million of gates (Virtex- 2000E) [16]. Table 1 summarizes the hardware resources required for the whole computing system with a single processing unit. It can be seen that a single processing unit consumes less than 25% of the hardware resources (slices) of the device therefore we can implement up to 4 units running in parallel on a single device.

**Table 1.** The table shows the memory and computational resources consumed by a single processing unit compared with the available resources on a Virtex 2000-E device

| Embedded memory blocks | Slices            | Equivalent gates |
|------------------------|-------------------|------------------|
| 80 out of 180          | 4581 out of 19200 | 1389247          |



**Fig. 3.** Computational resources of each pipeline stage. Each bar represents the resources percentage of a single processing unit

We can see the resources consumption of the different pipeline stages in Fig. 3.

S2 is the most expensive stage since it includes two IIR filters computed in parallel (with their multiplicative terms). These multiplications require high precision (14 bits for the input terms) to make the filters stable, hence this stage includes two fix point

multipliers of 14 bits. Furthermore, this stage needs to store the temporal states of the filters (two per neuron), which requires significant memory resources.

The number of time steps consumed in each pipeline stage is shown above the bars in Fig. 3. Since all the stages work in parallel synchronously, the data throughput is limited by the longest stage (S2 in this case). The longest pipeline stage consumes  $NTS_{lps} = 6$  time steps. Therefore each processing unit produces an updated neuron state every  $NTS_{lps}$  steps. There is a latency ( $L$ ) of 30 time steps in computing a neuron state ( $L = N_{stages} \cdot NTS_{lps}$ ). If  $F_{clk}$  is the clock frequency, the data throughput or number of neural state updates per second is given by:

$$R_{neuron} = \frac{F_{clk}}{NTS_{lps}} \quad (2)$$

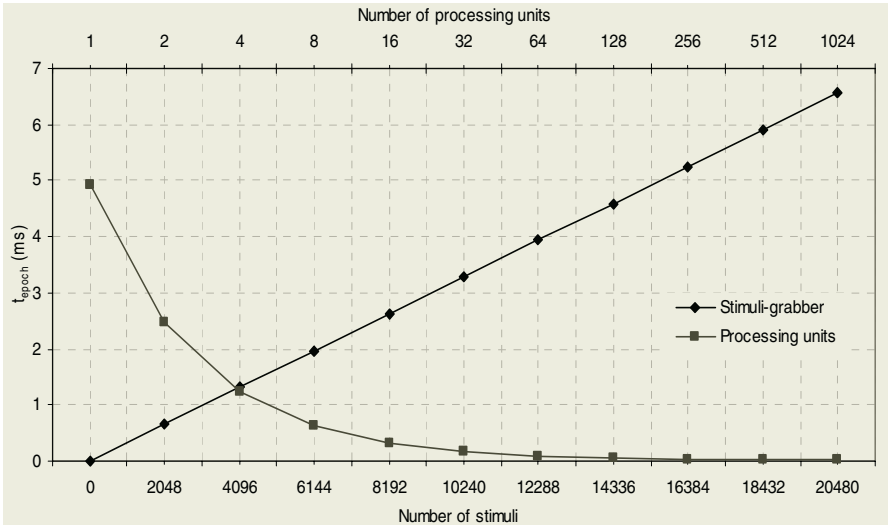
For a circuit design with  $NTS_{lps} = 6$  time steps and  $F_{clk} = 25$  MHz, then  $R_{neuron} = 4.16 \cdot 10^6$  updates/second. Using the FPGA of the RC1000 prototyping platform, we have implemented and tested the system with 1024 neurons and we are able to place up to 4 cores on the chip. The on-chip resources do not depend on the number of neurons simulated or the number of input/output spikes. This is so because they are stored into on-board memory and they are transferred to the FPGA in packets.

## 6 Computing Performance

In Section 2 we described the computing scheme based on epochs of  $N_t$  time cycles. This computing strategy reduces the communications over the PCI-bus. The computing scheme requires two main modules that run in parallel:

1. Stimuli grabber circuit. It loads the pre-synaptic spikes to be computed in the next cycle.
2. Processing units. They compute the neuron states iteratively taking in to account the inputs stimuli.

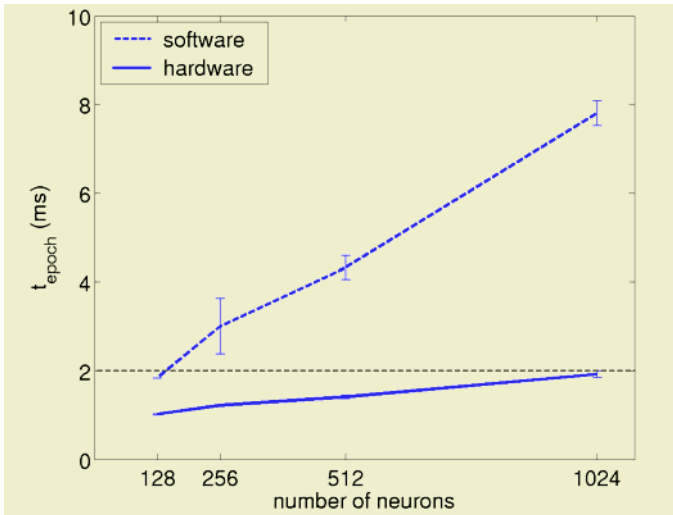
If the firing probabilities of the cells are low, we can assume that the processing units take longer than the stimuli grabber. If we choose the following configuration:  $N_t = 20$  cycles,  $N_{neurons} = 1024$  neurons,  $NTS_{lps} = 6$  time steps and  $F_{clk} = 25$  Mhz, then the time taken by the hardware using a single processing unit to compute one epoch is  $t_{epoch} \approx 4.94$  ms. The total time to compute an epoch is greater than the time taken by the hardware because it is necessary to include the time to communicate the pre- and post-synaptic spikes and the time to share data between and synchronise the threads in the software component. A suitable integration step for biologically plausible neurons based on the SRM is  $100\mu s$ . For a simulation to be in real time, an epoch of 20 time cycles must be computed in less than 2 ms. To achieve this we use several computing processing units in parallel. The Fig. 4 plot illustrates that the co-processor board can compute 1024 neurons in real-time when receiving less than 6144 stimuli per epoch (i.e.  $20 \times 100\mu s$  cycles/epoch in less than 2ms). The optimal configuration is with 4 processing units on the chip processing 4096 stimuli per epoch.



**Fig. 4.** Scalability study. Optimal configuration for real time processing omitting the transmission time through the PCI bus (100 MB/s of measured bandwidth using DMA). 1024 output events take about 0.1 ms and 2048 stimuli take 0.3 ms

## 7 Real-Time Cerebellar Simulations

The effective performance of the system depends on the network connectivity and load, emphasizing the importance of testing it within a biologically realistic context.



**Fig. 5.** Performance on cerebellar model. The speedup of the hardware co-processor over a software co-processor with respect to the total number of neurons

For this we use a model of the olivary-cerebellar system applied to a simple tracking task (abstracted as visual smooth pursuit). Full details of the model and its application to visual smooth pursuit are given in [17]. This model explores the role of climbing fibre-modulated plasticity in the cerebellar cortex and nuclei for cerebellar learning, however the performance is measured after training has stabilised and the learning rules are inactivated, in order to ensure that the computation cost is not bound by the learning. All of the neurons are calculated in the co-processor, except for the mossy fibre and feedback inputs. The cerebellar model is composed by 1000 granular cells (1-100 Hz), 2 Purkinje cells (30-120 Hz), 4 Interneurons (5-60 Hz), 4 Golgi cells (50-60 Hz), 2 Nuclear neurons (30-70 Hz), 2 Inferior Olive cells (0-10 Hz) and 21 Mossy inputs (20-80 Hz). The average activity of the network is around 2.4 Hz). The simulation speedup achieved by the hardware platform is shown in Fig. 5.

## 8 Discussion

With the current prototyping platform (RC1000) we can implement up to four processing units and compute in real time (with an integration time step of 100  $\mu$ s) up to 1024 neurons receiving 4096 input stimuli. But we have studied the scalability of the system and the design can be easily adapted to more powerful devices for real time computing of larger neural systems. The average network activity is critical, i.e. if the number of stimuli received is too high the “computing” time increases. With an integration time step of 100  $\mu$ s however, we expect there to be a low activity per time-step. In the cerebellar model, the average number of pre-synaptic spikes per neuron per 2 ms time-step is in the presented simulation. In fact, the presented approach uses an iterative computing strategy in the FPGA and an event-driven communication scheme between the host and the co-processing board. This iterative computing mode allows the exploration of specific issues difficult to track within an event-driven computing scheme.

## Acknowledgments

This work has been supported by the EU project SpikeFORCE (IST-2001-35271) and the National Spanish Grant (DPI 2004-07032).

## References

1. Mattia, M., Guidice, P. D.: Efficient event-driven simulation of large networks of spiking neurons and dynamical synapses. *Neural Computation*, vol. 12, (2000) 2305-2329
2. Reutimann, J., Guigliano, M., Fusi, S.: Event-driven simulation of spiking neurons with stochastic dynamics. *Neural Computation*, vol. 15, (2003) 811-830
3. Delorme, A., Thorpe, S.: SpikeNET: An event-driven simulation package for modelling large networks of spiking neurons. *Network: Computation in Neural Systems*, vol. 14, (2003) 613-627

4. Eckhorn, R., Bauer, R., Jordan, W., Brosh, M., Kruse, W., Munk, M., Reitboeck, H.J.: Coherent oscillations: A mechanism of feature linking in the visual cortex?. *Biol. Cyber.* Vol. 60, (1988) 121-130
5. Eckhorn, R., Reitboeck, H.J., Arndt, M., Dicke, P.: Feature Linking via Synchronization among Distributed Assemblies: Simulations of Results from Cat Visual Cortex. *Neural Computation*, vol. 2, (1990) 293-307
6. Gerstner, W., Kistler, W.: *Spiking Neuron Models*, Cambridge University Press, 2002.
7. Jahnke A., Schoenauer, T., Roth, U., Mohraz, K., Klar, H.: Simulation of Spiking Neural Networks on Different Hardware Platforms. *ICANN97*, (Springer-Verlag), *Lecture Notes in Computer Science*, vol. 1327, (1997) 1187-1192
8. Hartmann, G., Frank, G., Schaefer, M., Wolff, C.: SPIKE128K- An Accelerator for Dynamic Simulation of Large Pulse-Coded Networks. *MicroNeuro'97*, (1997) 130-139
9. Shaefer, M., Schoenauer, T., Wolff, C., Hartmann, G., Klar, H., Rueckert, U.: Simulation of Spiking Neural Networks – architectures and implementations. *Neurocomputing*, vol. 48, (2002) 647-679
10. Janke, A., Roth, U., Klar, H.: A SIMD/dataflow architecture for a neurocomputer for spike processing neural networks (NESPINN). *Proc. MicroNeuro'96*, (1996) 232-237
11. Schoenauer, T., Atasoy, S., Mehrtash, N., Klar, H.: NeuroPipe-Chip: A Digital Neuro-Processor for Spiking Neural Networks. *IEEE Trans. Neural Networks*, vol. 13(1), (2002) 205-213
12. Mehrtash, N., Jung, D., Hellmich, H.H., Schoenauer, T., Lu, V.T., Klar, H.: Synaptic Plasticity in Spiking Neural Networks (SP2INN): A System Approach. *IEEE Transactions on Neural Networks*, vol. 14 (5) 2003
13. Eckhorn, R., Reitboeck, H.J., Arndt, M., and Dicke, P.: Feature linking via stimulus evoked oscillations: Experimental results from cat visual cortex and functional implication from a network model. *Proc. ICNN I*, (1989) 723–720
14. Hill, J., McColl, W., Stefanescu, D., Goudreau, M., Lang, K., Rao, S., Suel, T., Tsantilas, T., Bisseling, R.: BSPLib: the BSP Programming Library,” *Parallel Computing*, vol. 24(14), (1998) 1947-1980
15. Celoxica, 2001-2004. [Online]. Available: <http://www.celoxica.com>
16. Xilinx, 1994-2003. [Online]. Available: <http://www.xilinx.com>
17. Arnold, M.: Feedback learning in the olivary-cerebellar system: PhD Thesis, The University of Sydney, 2001.

# Inter-spike-intervals Analysis of Poisson Like Hardware Synthetic AER Generation

A. Linares-Barranco<sup>1</sup>, M. Oster<sup>2</sup>, D. Cascado<sup>1</sup>, G. Jiménez<sup>1</sup>,  
A. Civit<sup>1</sup>, and B. Linares-Barranco<sup>3</sup>

<sup>1</sup> Arquitectura y Tecnología de Computadores, Universidad de Sevilla,  
Av. Reina Mercedes s/n, 41012-Sevilla, Spain  
alinares@atc.us.es

<sup>2</sup> Institute of Neuroinformatics, UNI - ETH Zürich,  
Winterthurerstrasse 190, CH-8057 Zurich, Switzerland

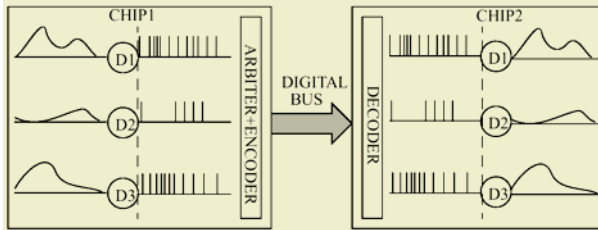
<sup>3</sup> Instituto de Microelectrónica de Sevilla, CSIC,  
Av. Reina Mercedes s/n, Edificio CICA, 41012-Sevilla, Spain

**Abstract.** Address-Event-Representation (AER) is a communication protocol for transferring images between chips, originally developed for bio-inspired image processing systems. Such systems may consist of a complicated hierarchical structure with many chips that transmit images among them in real time, while performing some processing (for example, convolutions). In developing AER based systems it is very convenient to have available some kind of means of generating AER streams from on-computer stored images. In this paper we present a hardware method for generating AER streams in real time from a sequence of images stored in a computer's memory. The Kolmogorov-Smirnov test has been applied to quantify that this method follows a Poisson distribution of the spikes. A USB-AER board and a PCI-AER board, developed by our RTCAR group, have been used.

## 1 Introduction

Address-Event-Representation (AER) was proposed in 1991 by Sivilotti [1] for transferring the state of an array of analog time dependent values from one chip to another. It uses mixed analog and digital principles and exploits pulse density modulation for coding information. Figure 1 explains the principle behind the AER basics. The Emitter chip contains an array of cells (like, for example, a camera or artificial retina chip) where each pixel shows a continuously varying time dependent state that change with a slow time constant (in the order of *ms*). Each cell or pixel includes a local oscillator (VCO) that generates digital pulses of minimum width (a few nano-seconds). The density of pulses is proportional to the state of the pixel (or pixel intensity). Each time a pixel generates a pulse (which is called "Event"), it communicates to the array periphery and a digital word representing a code or address for that pixel is placed on the external inter-chip digital bus (the AER bus). Additional handshaking lines (Acknowledge and Request) are also used for completing the asynchronous communication. The inter-chip AER bus operates at the maximum

possible speed. In the receiver chip the pulses are directed to the pixels whose code or address was on the bus. This way, pixels with the same code or address in the emitter and receiver chips will “see” the same pulse stream. The receiver pixel integrates the pulses and reconstructs the original low frequency continuous-time waveform. Pixels that are more active access the bus more frequently than those less active.



**Fig. 1.** Illustration of AER inter-chip communication scheme

Transmitting the pixel addresses allows performing extra operations on the images while they travel from one chip to another. For example, inserting properly coded EEPROMs allows shifting and rotation of images. Also, the image transmitted by one chip can be received by many receiver chips in parallel, by properly handling the asynchronous communication protocol. The peculiar nature of the AER protocol also allows for very efficient convolution operations within a receiver chip [2].

There is a growing community of AER protocol users for bio-inspired applications in vision and audition systems, as demonstrated by the success in the last years of the AER group at the Neuromorphic Engineering Workshop series [3]. The goal of this community is to build large multi-chip and multi-layer hierarchically structured systems capable of performing complicated array data processing in real time. The success of such systems will strongly depend on the availability of robust and efficient development and debugging AER-tools. One such tool is a computer interface that allows not only reading an AER stream into a computer and displaying it on its screen in real-time, but also the opposite: from images available in the computer’s memory, generate a synthetic AER stream in a similar manner as would do a dedicated VLSI AER emitter chip [4][6].

In the following sections we present the hardware implementation of one of the already existing methods for synthetic AER generation [9][10][11], and we evaluate the nature of the distribution of the events respect to the inter spike intervals (ISIs). For this analysis we have used a hardware USB-AER interface developed by RTCAR group.

## 2 Synthetic AER Generation by Hardware

One can think of many software algorithms that would transform a bitmap image into an AER stream of pixel addresses [5]. At the end, the frequency of appearance of the address of a given pixel must be proportional to the intensity of that pixel. Note that the precise location of the address pulses is not critical. The pulses can be slightly

shifted from their nominal positions because the AER receivers will integrate them to recover the original pixel waveform.

From the software methods already proposed, we present the hardware implementation in VHDL of the Random method. Some modifications have been made to reduce the resources needed in the FPGA.

This method sends a sequence of addresses to the AER bus without any buffer. Next event to be sent is selected in real-time from an image composed by  $N \times N$  pixels, with  $N$  a power of 2, and each pixel can have up to  $k$  grey levels, with  $k=255$  in this implementation. Each event needs a time to be sent. Let's call it time slot. The algorithm would implement a particular way of selecting the next address to be transmitted, through the AER bus, in the current time slot.

At each time slot an event could be placed in the AER bus, or not. This method uses a Linear Feedback Shift Register (LFSR) [7][8] for selecting the pixel of the image in charge of sending an event, and also to decide if the event is going to be sent or not. The LFSR has a resolution of  $\log(N \times N \times k)$ , so the random number obtained for each time slot is divided into:

1. An address for selecting a pixel of the image and
2. A gray level value.

Thanks to the LFSR, each gray level value of each pixel is generated only once. If a pixel in the image has a value  $p$ , then the method will validate  $p$  events along time and will send those  $p$ , from the  $k$  possible, for this pixel. They will not be perfectly equidistant in time, but in average they will be reasonably well spaced. This method is very simple to be implemented in hardware. Next Sections explains in more details the implementation issues for this method.

### 3 Random Method

This method is an implementation of Linear Feedback Shift Register (LFSR) based random number generators. LFSR random number generators are based on a linear recurrence of the form:

$$x_n = (a_1 x_{n-1} + \dots + a_k x_{n-k}) \bmod 2 \quad (1)$$

where  $k > 1$  is the order of the recurrence,  $a_k = 1$ , and  $a_j \in \{0, 1\}$  for each  $j$ . This recurrence is always purely periodic and the period length of its longest cycle is  $2^k - 1$  if and only if its characteristic polynomial

$$P(z) = -\sum_{i=0}^{k-1} a_i z^{k-i} \quad (2)$$

is a primitive polynomial over the Galois field with 2 elements.

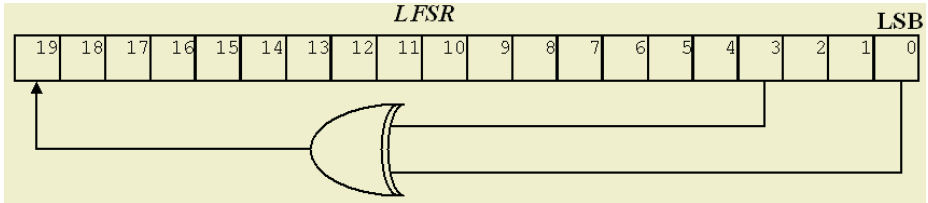
With these premises and limiting the maximum number of address events necessary to transmit an image, we know the number of bits needed for the LFSR and the primitive polynomial. For this implementation, the limit corresponds with a  $64 \times 64$  image of 256 gray levels, what implies a 20-bit LFSR.



The characteristics polynomial  $P(z)$  used for 20 bits is:

$$P(z) = z^{20} + z^{17} + 1 \tag{3}$$

which corresponds to the LFSR of Figure 2.



**Fig. 2.** Linear Feedback Shift Register for random synthetic AER generation

where all bits are ‘1’ after a reset, what is the seed of the random numbers generator. This way, the 20-bit numbers are divided in two parts: the gray level correspond with the 8 more significant bits, and the address of the pixel in the image are the other 12 bits. The method works as follows:

- For each time slot, the LFSR generates a random 20-bit number.
- The 12 less significant bits are used to address the gray level of the image stored in memory.
- Once addressed that pixel, its gray level is compared with the 8 more significant bits of the LFSR.
- If the gray level of the pixel is greater or equal to the 8 MSB of the LFSR, an event is transmitted with the 12 LSB of the LFSR as the address.
- In the other case, no event is produced for this time slot.

The LFSR ensures that each possible event of each pixel is obtained from the LFSR only once per each  $(2^{20}-1)$  time slots.

### 4 Inter-spikes-intervals Distribution Analysis

In this section we will compare the Inter-Spike-Intervals (ISIs) of this hardware synthetic AER generation method with the normalized distribution that it should have, using the Kolmogorov-Smirnov statistical test.

In neuro-inspired systems, signals can often be modelled by a Poisson distribution [12][13]. A Poisson distribution can be described by the following formula [14]:

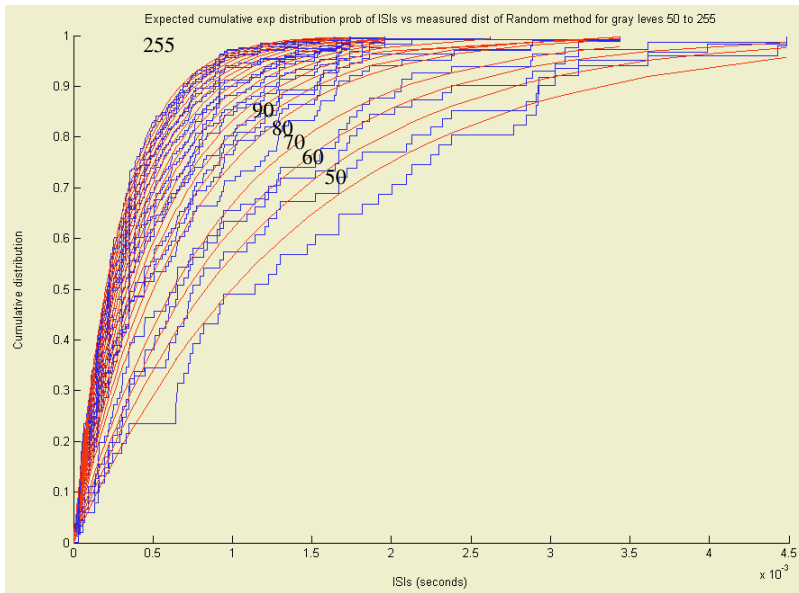
$$P_n(T) = \frac{(\lambda T)^n}{n!} e^{-\lambda T} \tag{4}$$

where P is the probability of having n events the interval time T. The distribution of ISIs is the probability that no event occurs in the interval:

$$P_0(T) = e^{-\lambda T} \tag{5}$$

which is the exponential distribution.

An USB-AER board with a Spartan II 200 FPGA has been used for the Random method, configured with different gray values for the same pixel address. And a PCI-AER board that captures events and their timestamp, controlled through MATLAB, has been also used to capture the ISIs. Figure 3 shows the cumulative probabilistic distribution of ISIs: the expected exponential distribution versus the measured distribution generated by the Random method ISIs for gray levels 50 to 255, 10 by 10. For a high gray level, the distributions are close to each other, what implies that the spike train generated with the Random method is close to the Poisson distribution.



**Fig. 3.** Expected cumulative Exponential ISIs distribution versus measured ISIs distribution generated by the Random method for gray levels

We use the Kolmogorov-Smirnov (KS) test to quantify how good the observed distribution of ISIs follows the theoretical exponential distribution. Figure 4 shows the result of applying the KS test to the Random distribution obtained with different gray levels. The test is passed if the result is below 5%. It is shown that for small gray levels, behaviour is not much Poisson like. This is due to the LFSR because all the possible numbers obtained from the LFSR are used to produce a sequence of events for just one frame, resulting in only few different ISIs for a pixel of low intensity. We will address this problem by increasing the number of bit in the LFSR, so the shift register has a longer period than a frame and has a different initialization at the start of each frame.

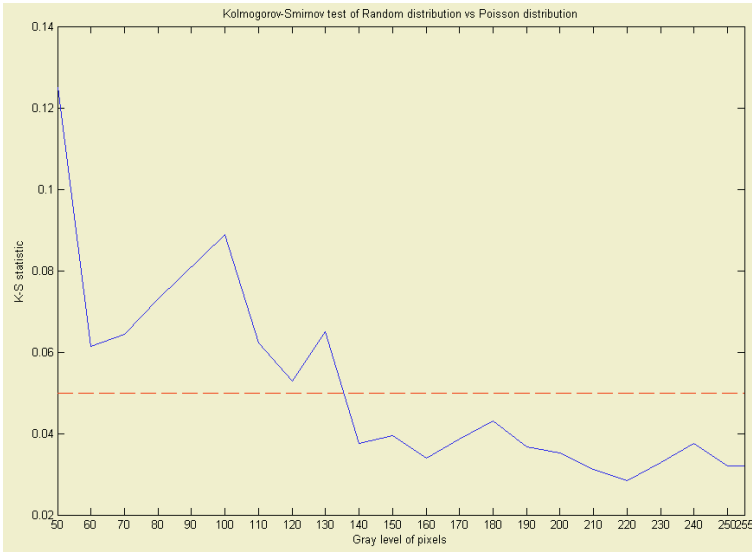


Fig. 4. Kolmogorov-Smirnov test results for Random method distribution

## 5 Conclusions

We have presented a hardware implementation of the Random method. We have demonstrated that it has a really well similar behaviour to the Poisson distribution. Therefore, it could be a very realistic method to be used for neuro-inspired systems, as AER systems. But for low frequency of events, this method becomes worse and worse for this kind of systems. So it could be improved in this way.

In contrast to existing AER generators like [15], this USB-AER board directly generates Poisson spike trains instead of sequencing raw spike trains.

The USB-AER board running with the Random method for AER Poisson synthetic generation, and the PCI-AER board with the MATLAB interface, are a useful tool for testing and debugging AER system. These two boards have been presented in the paper title “Test Infrastructure for Address-Event-Representation Communications” of this proceeding.

## Acknowledgements

This work was in part supported by EU grant IST-2001-34124 (CAVIAR), and spanish grant TIC-2003-08164-C03-02 (SAMANTA).

## References

- [1] M. Sivilotti, *Wiring Considerations in analog VLSI Systems with Application to Field-Programmable Networks*, Ph.D. Thesis, California Institute of Technology, Pasadena CA, 1991.

- [2] Teresa Serrano-Gotarredona, Andreas G. Andreou, Bernabé Linares-Barranco. "AER Image Filtering Architecture for Vision-Processing Systems". IEEE Transactions on Circuits and Systems. Fundamental Theory and Applications, Vol. 46, NO. 9, September 1999.
- [3] A. Cohen, R. Douglas, C. Koch, T. Sejnowski, S. Shamma, T. Horiuchi, and G. Indiveri, *Report to the National Science Foundation: Workshop on Neuromorphic Engineering*, Telluride, Colorado, USA, June-July 2001. [[www.ini.unizh.ch/telluride](http://www.ini.unizh.ch/telluride)]
- [4] Kwabena A. Boahen. "Communicating Neuronal Ensembles between Neuromorphic Chips". Neuromorphic Systems. Kluwer Academic Publishers, Boston 1998.
- [5] A. Mortara, E.A. Vittoz and P. Venier. "A Communication Scheme for Analog VLSI Perceptive Systems", IEEE Journal of Solid-State Circuits. Vol. 30, No. 6, pp. 660-669, 1995.
- [6] Misha Mahowald. VLSI Analogs of Neuronal Visual Processing: A Synthesis of Form and Function. PhD. Thesis, California Institute of Technology Pasadena, California, 1992.
- [7] Pierre L'Ecuyer, François Panneton. "A New Class of Linear Feedback Shift Register Generators". Proceedings of the 2000 Winter Simulation Conference.
- [8] Linear Feedback Shift Register V2.0. Xilinx Inc. October 4, 2001. <http://www.xilinx.com/ipcenter>.
- [9] A. Linares-Barranco. Estudio y evaluación de interfaces para la conexión de sistemas neuromórficos mediante Address- Event-Representation. Ph.D. Thesis, University of Seville, Spain, 2003. <http://www.atc.us.es/alinares>
- [10] A. Linares-Barranco, R. Senhadji-Navarro, I. García-Vargas, F. Gómez-Rodríguez, G. Jimenez and A. Civit. Synthetic Generation of Address-Event for Real-Time Image Processing. ETFA 2003, Lisbon, September. Proceedings, Vol. 2, pp. 462-467.
- [11] Alejandro Linares-Barranco, Gabriel Jimenez-Moreno, Antón Civit-Ballcells, and Bernabé Linares-Barranco. On Synthetic AER Generation. ISCAS 2004. Vancouver, Canada, May, 2004.
- [12] P. Dayan and L. Abbot, *Theoretical Neuroscience* (MIT Press, Cambridge, MA, 2001)
- [13] F. Rieke, D. Worland, R. de Ruyter van Steveninck, W. Bialek. "Spikes: Exploring the Neural Code". The MIT Press, 1999.
- [14] J.R. Cogdell. "Modeling Random Systems". Pearson Prentice Hall, 2004.
- [15] Dante, V. and Del Giudice, P. and Whatley, A. M. "PCI-AER Hardware and Software for Interfacing to Address-Event Based Neuromorphic Systems". *The Neuromorphic Engineer*, 2:(1) 5-6, 2005, (In Press)

# Ultra Low-Power Neural Inspired Addition: When Serial Might Outperform Parallel Architectures

Valeriu Beiu<sup>1</sup>, Asbjørn Djupdal<sup>2</sup>, and Snorre Aunet<sup>3</sup>

<sup>1</sup> School of Electrical Engineering and Computer Science,  
Washington State University, USA  
vbeiu@eecs.wsu.edu

<sup>2</sup> Norwegian University of Science and Technology,  
Department of Computer Science and Information Technology, Norway  
djupdal@idi.ntnu.no

<sup>3</sup> University of Oslo, Department of Informatics, Norway  
sa@ifi.uio.no

**Abstract.** In this paper we analyse a serial (ripple carry) and a parallel (Kogge-Stone) adder when operating in subthreshold at 100nm and 70nm. These are targeted for ultra low power consumption applications. The elementary gates used are threshold logic gates (perceptrons). Simulations have been performed both with and without considering the delay on the wires. These simulations confirm that wires play a significant role, reducing the speed advantage of the parallel adder (over the serial one) from 4.5x to 2.2–2.4x. A promising result is that the speed of both adders improves more than 10x when migrating from 100nm to 70nm. The full adder based on threshold logic gates (used in the ripple carry adder) improves on previously known full adders, achieving 1.6fJ when operated at 200mV in 120nm CMOS. Finally, the speed of the parallel adder can be matched by the serial adder when operating at only 10–20% higher  $V_{dd}$ , while still requiring less power and energy.

## 1 Introduction

The scaling of CMOS in the few tens of nm range, brings with it many challenges [1]. A recent analysis of these challenges [2], has revealed (once more) that the most important ones are: power consumption (and the dissipation of the heat associated to it), and reliable functioning (i.e., both defect- and fault-tolerance). In this paper we shall focus on neural inspired ultra low power solutions. We will show that neural-inspired gates, combined with serial architectural approaches might outperform parallel solutions when run at the same speed in subthreshold scale CMOS.

## 1.1 Power Consumption

The power consumption of a CMOS integrated circuit contains three parts: dynamic power, short-circuit power, and static power. The dynamic average power component (capacitive switching) has seen a fair share of analysis and numerous mature methods to manage it. The short-circuit component has only rarely been considered, and it is quite common to ignore it, or estimate its contribution to about 10% of the dynamic power. The third component is the static power, and this has recently become important. Static power includes both leakage (which translates into stand-by power) and DC currents. The drastic increase of the leakage currents is raising concerns that, for future technology nodes, the standby-power will become larger than dynamic power, and is the culprit of scaling. Therefore, reducing leakage by design has started to be investigated.

## 1.2 Subthreshold Operation

One approach in the bid to lower power dissipation is the reduction of  $V_{dd}$ , obviously a very effective way of reducing all the components of power. The aggressive scaling of  $V_{dd}$  to below  $V_{Th}$  (known as subthreshold operation) has been known, and used in ultra low power designs [3, 4, 5]. The major disadvantage is the very slow speed, and subthreshold operation has been considered a poor approach, in that the much-needed speed is sacrificed for ultra low power (limiting its application range). Still, the papers reporting on subthreshold designs are mostly based on older technology nodes. The advantage of subthreshold operation is that it puts (at least part of) the leakage currents to good use.

## 1.3 Scaling and Optimal Design

The scaling of subthreshold designs was not properly investigated, and we expect that subthreshold could become an interesting design approach particularly because the operation speeds would be improved as scaling proceeds towards smaller technology nodes. It is not difficult to envision a situation in which designs in older technology nodes operating at standard power supply voltages would have comparable operation speeds to those in advanced technologies at subthreshold voltages. This would mean that for example a microprocessor designed to run at 1 GHz in 0.18  $\mu\text{m}$  at normal  $V_{dd}$  might be redesigned to operate at 1 GHz in subthreshold in say 45 nm. The main advantage would be a power reduction of a few orders of magnitude. If these were confirmed, the range of applications that could enjoy ultra-low power and improved speed would increase significantly. It must also be noted here that the available literature on subthreshold CMOS have ring oscillators as the main type of circuits studied, while only a few articles discuss gates, and small systems are the exception rather than the rule:

- a 16-bit serial adder [6];
- a 32-bit adder [7] (we infer that the adder is serial, as the authors mention that it has 2.3k transistors, using standard CMOS gates); and
- an 8x8 serial multiplier [8].

Only very recently, analysis of the optimality of subthreshold designs has started to be explored: (i) transistor level optimisations have been reported in [9], (ii) transistor sizing have only recently been discussed [10], while a very preliminary comparison of architectural approaches will be presented in this paper. It is obvious that subthreshold operation shows different design tradeoffs, and that optimal designs (at the transistor, gate, and system level) have yet to be identified.

## 1.4 Architectural Approaches

The particular example we are going to use in this paper is a 32-bit adder. Many different design alternatives are possible. It is commonly considered that the slowest one is the ripple carry adder (RCA, or serial solution), while the Kogge-Stone is expected to be the fastest. Classical CMOS gates are almost never used, when fast addition is in the picture, and domino gates are the norm.

Threshold logic gates (perceptrons) have also been advocated for fast addition, as they allow for shallow neural-inspired structures [11]. This theoretically leads to very high speeds [12], and a mixed dynamic and threshold logic is probably the fastest [13]. Threshold logic gates working in subthreshold have been reported in [14, 15], and this type of gate has been used for the circuits presented here. Finally, highly optimised threshold logic adders have also been used for achieving higher speeds in inherently slower single electron technologies [16].

Scaled power supply voltages lead to additional difficulties, e.g. noise margins. Process and environmental variations will also be major factors in designing systems that operate in subthreshold (due to the exponential dependence of the leakage current with respect to gate-to-source, threshold and drain-to-source voltages). Reducing the voltage supply might save the day for power consumption, but will adversely affect reliability.

The solution to this problem is to add redundancy to the design. From this perspective an RCA is easier to integrate with e.g. MAJ-3 multiplexing than a more complex parallel adder [17].

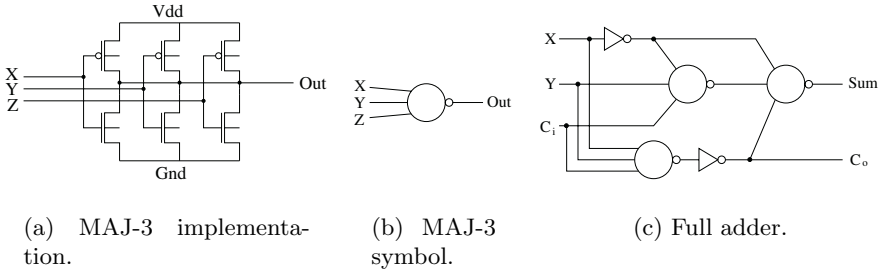
# 2 Serial and Parallel Adders

## 2.1 Details of the Circuits Used

Two 32-bit adder structures have been investigated. Both use the MAJ-3 gate of Fig. 1(a) as the basic logic gate.

The first adder is an RCA having a critical path of 32 stages. Each stage is a full-adder like the one in Fig. 1(c) [18]. This full adder has improved the PDP from 6.51fJ down to 1.57fJ and the power dissipation from 15.5nW to 7.3nW, at  $V_{dd} = 200mV$  in STM 120nm compared to all previously published full adders using the same basic logic gate. This RCA implementation has 704 transistors.

The second adder is a Kogge-Stone parallel prefix adder [19]. This adder can be seen in Fig. 2(a), while the different building blocks have been implemented using MAJ-3 gates (see Fig. 1(a)) and are detailed in Figs. 2(b), 2(c) and 2(d). This adder uses a total of 4180 transistors, and has a critical path of seven stages.



**Fig. 1.** MAJ-3 logic gate and full adder

Theoretically, the relative delay of RCA vs. Kogge-Stone should be  $\frac{32}{7} = 4.5$ .

## 2.2 Device Models and Experimental Setup

All simulations were done in Spice3f5 (ngspice). Berkeley Predictive Technology Models (BPTM) [20] have been used for all transistors. Each BPTM node was generated with the standard parameters suggested in [20].

Each input wire to an adder is driven by a gate, and each output has a gate as a load.

The addition performed in all experiments was  $0\text{xfffffff} + 1$ , which exercises the longest / critical path (from bit 0 to the carry out). Delays are measured from the time the input signal is asserted till the carry out reaches  $\frac{V_{dd}}{2}$ .

Energy was estimated from the current flowing to  $V_{ss}$  during the addition.

## 2.3 Evaluating Wires

For the simulations where wire delays were included, all wires have been modelled using the four-segment  $\pi$ -model. Parameters for the wires have been determined using the BPTM interconnect calculator with the values suggested for each node.

Length estimates of the longer wires were found by assuming that each building block of the Kogge-Stone adder in BPTM 70nm is  $7 \times 7 \mu\text{m}^2$ . The length of the shorter wires (which are the only ones present in the RCA adder) have been estimated based on a layout of the MAJ-3 gate. For other technology nodes, these lengths have been scaled accordingly.

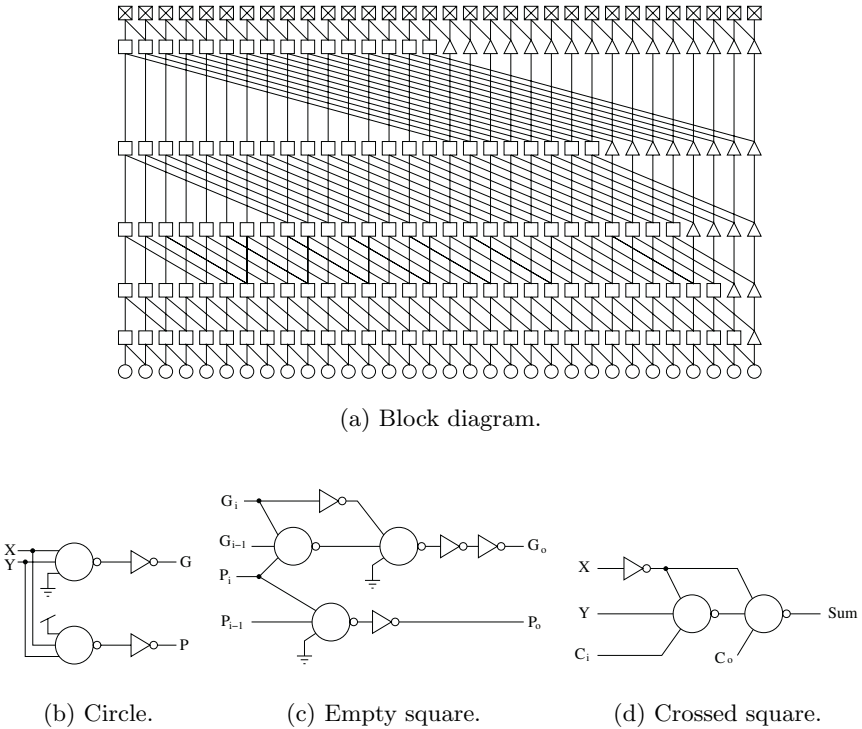
## 3 Results and Discussions

The results of the simulations are synthesised in Table 1. The most important ones are also presented in Figs. 3 and 4.

As expected, the parallel adder is faster than RCA when the same  $V_{dd}$ , technology node, and transistor sizes are used.

Simulations without considering wire delays are in very good agreement with the theoretical  $\frac{32}{7} = 4.5$  speed advantage of the parallel adder.





**Fig. 2.** Kogge-Stone adder

Simulations including wire delays show that these have a large impact on adder performance. In BPTM 100nm, the Kogge-Stone adder is 4.5 times faster than RCA, while with wire delays it is only 2.2–2.4 times faster. Similar tendencies hold true for 70nm (see Table 1).

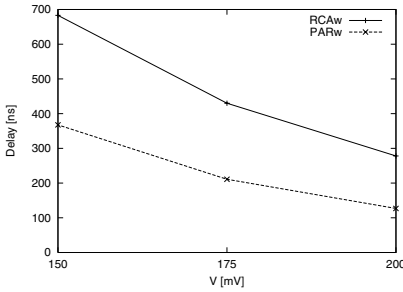
Figs. 3(a) and 3(b) reveal the correlation between  $V_{dd}$  and propagation delays. As  $V_{dd}$  is increased, power consumption is also increased. Hence, Fig. 4 shows the estimated energy for both the parallel adder and RCA.  $V_{dd}$  for the parallel adder is varied from 150mV to 250mV giving propagation delays spanning one order of magnitude: 5000ns to 500ns.  $V_{dd}$  for RCA was set such as to achieve the same performance. All our experiments show that this  $V_{dd}$  is between 110% and 120% of the  $V_{dd}$  of the parallel adder. For all these experiments, *the RCA was more energy efficient than the parallel adder when running at the same speed.*

When using the same  $V_{dd}$  in both 100nm and 70nm, we have also seen more than 10x reduction in delays for both adders. There is no clear indication that delays in subthreshold due to local wiring is scaled any differently from transistors across technology nodes.

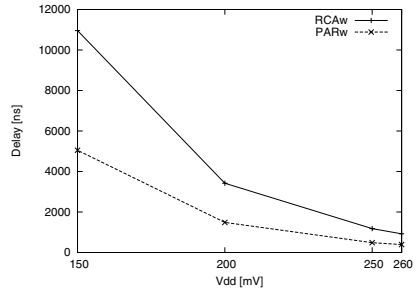
All the results reported here are from simulations and not from chip measurements. This must be taken into account when interpreting the data, as errors

**Table 1.** Summary of simulation results. RCA = Ripple Carry Adder, PAR = Kogge-Stone Adder, “w” indicates that wire delays are included in the simulations

| Circuit | $V_{dd}$<br>[mV] | Node<br>[nm] | $t_{delay}$<br>[ns] | $E$<br>[pJ] | $L_{PMOS}$<br>[nm] | $W_{PMOS}$<br>[nm] | $L_{NMOS}$<br>[nm] | $W_{NMOS}$<br>[nm] |
|---------|------------------|--------------|---------------------|-------------|--------------------|--------------------|--------------------|--------------------|
| RCA     | 260              | 100          | 767.9               |             | 120                | 690                | 120                | 120                |
| PAR     | 260              | 100          | 168.8               |             | 120                | 690                | 120                | 120                |
| RCAw    | 260              | 100          | 922.6               |             | 120                | 690                | 120                | 120                |
| PARw    | 260              | 100          | 391.0               |             | 120                | 690                | 120                | 120                |
| RCA     | 250              | 100          | 1179                |             | 120                | 690                | 120                | 120                |
| PARw    | 250              | 100          | 484.8               |             | 120                | 690                | 120                | 120                |
| RCA     | 200              | 100          | 2847                |             | 120                | 690                | 120                | 120                |
| PAR     | 200              | 100          | 627.6               |             | 120                | 690                | 120                | 120                |
| RCAw    | 200              | 100          | 3421.7              |             | 120                | 690                | 120                | 120                |
| PARw    | 200              | 100          | 1484.3              |             | 120                | 690                | 120                | 120                |
| RCAw    | 150              | 100          | 10961               |             | 120                | 690                | 120                | 120                |
| PARw    | 150              | 100          | 5051                |             | 120                | 690                | 120                | 120                |
| RCA     | 200              | 70           | 291                 |             | 90                 | 590                | 90                 | 90                 |
| PAR     | 200              | 70           | 77                  |             | 90                 | 590                | 90                 | 90                 |
| RCA     | 200              | 70           | 275.7               |             | 100                | 600                | 100                | 100                |
| PAR     | 200              | 70           | 85                  |             | 100                | 600                | 100                | 100                |
| RCA     | 200              | 70           | 241.3               |             | 80                 | 460                | 80                 | 80                 |
| PAR     | 200              | 70           | 64.8                |             | 80                 | 460                | 80                 | 80                 |
| RCAw    | 200              | 70           | 278                 |             | 80                 | 460                | 80                 | 80                 |
| PARw    | 200              | 70           | 127                 |             | 80                 | 460                | 80                 | 80                 |
| RCAw    | 175              | 70           | 430                 |             | 80                 | 460                | 80                 | 80                 |
| PARw    | 175              | 70           | 211                 |             | 80                 | 460                | 80                 | 80                 |
| RCA     | 150              | 70           | 595.5               |             | 80                 | 460                | 80                 | 80                 |
| PAR     | 150              | 70           | 186.6               |             | 80                 | 460                | 80                 | 80                 |
| RCAw    | 150              | 70           | 683.0               |             | 80                 | 460                | 80                 | 80                 |
| PARw    | 150              | 70           | 367.6               |             | 80                 | 460                | 80                 | 80                 |
| RCAw    | 183.8            | 100          | 5045.5              | 1.48        | 120                | 690                | 120                | 120                |
| PARw    | 150              | 100          | 5051                | 2.26        | 120                | 690                | 120                | 120                |
| RCAw    | 238.5            | 100          | 1479.9              | 2.48        | 120                | 690                | 120                | 120                |
| PARw    | 200              | 100          | 1484.3              | 3.4         | 120                | 690                | 120                | 120                |
| RCAw    | 292.5            | 100          | 488.2               | 5.0         | 120                | 690                | 120                | 120                |
| PARw    | 250              | 100          | 484.8               | 6.57        | 120                | 690                | 120                | 120                |



(a) BPTM 70nm.



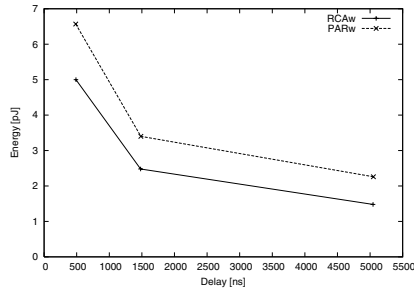
(b) BPTM 100nm.

**Fig. 3.** Impact of  $V_{dd}$  on delay

might come from the simulator, the Berkeley models, or from incorrect assumptions regarding transistor sizes and wire lengths.

The results have also shown to depend heavily on transistor sizing. Because of the difficulty of ensuring representative scaled sizing across technology nodes, comparisons across different technology nodes should be taken with care.

For FPGAs, wire delays are larger than for a custom design due to longer wires as well as switchbox resistance. Therefore a serial architecture may have



**Fig. 4.** Energy used by 32 bit adders (BPTM 70nm).  $V_{dd}$  for Kogge-Stone adder was 150mV, 200mV and 250mV.  $V_{dd}$  for RCA was 183mV, 238mV, 292mV for giving the same delay as the Kogge-Stone adder

an even larger advantage when compared to parallel in subthreshold FPGAs, and such techniques are already used in today's FPGAs.

## 4 Conclusions and Future Work

This paper has presented simulations in subthreshold that back the followings:

- migrating from 100nm to 70nm reduces the propagation delay by over 10x;
- full adders using threshold logic gates require very small energy;
- serial adders can match the speed of parallel ones in subthreshold when  $V_{dd}$  is increased only by 10-20%;
- at equal speeds in subthreshold, a serial adder has an advantage over parallel ones with respect to power consumption (and energy).

The more general conclusion is that, when operated in subthreshold serial solutions might do better than parallel ones, both for full custom arithmetics and FPGA based arithmetics.

Future work should concentrate on: optimising the implementations of the gates (e.g., combinations of gates), better characterisation (through fabrication and measurements), and the analysis of reliability enhanced adders using both low level techniques (e.g., high matching), as well as circuit level ones (based on redundancy).

## References

1. ITRS: The International Technology Roadmap for Semiconductors. <http://public.itrs.net/> (2004)
2. Beiu, V., Rückert, U., Roy, S., Nyathi, J.: On nanoelectronic architectural challenges and solutions. In: Proc. IEEE Conference on Nanotechnology. (2004) 628–631

3. Burr, J.B., Shott, J.: A 200 mV self-testing encoder/decoder using stanford ultra-low-power CMOS. In: Proc. IEEE International Solid-State Circuits Conference. (1994) 84–85
4. Lande, T.S., Wisland, D.T., Sæther, T., Berg, Y.: FLOGIC-floating-gate logic for low-power operation. In: Proc. International Conference on Electronics, Circuits, and Systems. Volume 2. (1996) 1041–1044
5. Kim, C.H., Soeleman, H., Roy, K.: Ultra-low-power DLMS adaptive filter for hearing aid applications. *IEEE Trans. on VLSI Systems* **11** (2003) 1058–1067
6. Wentzloff, D.D., Calhoun, B.H., Min, R., Wang, A., Ickes, N., Chandrakasan, A.P.: Design considerations for next generation wireless power-aware microsensor nodes. In: Proc. International Conference on VLSI Design. (2004) 361–367
7. Ishibashi, K., Yamashita, T., Y, A., Minematsu, I., T.Fuji-moto: A 9uW 50MHz 32b adder using a self-adjusted forward body bias in SoCs. In: Proc. International Solid-State Circuits Conference. Volume 1. (2003) 116–119
8. Soeleman, H., Roy, K., Paul, B.: Robust subthreshold logic for ultra-low power operation. *IEEE Transactions on VLSI Systems* **9** (2001) 90–99
9. Paul, B.C., Raychowdhury, A., Roy, K.: Device optimization for digital subthreshold logic operation. *IEEE Transactions on Electron Devices* **52** (2005) 237–247
10. Calhoun, B.H., Wang, A., Chandrakasan, A.: Device sizing for minimum energy operation in subthreshold circuits. In: Proc. IEEE Custom Integrated Circuits Conference. (2004) 95–98
11. Beiu, V., Quintana, J., Avedillo, M.: VLSI impl. of threshold logic: A comprehensive survey. *IEEE Transactions on Neural Networks* **14** (2003) 1217–1243
12. Beiu, V.: Constructive threshold logic addition: A synopsis of the last decade. In: Proc. International Conference on Neural Networks. Number 2714 in LNCS (2003) 745–752
13. Celinski, P., Al-Sarawi, S., Abbott, D., Cotofana, S., Vassiliadis, S.: Logical effort based design exploration of 64-bit adders using a mixed dynamic-cmos/threshold-logic approach. In: Proc. Annual Symposium on VLSI. (2004) 127–132
14. Aunet, S., Berg, Y., Tjore, O., Næss, ., Sæther, T.: Four-MOSFET floating-gate UV-programmable elements for multifunction binary logic. In: Proc. Multiconference on Systemics. Volume 3. (2001) 141–144
15. Aunet, S., Oelmann, B., Abdalla, S., Berg, Y.: Reconfigurable subthreshold CMOS perceptron. In: Proc. IEEE International Joint Conference on Neural Networks. (2004) 1983–1988
16. Sulieman, M., Beiu, V.: Characterization of a 16-bit threshold logic single electron technology adder. In: Proc. International Symposium on Circuits and Systems. (2004) 681–684
17. Beiu, V.: A novel highly reliable low-power nano architecture: When von Neumann augments Kolmogorov. In: Proc. Application-specific Systems, Architectures and Processors. (2004) 167–178
18. Iwamura, H., Akazawa, M., Amemiya, Y.: Single-electron majority logic circuits. *IEICE Transactions on Electronics* **E18-C** (1998) 42–48
19. Kogge, P., Stone, H.: A parallel algorithm for the efficient solution of a general class of recurrence equations. *IEEE Transactions on Computers* (1973) 786–793
20. Cao, Y., Sato, T., Orshansky, M., Sylvester, D., Hu, C.: New paradigm of predictive MOSFET and interconnect modeling for early circuit design. In: Proc. IEEE Custom Integrated Circuits Conference. (2000) 201–204 <http://www-device.eecs.berkeley.edu/~ptn>.

# An Asynchronous 4-to-4 AER Mapper

H. Kolle Riis and Ph. Häfliger\*

Department of Informatics, University of Oslo, Norway

**Abstract.** In this paper, a fully functional prototype of an asynchronous 4-to-4 Address Event Representation (AER) mapper is presented. AER is an event driven communication protocol originally used in VLSI implementations of neural networks to transfer action potentials between neurons. Often, this protocol is used for direct inter-chip communication between neuromorphic chips containing assemblies of neurons. Without an active device between two such chips, the network connections between them are hard-wired in the chip design. More flexibility can be achieved by communicating through an AER mapper: The network can freely be configured and, furthermore, several AER busses can be merged and split to form a complex network structure. We present here an asynchronous AER mapper which offers an easy and versatile solution. The AER mapper receives input from four different AER busses and redirects the input AE to four output AER busses. The control circuitry is implemented on an FPGA and is fully asynchronous, and pipelining is used to maximize throughput. The mapping is performed according to preprogrammed lookup tables, which is stored on external RAM. The mapper can emulate a network of up to  $2^{19}$  direct connections and test results show that the mapper can handle as much as  $30 \times 10^6$  events/second.

## 1 Introduction

The AER protocol [7] is a popular tool in neuromorphic applications. It is used to emulate direct connections in for example neural networks. It uses a high speed digital bus which events are asynchronously multiplexed onto. A unique address identifies each sender, e.g. a neuron, and the receiver is responsible for distributing this event to the correct location. Since the speed of the bus exceeds the frequency of events, very few collision occur and can be handled with a minimal delay. However, the virtual connection must be designed in hardware in advance and can not be changed during operation. So to use the AER protocol with multiple PCB's, one needs to carefully plan the interconnections. In many applications, this is difficult, in some cases impossible. One example is evolutionary hardware, where a genetic algorithm is used to determine the configuration directly in hardware. Thus, the ability to change the connectivity of several components is crucial to test a system in real time and the need to be able to program the interconnections "on the fly" is apparent. Two research

---

\* Supported by the EU 5<sup>th</sup> Framework Programme IST project CAVIAR.

groups in Rome and Sevilla have developed devices which address this issue. The Rome-board [1] is a PCI-AER board and is mainly constructed to be an interface between a PC and boards that communicate with the AER protocol. It can work as an AER mapper, it can monitor the communication on an AER bus or it can send sequences of events to an AER bus to emulate a real stimuli. Though it has many nice features, it is fairly slow ( $5 \times 10^6$  events/sec) and it needs to be connected to a PC to operate. Like the Sevilla-board, which is a simpler, dedicated and faster AER mapper, the design is synchronous. (Both boards are under development and most of the information is based on private communication since there exists no publications to refer to.) And since the time domain is of importance in the AER protocol, e.g. information lies in the timing between successive events [6], to approach the problem from an asynchronous point of view may seem preferable. This is due to the fact that synchronous devices quantize information in the time domain, thus vital timing information can be lost. Therefore, we introduce the asynchronous 4-to-4 AER mapper, which can easily be programmed to emulate any network of up to eight individual components.

## 2 Model

The AER mapper is a four-to-four mapper. It receives input from four different AER input busses and redirects these addresses to four output AER busses. In this way, the device is capable of interconnecting up to eight individual chips or circuit boards, thus emulating a huge network of connections. The total amount of direct connections which can be emulated is  $2^{19}$ , over 0.5 million connections. The mapping is performed according to preprogrammed lookup tables, which we

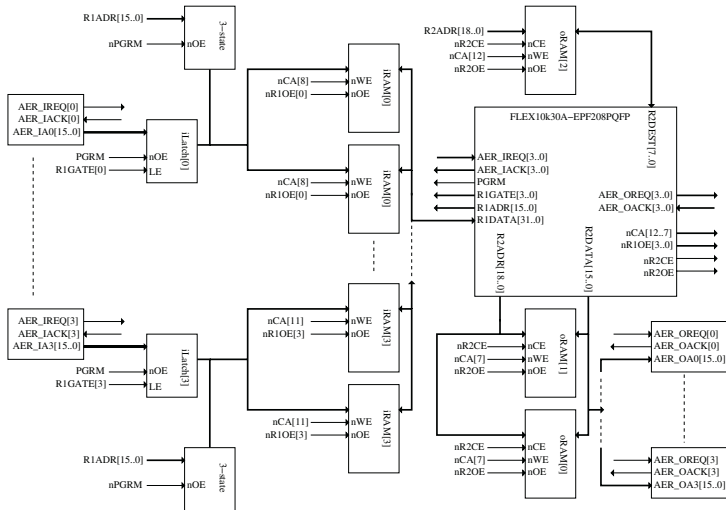


Fig. 1. The AER mapper

store on external RAM. This mapping can be changed during normal operation by a separate AER cable. The process of programming is covered in section 4.

Each input bus takes as input a 16 bit address and sends out an arbitrarily amount of output AE's of the same length. The output AE's can be sent on any of the four output busses, the output bus does not need to be the same for each AE and an output AE can be sent on several output busses at the same time. Since there are four input busses with an address space of  $2^{16}$  and a total of  $2^{19}$  possible direct connections, one input event can on average cause two output events if the mapper operates at the limit of its input capacity. However, individual inputs can cause up to  $2^{13}$  outputs, and inputs that cause the same set of outputs as previously programmed input, need not consume extra mapping capacity. This is possible, because the mapper is constructed with two blocks of RAM. The first block of RAM (*i*RAM), has a 16 bit address space (64k) of 32 bit words, where 13 bits are used to denote the number of output AE's and 19 bits are used as a pointer to the second block of RAM (*o*RAM). The second block of RAM has a 19 bit address space (512k) of 24 bit words, where 16 bits are used as the output address and 4 bits for selecting output bus (4 bits not used).

The mapping is performed in two main stages. A schematic of the AER mapper and the control logic on the FPGA are shown in figure 1 and 2 respectively. First, when an event is received on one of the four input AER busses, e.g. bus number 3, a request is sent to the FPGA. The request is processed and triggers LE[3] such that the incoming address is stored on an external latch (*i*Latch[3]). The request is then acknowledged. The input AER bus is again ready to receive an event. At the same time a request is sent to the next stage on the FPGA which grants access to *i*RAM. A full non-greedy arbitration is performed for all four inputs such that collisions are avoided. When the event is granted access, **nr10E**[3] goes active and the data is loaded from *i*RAM[3] and sent to the FPGA, where it is latched and acknowledged. The first stage is now complete and a new address can be stored on *i*Latch[3].

The second stage uses the data from *i*RAM to determine which addresses to send to *o*RAM. A simple example illustrates the process. If the 19 bit pointer to *o*RAM is 1000 (DEC) and the 13 bit number 10 (DEC), the mapper will send 10 successive addresses, i.e. 1009, 1008, .. , 1001, 1000. The first address (1009) is sent to a new internal latch (*m*Latch) along with a request, where it is stored and acknowledged. The next address (1008) can then be calculated. The first address is then granted access to *o*RAM by **nr20E** and the data is loaded. The data is 20 bits wide and contains the 16 bit output AER address and a 4 bit number. The number is sent back to the FPGA and stored on a latch (*o*Latch) and acknowledged. Thus, the second address can be latched by *m*Latch. The latched number determines which of the four output AER busses the output AER address is to be sent to. Thus, if the number is 0101, the AER output address is sent on AER bus 1 and 3. When this output AER address is acknowledged, the second address is granted access to *o*RAM and new data is loaded. This process continues until all addresses are processed.

### 3 FPGA Implementation

The control logic of the AER mapper is implemented on an FPGA. We have used an Altera Flex10K30A FPGA which has 208 I/O pins. It operates on a 3.3V supply. The FPGA is programmed using a MasterBlaster serial communication cable, which allows us to program the device from a PC or UNIX workstation. We have also chosen to include a second configuration device (EPC1PDIP3), which is a ROM where the final version of the FPGA design can be programmed and loaded at startup. An alternative solution in future implementations could be to use a flash card with a USB interface to a PC or UNIX workstation.

The control logic of the mapper is a fully asynchronous design and is based on the work by Häfliger in [3]. Since asynchronous design is not a very common and preferable design method in FPGA implementations, and commercial FPGA's are solely constructed for synchronous design [5], there exist no supported timing or delay elements which can be used in commercial FPGA design. Both specialized FPGA's [4] and different methods have been developed for asynchronous implementations [2], but these remain expensive and cumbersome to use, and the extra effort and money would probably not justify the choice of asynchronous over synchronous. And since timing is a crucial part of asynchronous design, we therefore had to find a method to set a more or less fixed delay on the control signals on a common cheap off-the-shelf FPGA. The solution to the problem was to use a LCELL primitive supported by the Altera Quartus software package. The primitive is basically a buffer, with a simulated delay of approximately 2ns for one LCELL. Test results also showed that the delay was as expected with only minimal variations. Thus, we were able to create fairly accurate delay ele-

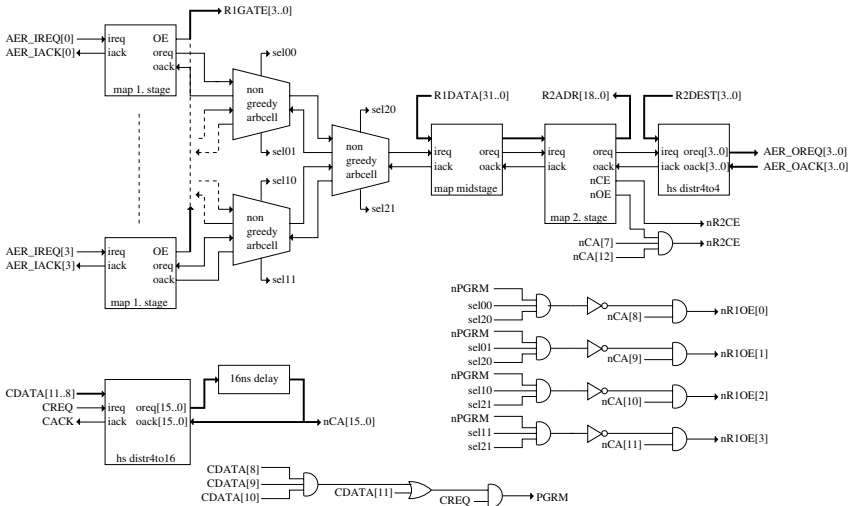


Fig. 2. FPGA schematic

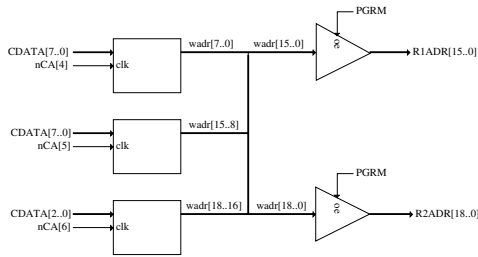


ments throughout the design. Though not a failsafe way of designing, it proved to be a powerful tool in easily creating the control logic of the asynchronous AER mapper.

## 4 Programming

A separate AER cable is used to program the RAM. It takes as input a 12 bit address ( $CDATA[11..0]$ ), where 8 bits are data ( $CDATA[7..0]$ ) and the remaining 4 bits ( $CDATA[11..8]$ ) are used to determine what the data is to represent.  $CDATA[11..8]$  is demultiplexed by  $HSdist4to16$ , and the resulting signals ( $CA[12..0]$ ) are used as clock input to internal latches ( $CA[6..0]$ ) and to control the write enable of the external RAM ( $nCA[12..7]$ ).

The programming is not done directly, but the data is first stored on several internal latches. This is done such that the whole address and data for one entry can be stored before the RAM itself is programmed. For example, to program a set of data at one specific address in *i*RAM, one need to latch 48 bits internally, i.e. 16 bit address and 32 bit data, before the data can be written to the RAM. Which latch the data is stored on, determines what the data is used for. We have four internal latches for the RAM data and three more for the RAM addresses. Several tri-state buffers are used to separate the different physical lines, such that the same data can be used for both *i*RAM and *o*RAM. The programming latches for the address part are shown in figure 3.



**Fig. 3.** The internal programming latches. Two tri-state busses separates the two different physical lines  $R1ADR[15..0]$  and  $R2ADR[18..0]$

The programming is executed in MATLAB and some scripts are written to facilitate the operation. The main function is

```
mapping=pgrmMapping(in,out,inbus,outbus);
```

where *in* is the input address, *out* is an array with the output addresses, *inbus* the input bus number and *outbus* an array with the the output bus numbers. It returns a matrix *mapping* with all the mappings executed so far.

The main function consists of three sub functions

```
[r2adr,mapping]=getR2adr(in,out,inbus,outbus);
[r,c]=size(out);
pgrmRAM1(in,r2adr+r*2^19,inbus);
for i=1:r
 pgrmRAM2(r2adr+(i-1),out(i,1),outbus(i,1));
end;
```

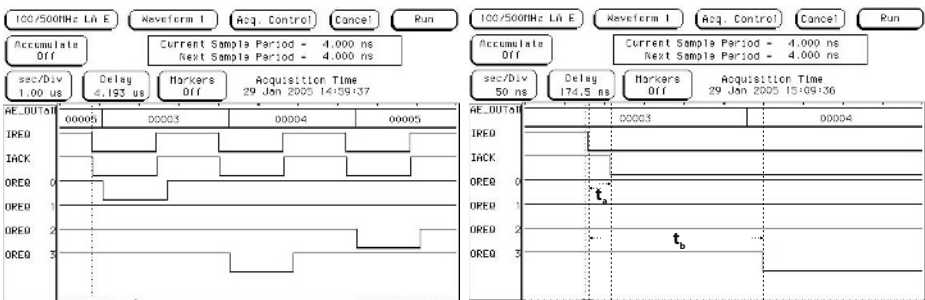
First, the main function checks if the inbus and outbus are correctly specified, i.e. if they are a number from 0 to 3. Then, the first sub function, `getR2adr`, is called. This function loads a mapping database, `mapping`, where all previous mappings are stored. The database is constructed such that the column of one entry denotes which *o*RAM address the output AER address is stored. Each entry holds the information given to the main functions. Therefore, the mapping can easily be retrieved, it prevents the user from overwriting entries in the *o*RAM and, furthermore, the user do not need to be concerned with both the *i*RAM data and the *o*RAM address when programming a mapping.

Based on the information retrieved from the first sub function, the two next functions are called. `pgrmRAM1` takes as input the input address and the input bus number directly from the main function. The data to be stored ( $r2adr+r*2^{19}$ ) is a pointer to *o*RAM (first 19 bits) plus a number *r* (last 13 bits) which denotes how many output AER addresses which are to be generated.

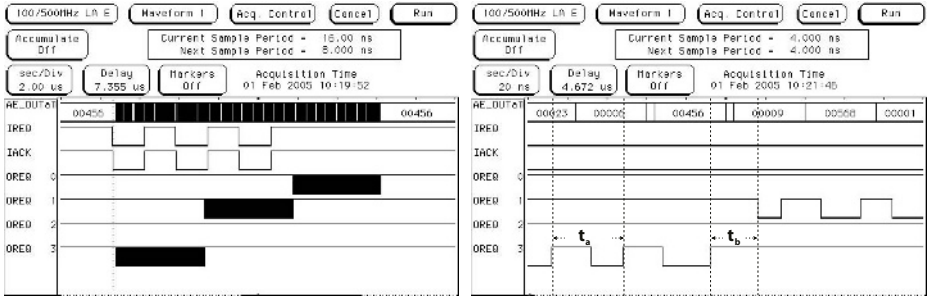
`pgrmRAM2` is then called for each of the output AER addresses. It takes as input the computed *o*RAM address, which is increased by one for each call, the output AER address and its corresponding output bus number. The programming is now complete for one input AE.

## 5 Results

The circuit was tested with a National Instrument (NI) DAQ board (PCI-6534 High Speed Digital I/O PCI module) connected to a PC workstation. In addition, a PCI-to-AER and a 5V-to-3.3V PCB was used to connect the PCI-bus to the



**Fig. 4.** Measurement 1: The timing of input request/acknowledge and the output request is highlighted to the right.  $t_a$  is 25ns while  $t_b$  is 225ns



**Fig. 5.** Measurement 2: The transition from different input AE is highlighted to the right.  $t_a$  is 45ns and  $t_b$  is 20ns

mapper board. All signals from the PCI-bus have a length of  $1.6\mu s$ , independent of when the acknowledge is received. We used a Hewlett Packard 16500C Logic Analysis System to sample and plot the test data. The sampling period of the 16500C is 4ns, which is sufficient for our test measurements. For more accurate timing measurements, e.g. measuring the delay of the LCELL primitive, we used a Agilent 54624A digital oscilloscope.

In figure 4, a simple mapping experiment is plotted where IREQ is the input request from the PCI-board, IACK the corresponding request, OREQ[3..0] the output requests of the four output busses and AE\_OUT the output address. Three successive AE's are received on input bus 0 and redirected to output bus 0, 3 and 2 respectively. The timing of the initial handshake is highlighted to the right. From the figure, one can see that it takes about 25ns ( $t_a$ ) from the input request is acknowledged and approximately 225ns ( $t_b$ ) before the output request is sent. Since the input request from the PCI-board has a fixed period of  $3.2\mu s$ , we are not able to test the speed of such a one-to-one mappings directly. Thus, the effect of pipelining is not taken advantage of and the full potential of the mapper is not shown.

However, if we perform a one-to-multiple mapping experiment, i.e. if one input AE results in the sending of multiple output AE's, several features of the AER mapper can be utilized. Such an experiment is plotted in figure 5. We plot the same signals as in the previous experiment and three input AE's are programmed to send out 120 AE's each on different output busses. The temporal resolution of the Logic Analyzer is not high enough to plot the individual spikes, so the output AE's are shown as black solid lines. The whole process takes roughly  $13\mu s$  which means that the mapper can send just under  $30 \times 10^6$  events/second.

The transition between two of the incoming AE's is again plotted to the right. We see that the period  $t_a$  of the 360 output AE's is approximately 45ns and that the delay  $t_b$  between AE's from the two different inputs is not more than 20 ns. This suggests that one-to-one mappings also can be processed at a similar rate. It must be noted that the global delay of the signals still is about 225ns, but

since this is a fixed delay forced on any of the incoming events, the timing of events are preserved throughout the system.

From the same figure, we see that the output address is valid about 5-10ns before and after the output request is set. This is a safety margin which is set by using the previously described delay elements. This margin may be reduced to improve the performance of the system. If only reduced by a total of 4-5 ns, thus reducing the AE period to 40ns, the overall performance is improved by nearly 10% .

## 6 Conclusion

An asynchronous AER mapper has been presented. It can be used as a passive or active device in a multi-node network that uses the AER protocol for inter-chip communication. Its ability to emulate complex networks structures combined with speed and robustness makes it a powerful tool for interconnecting relatively large systems.

The asynchronous FPGA implementation is at the present moment not optimized for speed, but instead we have concentrated on making the control circuitry fail proof without any glitches. This is very important in asynchronous design, because any ill timed signal may set the system in a non valid state. However, several improvements are in progress which can significantly speed up the communication without compromising on robustness. Also, some improvements in handling non valid addresses are currently tested, i.e. addresses which are not programmed to be redirected. The final result of the mapper should be an easy and “plug-and-play”-like device, such that anyone who is interested in using the mapper, needs only minimal knowledge of the circuit design and would only need a higher level software script to program and prepare the mapper for operation.

## References

1. V. Dante and P. Del Giudice. *The pci-aer interface board*. 2001 Telluride Workshop on Neuromorphic Engineering Report, <http://www.ini.unizh.ch/telluride/previous/report01.pdf>, pp. 99-103.
2. S.Y. Tan, S.B. Furber and Wen-Fang Yen. *The design of an asynchronous VHDL synthesizer*. Proceedings - Design, Automation and Test in Europe, pp. 44-51, 1998.
3. P. Häfziger. *Asynchronous Event Redirecting in Bio-Inspired Communication*. International Conference on Electronics, Circuits and Systems, vol. 1:pp. 87-90, 2001.
4. S. Hauck and S. Burns, G. Borriello and C. Ebelingw. *An FPGA for implementing asynchronous circuits*. IEEE Design & Test of Computers, vol.11:pp. 60, 1994.
5. R. Payne. *Asynchronous FPGA architectures*. IEE Proceedings - Computers and Digital Techniques, vol. 143:pp. 282-286, 1996.
6. Wulfram Gertsner. *Basic Concepts and Models: Spiking Neurons*, Chapter 1. Bradford Books, 1998.
7. M. Mahowald. *The Silicon Optic Nerve from An Analog VLSI System for Stereoscopic Vision*, Chapter 3. Kluwer Academic Publishers, 1994.

# Fast Optoelectronic Neural Network for Vision Applications

Marta Ruiz-Llata and Horacio Lamela

Universidad Carlos III de Madrid,  
28911 Leganés, Madrid, Spain  
<http://gdo.uc3m.es/got1/>

**Abstract.** This paper reports the recent steps to the attainment of a compact high-speed optoelectronic neuroprocessor based on an optical broadcast architecture that is used as the processing core of a vision system. The optical broadcast architecture is composed of a set of electronic processing elements that work in parallel and whose input is introduced by means of an optical sequential broadcast interconnection. Because of the special characteristics of the architecture, that exploits electronics for computing and optics for communicating, it is readily scalable in number of neurons and speed, thus improving the performance of the vision system. This paper focuses on the improvement of the optoelectronic system and electronic neuron design to increase operation speed with respect to previous designs.

## 1 Introduction

Neural networks are computational approach that is nowadays demonstrating applicability to many computational cost problems [1]. On the contrary of traditional processors, an artificial neural network can be seen as a set of very simple processing elements with a high degree of interconnectivity between them. The computation strength of a simple processing element is very small, but the massive cooperation between lots of them results in a powerful machine for tasks as pattern recognition and classification [2].

For vision there have been a wide range of neural systems with application ranging from early to high level vision operations. Examples are the hardware implementation of the bio inspired artificial retina [3] or – more recent – pulse or spiking neural networks (PCNN) [4] for low or medium level tasks. For high level operation such image classification, neural networks are emulated via software [1]. Software based implementations of neural networks are often not fast enough to meet the real time requirements of some vision application but the main disadvantage is that their development require high computer literacy and math background; which contrast with one of the most exciting properties of neural networks that is its ability of learning.

There have been also hardware implementations of neural network processors. The idea was the development of parallel processing architectures to optimize neural network algorithms; examples were the CNAPS processor [5] and the analogue Intel ETANN [6] among others [7]. Recent trends are the implementation of neural proc-

essing systems instead of parallel processing architectures. As example we can find the VindAx processor from Axeon Ltd. [8] implementing a self-organizing map and the IBM zero instruction set computer (ZISC) implementing a radial basis function (RBF) neural network [9]. The ZISC has been tested and compared with other neural network hardware implementation for a quite complicated vision task: face tracking and identification [10]. Although higher operation speed, the performance of this system is worse than the other proposed implementations (RBF on FPGA and DSP); the performance difference was linked to the limit in the size of input pattern to 64 elements.

Our work is focused on the implementation of a neural processing system for smart vision applications. The performance demanding for such a device are fast operation speed and large size. Large size means that the hardware architecture must not impose a (small) limit in the size of input patterns nor to the number of neural nodes and interconnections between them.

It is also our understanding that optics may help in the implementation of neural processing architectures, but with a different approach that is used in optical computing architectures – such optical correlators [11] and optical neural networks [12]. In those systems optics is used for computing, basically for spatial filtering, in optical correlators, and for multiplication and addition, in optical vector-matrix multipliers. Some problems encountered in the realization of large-scale optoelectronic systems are: optical alignment, interconnection weight reconfiguration and assignment by spatial light modulators (SLM), and the construction of the associated opto-mechanical sub-systems with reduced dimensions. Our point of view is that optics must be used only for interconnects, while electronics must be used for computing [13].

In this paper we describe in section 2 the basic principles of the optoelectronic architecture for neural networks we have proposed [14] and that has been used as the processing core of a vision system [15][16]. In section 3 we describe the implementation of a new fast prototype based on a new neuron design. In section 4 we describe the vision system that will be benefited by the new fast optoelectronic neural network design and its preliminary results. We finish with conclusions in section 5.

## 2 Broadcast Neural Network Processor

As represented in equation (1), the output of a single neuron  $y_j$  in a neural network is a non-linear function of a weighted sum of inputs:

$$y_j = f \left( \sum_{i=1}^N w_{ji} x_i \right) \quad (1)$$

where  $N$  is the size of the input pattern,  $x_i$  is one element of the input and  $w_{ji}$  is the interconnection weight between input  $i$  and neuron  $j$ . Hardware implementations of neural processors, especially optoelectronic implementations [11][12], usually try to construct an architecture where basic operations are executed simultaneously for all neurons. The hardware architecture we propose simplifies the optical interconnection scheme and avoids the use of SLM but takes advantage of the high spatial and temporal communication bandwidth of optical beams. As represented in figure 1, it is composed of a set of  $K$  cells comprising of  $M$  optoelectronic neurons that share the same

optical input and that execute neural operations – i.e. multiplications and additions – in parallel. We have shown this architecture is configurable [16][17] and can be package in a small volume by the use of CMOS circuits for the optoelectronic neurons and holographic interconnections elements [17].

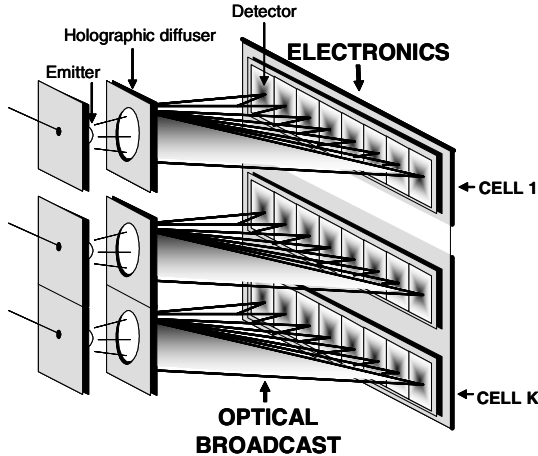


Fig. 1. Optoelectronic architecture with K cells with M neurons in each cell

### 2.1 Neuron Functionality

The option we propose for our optoelectronic architecture is time-sequential processing of the inputs [14]. We propose a system composed of a large number of processing elements whose functionality can be seen in figure 2.

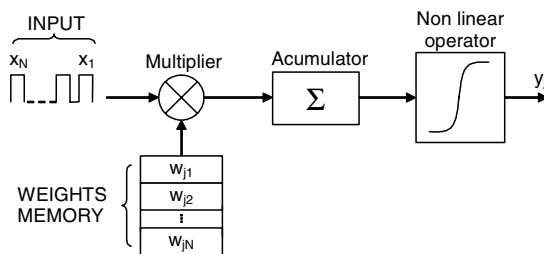


Fig. 2. Neuron model in the optical broadcast architecture

Each processing element comprises of one multiplier and one accumulator whose input is presented sequentially, one input element at a time. To obtain the output  $y_j$  as a function of the sum of N weighted inputs, N time slots are required. In each time slot one input element is introduced, multiplied by the corresponding interconnection weight and accumulated to previous result. Time-sequential input processing seems to

delay neuron output, but it benefits the overall speed performance of our optoelectronic architecture as it is discussed in reference [17].

## 2.2 Optical Interconnection for the Broadcast Cell

Optics has been proposed as a solution to overcome fundamental problems for scaling electrical interconnections [13]. A neural network cell in our architecture is composed of an array of  $M$  neurons. All the neurons in a cell share the same time-distributed input. We propose to use an optical-broadcast interconnection to distribute the inputs to all the neurons in a cell, as represented in figure 3. This optical interconnection will allow the distribution of the input signal, at high speed, to a very high number of processing elements that work in parallel. As the input is presented sequentially and distributed to all the neurons in a cell, we obtain a fully-interconnected architecture, that is, any input can contribute to the activation of any neuron.

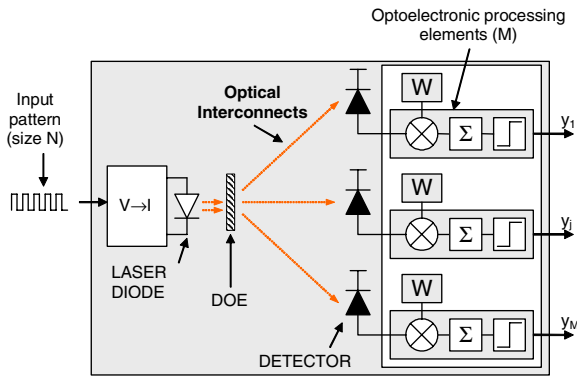


Fig. 3. Optical broadcast cell

## 3 Fast Prototype Implementation

The input vector is distributed sequentially to all neurons in the cell by means of an optical time modulated signal. The optoelectronic neuron must multiply each input with the corresponding interconnection weight and accumulate the result until the entire input vector has been presented at the end of the operation cycle. The basic neuron scheme that implements that functionality is presented in figure 4. The light pulses that reach one neuron are converted to current pulses by the photodetector. A switch, controlled by the corresponding interconnection weight (signal  $W$ ), connect or not the detector with a capacitor, that acts as the storage element. By this way, the product operation is implemented by the switch and the accumulation function of the neuron is implemented by the increasing voltage in the capacitor (equation 1). Additionally there is a switch controlled by signal  $CLR$  that resets the capacitor at the beginning of an operation cycle.



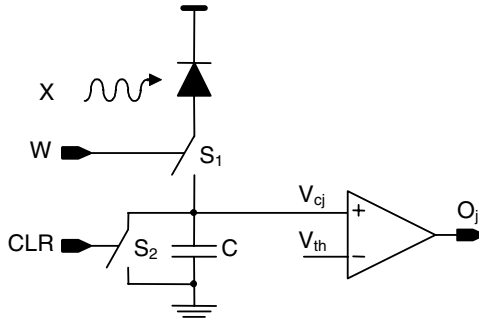


Fig. 4. Basic neuron design

Equation (2) summarizes how the neuron works.

$$\Delta V_c = XW \left( \frac{1}{C} PR\Delta t \right) \tag{2}$$

It represents the increment of the voltage of the capacitor in a time slot which is proportional to the product of input (X) and interconnection weight (W) and the design parameters of the system. They are the value of the storage capacitor (C), the optical power that reaches the detector (P), the responsivity of the detector (R) and the time slot ( $\Delta t$ ). For binary unipolar inputs and interconnection weights, if input X is ‘0’ the voltage in the capacitor does not increase because no light arrives to the detector and no photocurrent is generated; if input X is ‘1’, the voltage in the capacitor increases only if the interconnection is ‘1’ because in this case the detector is connected to the capacitor.

We have designed a previous neuron design that allows  $\Delta t$  to have a maximum value of 10  $\mu s$  [18]. It seems (equation 2) that if we need to decrease  $\Delta t$ , we only need to reduce the value of the storage capacitor C. But doing so we observe injection charge effects due to the commutation of switch S1, we also see unwanted discharge of the capacitor. Additionally a high speed switch must be used.

So the new neuron circuit we propose for the implementation of a fast prototype is based on a differential configuration (see figure 5). The new circuit functionality is the same as the circuit in figure 4: If input is ‘0’ there is no current so the voltage in the capacitor is constant. If input is ‘1’ there is a current that increments the charge in the capacitor if interconnection weight W is ‘1’; in this case the voltage applied to the base of transistor Q1 is higher than the voltage in the base of Q2. If interconnection weight W is ‘0’, the voltage applied to the base of Q1 is lower so the current flows through Q1. Figure 5 shows first test of the fast neuron circuitry. Signal CLR that resets the capacitor at the beginning of the operation cycle; we show both inputs (X) and interconnection weights (W) alternate between ‘0’ and ‘1’, the operation cycle is composed of 32 time slots. Last waveform is the voltage in the storage capacitor, which increases when both input and interconnection weight are ‘1’. This new neuron design allows reducing the time slot  $\Delta t$  to 7 ns (three orders of magnitude with respect previous prototype [19]). Emitters and detectors and optical interconnections have also been modified [19].

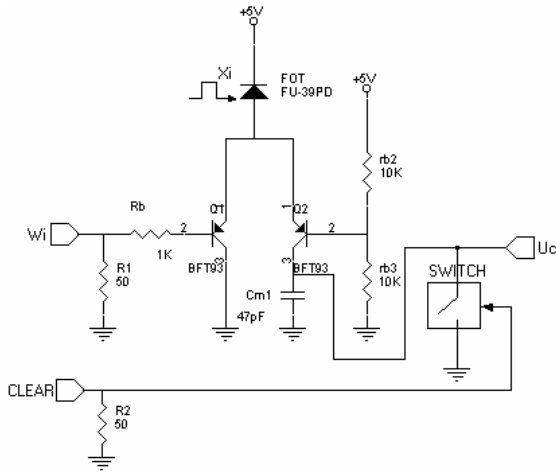


Fig. 5. Fast neuron design

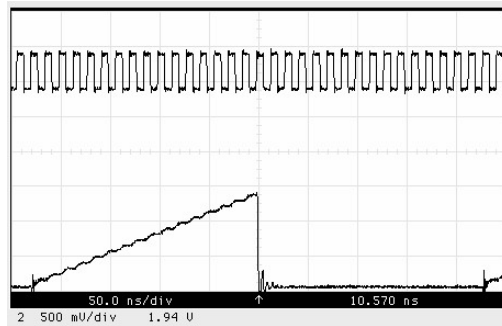


Fig. 6. Neuron measurements

## 4 Fast Vision System

The vision system we have implemented is composed of a CMOS image sensor and the optoelectronic broadcast neural network configured as a Hamming classifier. A controller provides control signals. This system captures an image and classifies it between four classes that are represented by a sample pattern [16]. As the CMOS image sensor works as a random access analog memory, its control signals are the same that are used to the sample patterns memory and are independent of the size of the optoelectronic classifier. To build a high speed system we need to use a fast sample patterns memory (TI SN74V273-6PZA). The block diagram of the fast optoelectronic neural network for vision applications is presented in figure 7. The optoelectronic part is composed by: (1) high speed laser diode (Mitsubishi FU-427SLD) with fiber pigtail and 155 Mb/s laser driver (based on MAX3263 device), (2) 1:16 fiber coupler, and (3) 2 GH discrete InGaAs PIN photodiodes integrated in a pig-tail module (Mitsubishi FU-39PD) with associated neuron circuitry (figure 5).

The vision system operation is benefited by the use of a FIFO memory. There are two steps. First user selects input and sample pattern and download it to the FIFO memory. Second is the operation cycle of the optoelectronic classifier, where the FIFO memory is read at a rate of 155 MHz, that gives an input pattern element value  $X$  and all corresponding sample patterns elements ( $W_j$ ). Results are presented in figure 8, we see the evolution of the voltage on the storage capacitor for different input ( $X$ ) and sample patterns ( $W$ ) that are  $64 \times 64$  pixel resolution ( $N=4096$  elements). Next step is to provide input image ( $X$ ) from the CMOS sensor as described in [16].

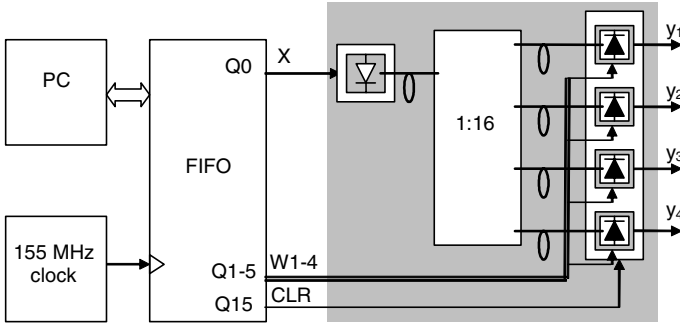


Fig. 7. Fast vision system

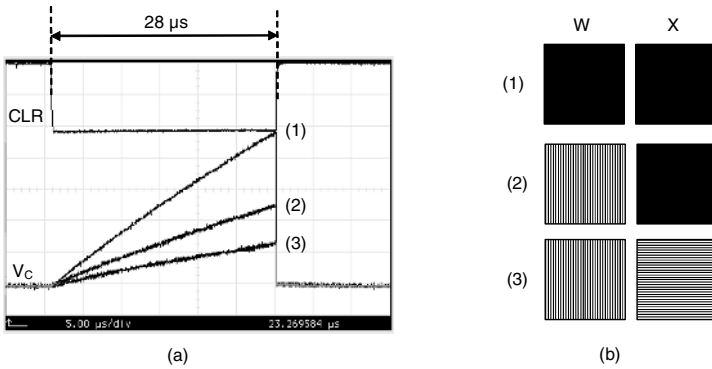


Fig. 8. Fast vision system preliminary results: (a) waveforms, (b) input and sample patterns

### 5 Conclusions

We have proposed a novel optoelectronic architecture for neural networks whose main advantage over electronic counterparts is that it is scalable in speed and size. Optical interconnections benefits the processor size as the number of neurons or processing nodes that compound the hardware architecture can be made very high as electronic wires are removed from the circuit. Additionally optical interconnects promise higher bandwidth than electrical interconnects [13], so input signal distribution can be made faster.

We have designed a fast optoelectronic circuit for neurons and a fast system controller. The control signals that are needed do not depend on the number of processing elements that compound the optoelectronic neural network architecture, which also help on the scaling of the system. We have shown that this fast prototype is able to perform the comparison of a 64×64 pixel resolution input image between a set of sample patterns in 28  $\mu$ s, which is three orders of magnitude faster than our previous prototype [18][19]. The higher number of processing nodes the higher the number of classes, but the classification speed remains constant.

## References

- [1] Malamas EN, Petrakis EGM., Zervakis M, Petit L, Legat J. A survey on industrial vision systems, applications and tools. *Image and Vision Computing* 21 (2003) 171-188.
- [2] Lippmann RP. An introduction to computing with neural nets. *IEEE ASSP magazine* (April 1987) 4-22.
- [3] Carver Mead. *Analog VLSI and neural systems*. Addison-Wesley, cop. 1989. ISBN 0201059924.
- [4] Christodoulou C, Bugmann G, Clarkson TG. A spiking neuron model: applications and learning. *Neural networks* 15 (2002), 891-908.
- [5] Hammerstron D. A VLSI architecture for High-performance, low cost, on-chip learning. In *International Joint Conference on Neural Networks*, (1990).
- [6] Holler M, Tam S, Castro H, Benson R. An electrically trainable artificial neural network (ETANN). In *International Joint Conference on Neural Networks*, (1989).
- [7] Lyndsey C, Lindblad T. Review of hardware neural networks: a user perspective. *Int J neural systems* 6 (1995) 215-224.
- [8] Hendry DC, Duncan AA, Lighttowler N. IP core implementation of a self-organizing neural network. *IEEE trans neural networks* 14 (2003) 1085-1096.
- [9] *Zero Instruction Set Computer Technology Reference Manual*. General Vision Inc. Revision date: September 2003.
- [10] Yang F, Paindavoine M. Implementation of an RBF Neural Network on Embedded Systems. *IEEE trans. Neural Networks* 14 (2003) 1162-1175.
- [11] F.T.S. Yu, D.A. Gregory. *Optical Pattern Recognition: Architectures and Techniques*". *Proceedings of the IEEE* 84 (1996), 733-752.
- [12] Caulfield HJ, Kinser J, Rogers SK. *Optical Neural networks*. *Proceeding of the IEEE* 77 (1989), 1573-1583.
- [13] Miller DAB. *Rationale and Challenges for Optical Interconnects to Electronic Chips*. *Proceedings of the IEEE* 88 (2000), 728-749.
- [14] H. Lamela, M. Ruiz-Llata, C. Warde, *Optical broadcast interconnection neural network*. *Optical Engineering Letters* 42 (2003), 2487-2488.
- [15] H. Lamela, M. Ruiz-Llata, C. Warde. *Prototype Optoelectronic Neural Network for Artificial Vision Systems*. In *Proceedings of IECON02* (2002)
- [16] M. Ruiz-Llata, H. Lamera. *Image classification system based on the optical broadcast neural network processor*. *Applied Optics* (accepted for publication in December 2004)
- [17] M. Ruiz-Llata, H. Lamela, C. Warde. *Design of a Compact Neural Network Based on an Optical Broadcast Architecture*. *Optical Engineering* (accepted in November 2004)
- [18] M. Ruiz-Llata, D.M. Cambre, H. Lamela. *Prototype optoelectronic Hamming Neural Network*. In *International Joint Conference on Neural Networks* (July 2004).
- [19] H. Lamela, M. Ruiz-Llata, D.M. Cambre, C.Warde. *Fast prototype of the optical broadcast interconnection neural network architecture*. *Proc. of SPIE* 5557 (2004), 247-254.

# A Computational Tool to Test Neuromorphic Encoding Schemes for Visual Neuroprostheses

Christian A. Morillas<sup>1</sup>, Samuel F. Romero<sup>1,2</sup>, Antonio Martínez<sup>1</sup>,  
Francisco J. Pelayo<sup>1</sup>, and Eduardo Fernández<sup>3</sup>

<sup>1</sup> Dept. of Computer Architecture and Technology, University of Granada, Spain  
{cmorillas, sromero, amartinez}@atc.ugr.es, fpelayo@ugr.es

<sup>2</sup> Dept. of Computer Sciences, University of Jaén, Spain  
sromero@ujaen.es

<sup>3</sup> Dept. of Histology and Inst. of Bioengineering, University Miguel Hernández,  
Alicante, Spain  
e.fernandez@umh.es

**Abstract.** Recent advances in arrays of microelectrodes open the door to both better understanding of the way the brain works and to the restoration of damaged perceptive and motor functions. In the case of sensorial inputs, direct multi-channel interfacing with the brain for neuro-stimulation requires a computational layer capable of handling the translation from external stimuli into appropriate trains of spikes. The work here presented aims to provide automated and reconfigurable transformation of visual inputs into addresses of microelectrodes in a cortical implant for the blind. The development of neuroprostheses such as this one will contribute to reveal the neural language of the brain for the representation of perceptions, and offers a hope to persons with deep visual impairments. Our system serves as a highly flexible test-bench for almost any kind of retina model, and allows the validation of these models against multi-electrode recordings from experiments with biological retinas. The current version is a PC-based platform, and a compact stand-alone device is under development for the autonomy and portability required in chronic implants. This tool is useful for psychologists, neurophysiologists, and neural engineers as it offers a way to deal with the complexity of multi-channel electrical interfaces for the brain.

## 1 Introduction

The study of the role of electricity in neural tissues has been a concern of medicine for decades, with revealing precedents in past centuries. The understanding of how the brain encodes information using electrical signals has made significant advances thanks to the studies carried out in the fifties by Stephen Kuffler, in which the receptive fields of retinal ganglion cells in cats were determined. In 1958, Hubel and Wiesel recorded electrical activity in primary visual cortex of a cat, using microelectrodes. These studies confirmed that the activity of single neurons encodes information [1]. However, later discoveries showed that there is information not only in the activity of single cells, but also in the interrelated firing of groups of neurons. Popula-

tion coding is still a research field in which (see [2] for a review) a number of groups are working with different approaches. These studies benefit from the development of arrays of multiple electrodes [3, 4, and 5] allowing to record correlated neural activity for controlled stimuli, both in vitro and in vivo setups.

The pairs stimulation-recording of neurons, pave the way to understand the encoding of sensorial signals, which is useful to elicit adequate patterns for stimulation. This is the key for the development of sensorial neuroprostheses, so the uses of microelectrode arrays for recording and stimulation are symbiotic.

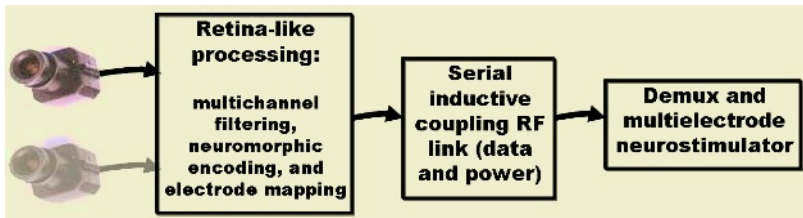
The electrical stimulation of neural tissue is being employed in patients with Parkinson's disease [6] with very long penetrating electrodes inserted in deep brain structures. Another therapeutic application of electrical currents is the restoration of muscle control in spinal cord injuries of patients having a good functional muscular tissue, but damaged neural control of it [7]. Some research laboratories are undertaking investigations to develop artificial limbs with neural control and sensorial feedback for amputees [8, 9].

An exciting field of research is the restoration of sensorial abilities, like audition and vision, by means of electrical stimulation. Cochlear implants are a good sample of this, allowing totally deaf subjects to understand spoken words and even good pronunciation after training. The success in this kind of prostheses [10] has encouraged scientists to guide their next steps to vision, in which additional difficulties arise, due to the complexity and amount of information of images with respect to sound.

## 2 Computational Retina-Like Layer for Stimulating the Neural Interface

The work presented in this paper is developed in the framework of a European project, named CORTIVIS (Cortical Visual Neuroprosthesis for the Blind, <http://cortivis.umh.es>), which aims to build an implant that would automatically deliver stimulation currents to the visual area of the brain cortex in blind individuals (see Fig. 1). This is a challenge that is also being faced with similar and different approaches by several research groups world-around.

Desirable features for such a device are portability, reconfigurability, real-time operation, wireless trans-cranial transmission of data and energy, and the sending of as much information as possible through a limited number of electrode channels [11].



**Fig. 1.** Main blocks of the CORTIVIS prosthetic system, including one or two camera inputs, filtering and coding stages and wireless transmission to one or more implanted arrays of microelectrodes

The last stage of the prosthesis is a neural interface with the brain cortex. In our case, the penetrating microelectrode array of Utah [3] is being considered due to the capability of this device to stimulate the ‘input’ neurons of each column of the cortex structure, located typically in layer IV, 1-1.5 mm beneath the surface.

The handling of multiple channels for stimulation is a complex task in which several abstraction levels must be considered. From an engineering perspective, the lower level is a physical layer regarding the size, shape, composition and biocompatibility of the electrodes. The next upper layer considers the electrical features of the signals to be delivered to each electrode, including amplitude, pulse widths, number of pulses in a burst, etc. The following level in this scheme is a communication protocol layer, concerning the encoding of commands, error correction, energy and data transmission, and housekeeping telemetry. The top of this organization is the computational data layer. Its mission is to translate external stimuli into spike patterns in the form of events to be addressed to the target microelectrodes.

The subject of this paper focuses mostly on this last upper layer. This stage must allow for flexibility, as long as retina and visual pathways have not completely been characterized and multiple approximations are being considered by different research groups. This way, different models with changing parameters can be essayed on the prosthesis.

When designing this top layer for the handling of electrical cortical multi-channel interfaces, we must take into account some desirable features. First of all, the layer must be bio-inspired, that is, compliant with accepted retina models. This level should have images as input and produce spikes timely as output to target electrodes, mimicking the function of a natural retina.

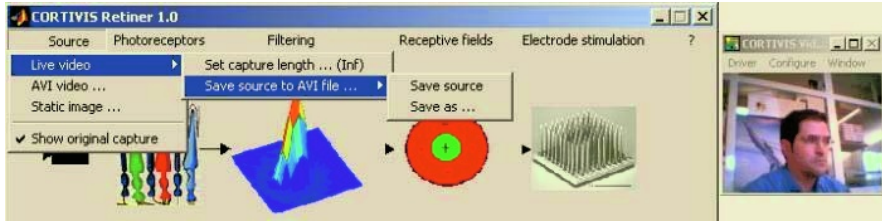
There are a considerable number of parameters implied in retina models, which must be adjusted to produce outputs that are coherent with biological experiments, for the same external stimuli. So flexibility is the second requirement for this layer. Additionally, retina transfer functions are under study to define retina models, so we should allow variations in this stage.

Some experiments are being undertaken, in which in-vitro retinas are exposed to controlled stimuli and the response is recorded. Later, the data is matched and analyzed by using statistical and signal processing procedures [12].

### **3 Retiner: A Test-Bench for Retina Models**

Driven by the main lines described before, we have developed a software platform to easily define and test retina models. This is a program developed under Matlab<sup>TM</sup> that allows entering descriptions of retina models and observing the results of applying them to different inputs. The purpose is to offer the physicians and neuro-physiologists a test-bench to essay retina models and check them against biological experimental data, specifically spiking neurons recordings obtained from natural retinas.

The program, named “Retiner” [13], feeds from a variety of source images, ranging from static picture files in different standard formats (BMP, JPEG, PNG, TIFF...) to AVI video files, or even live-capture from any camera supported by the Microsoft® Video for Windows® driver (Fig. 2).



**Fig. 2.** A view of the Retiner program front-end, showing different choices for image and video input

In the case of live acquisition, the input can be recorded for later matching analysis with the output. Some tests have been carried out with a logarithmic response camera. Common CCD or CMOS sensors, with linear response to light, get saturated in high contrast scenes, causing a loss of information at the entry point of the system, while in the case of logarithmic sensors, the proportional response to high illumination is lower than to dark parts of the image, allowing the camera to register dark and bright objects in the same frame.

Whichever is the input, Retiner separates it into three-color planes corresponding to each kind of photoreceptors, whose gain can be selected independently.

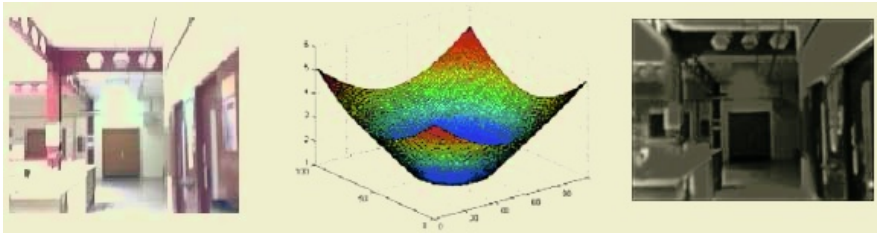
Over these color channels and the intensity channel we can design a set of spatially opponent filters to simulate the function of bipolar cells. This opposition between the center and the periphery of the receptive field of these cells is modeled with a difference-of-Gaussians filter. As our purpose is providing a general test-bench for retina models, we have decided to include different kinds of spatial filters like Gaussians, difference of Gaussians, laplacian of Gaussians and any user-defined expression that Matlab™ can compute. The convolution is the method used in these cases to apply the filter to the image. This operation involves a mask with the weighting factors for a pixel neighborhood.

Retiner includes temporal filtering, which enhances image changes between consecutive frames, thus enhancing those parts of the scene where changes are taking place (due to external movement, ego-motion or both). The temporal enhancement takes as input two consecutive frames. Both frames are pre-processed to reduce noise through a parameterized gaussian blurring filtering. In any case, this previous filter is the same for both frames.

It is possible to select a foveated processing for the temporal enhancement. In such a case, a circular region centered on the image keeps its values, and the rest of the pixel values are magnified or reduced (Fig. 3), so that the pixels out of the circular area are multiplied by an interpolated value. The radius of the fovea and the size of the mask are some of the parameters of this filter that can be varied.

At this moment we are developing a new module to include into Retiner. This module works with two cameras simultaneously, taking advantage of the new Image Acquisition Toolbox included in Matlab™ 7.0 [14]. It approximates depth of the scene based on stereovision.



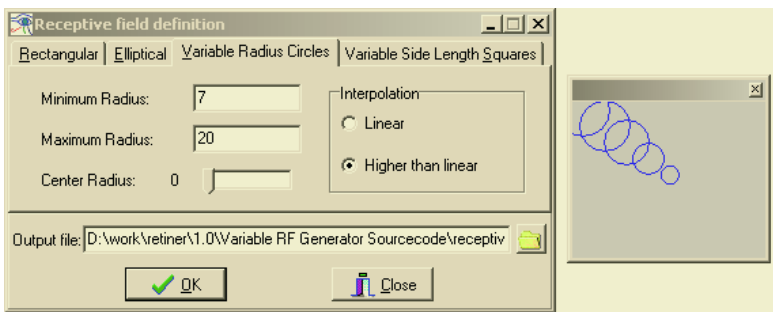


**Fig. 3.** Foveated processing shape for the temporal enhancement filter, which is emphasized in the periphery

The relative distance to the objects in the scene is estimated by computing the disparity between the positions of the same point when captured by both cameras. Disparity can be translated into depth considering that points with smaller disparities are farther than points with greater disparities. The correspondence algorithm has been implemented and enhanced by including a preprocessing stage in order to adjust the cameras. This stage moves the images one over another and gets the global displacement where differences are minimum. It has been included in the implementation because of the difficulty to comply the epipolar requirement [15]. The information obtained can be used to emphasize image areas where closer objects are present. Disparity calculation is very expensive in terms of computing time, and is not able to run in real-time while in live capture for a software model of the retina. However, a hardware implementation of the algorithm will achieve the required real-time processing.

Using the output of the filters we must build an arithmetic combination, including some or all of them, in order to determine the relative importance of the information every one contributes with.

Once we have computed the combination of the filters, the following step is defining the receptive field associated to each electrode of the array, that is, to determine the set of pixels in the image taking part in every electrode's activity. The resolution we can obtain depends on the number of electrodes in the array and on the input image size. For example, with a given size of the image, reducing the number of electrodes means an increasing amount of pixels activating it, and a lower resolution. In

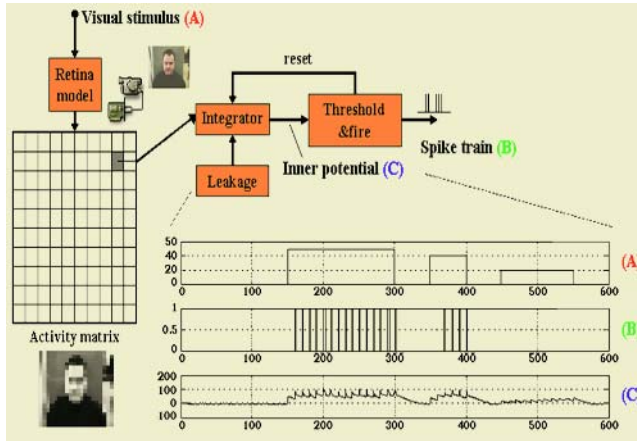


**Fig. 4.** Retiner user interface to select shape and size for the receptive field of every channel

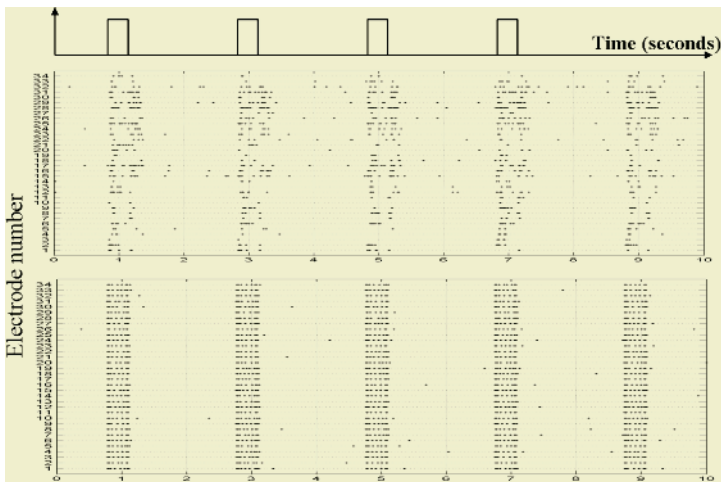
our software different microelectrode array sizes, image sizes, and configurations can be selected for the receptive fields.

Fig. 4 shows the tool designed for the receptive field setup. It allows the following shapes: variable height and width rectangles with any orientation, ellipses with axes of variable length, variable radius (center-periphery) circles, and variable size squares.

The last two options allow us to model features present in the human retina, varying from the fovea, where the correspondence photoreceptor-ganglion cell is near one-to-one, to the outer areas of the retina, where a ganglion cell gathers information from a high number of photoreceptors.



**Fig. 5.** Implementation scheme of the integrate-and-fire spiking neuron model, which applies to each component of the spatio-temporal filtered image (activity matrix)



**Fig. 6.** Full-field flashing stimulation of a retina: (top) stimulus occurrences, (middle) in vivo recording from a rabbit retina, and (bottom) output of Retiner simulation for this stimulus

The function of the next stage is neuromorphic coding: translating activity levels of the electrodes into spike trains that the brain can understand. The implementation is based on an integrate-and-fire spiking neuron model including a leakage term (Fig. 5). Some parameters of this process may be configured, like the threshold each electrode must reach for producing a spike, the resting level after an event, and a leakage factor that decreases the accumulated value when no input is present.

All the spike events produced in a simulation session can be stored in a text file for a later off-line graphical analysis. The file format is compliant with the NEV format from the BIONIC Technologies acquisition software [16]. At this window we also can study spikes generated in any in-vivo recording experiment. For the analysis we can display activity produced at the whole microelectrode array, or select some of them, when interested in a deep analysis on a certain area of the prosthesis (see Fig. 6).

## 4 Discussion

We have presented a new tool that assists neuroengineers in the design of retinal processing prosthetic systems. It has a high degree of flexibility due to the amount of parameters considered, and permits the simulation of processing of visual inputs, and the comparison of the spike-train outputs with records obtained in experiments with biological retinas.

## Acknowledgements

The work here presented has been carried out thanks to the support of the EC Project CORTIVIS (ref. QLK6-CT-2001-00279) and the National Spanish Grant (ref. DPI 2004-07032).

## References

1. Hubel D., *Eye, Brain, and Vision*, Scientific American Library, 1988.
2. Victor J., *Temporal aspects of neural coding in the retina and lateral geniculate*, Network. 1999 Nov;10(4):R1-66. Review. PMID: 10695759.
3. Maynard E.M., Nordhausen C.T., Norman R. A., *The Utah intracortical electrode array: a recording structure for potential brain-computer interfaces*, Electroenceph. Clin. Neurophysiol. 102:228-239 (1997).
4. Microelectrodes from Thomas Recording. <http://www.thomasrecording.com>
5. M. Ghovanloo and K. Najafi, *A modular 32-site wireless neural stimulating microsystem*, IEEE International Solid-State Circuits Conference, Digest of Technical Papers, pp. 226-227, Feb. 2004.
6. Carsten R. Bjarkam and Jens Chr. Sørensen and Niels Å. Sunde and Finn A. Geneser and Karen Østergaard, *New strategies for the treatment of Parkinson's disease hold considerable promise for the future management of neurodegenerative disorders*, Biogerontology 2: 193-207, 2001. Kluwer Academic Publishers.
7. EC Project SUAW: Stand Up And Walk. <http://www.suaw.com>

8. Jose M. Carmena, Mikhail A. Lebedev, Roy E. Crist, Joseph E. O'Doherty, David M. Santucci, Dragan F. Dimitrov, Parag G. Patil, Craig S. Henriquez, Miguel A. L. Nicolelis, *Learning to control a Brain-Machine Interface for Reaching and Grasping by Primates*, PLoS Biology, Vol. 1, Issue 2, October 2003.
9. The Cyberhand Project, <http://www.cyberhand.org>
10. Shepherd R. K., (2003). *Central Auditory Prostheses. Neuroprosthetics: Theory and Applications*. Eds.: K. W. Horch and G. S. Dhillon. Kluwer Scientific Publishers. ISBN 981-238-022-1
11. L. Sousa, P. Tomás, F. Pelayo, A. Martínez, C. A. Morillas, S. Romero, *Bioinspired Stimulus Encoder for Cortical Visual Neuroprostheses*, Chapter 22 of the book *New Algorithms, Architectures, and Applications for Reconfigurable Computing*. Ed. P. Lysaght and W. Rosenstiel. Springer. December 2004. ISBN: 1-4020-3127-0.
12. Bongard M., Ferrandez J.M., Bolea J.A., Ammermüller J., Fernandez E., *Extracting common features from the responses of different ganglion cell populations*. *Ophthal Res* 34/S1: 363.
13. F.J. Pelayo, S. Romero, C. Morillas, A. Martínez, E. Ros, E. Fernández, *Translating Image Sequences into Spike Patterns for Cortical Neuro-Stimulation*, *Neurocomputing*, Volume 58-60 (June 2004), special issue on Computational Neuroscience: Trends in Research 2004, pp. 885-892, ISSN: 0925-2312.
14. The Mathworks – Matlab and Simulink for technical computing. <http://www.mathworks.com>
15. Faugeras O., *Three-Dimensional Computer Vision*, MIT Press, 1993.
16. BIONIC Technologies LLC (now Ciberkinetics Inc.) <http://www.ciberkineticsinc.com>

# Test Infrastructure for Address-Event-Representation Communications

R. Paz, F. Gomez-Rodriguez, M.A. Rodriguez,  
A. Linares-Barranco, G. Jimenez, and A. Civit

Departamento de Arquitectura y Tecnología de Computadores,  
Universidad de Sevilla,  
Av. Reina Mercedes s/n, 41012-Sevilla, Spain  
rpaz@atc.us.es  
<http://www.atc.us.es>

**Abstract.** Address-Event-Representation (AER) is a communication protocol for transferring spikes between bio-inspired chips. Such systems may consist of a hierarchical structure with several chips that transmit spikes among them in real time, while performing some processing. To develop and test AER based systems it is convenient to have a set of instruments that would allow to: generate AER streams, monitor the output produced by neural chips and modify the spike stream produced by an emitting chip to adapt it to the requirements of the receiving elements. In this paper we present a set of tools that implement these functions developed in the CAVIAR EU project.

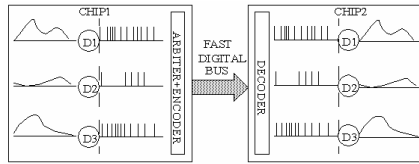
## 1 Introduction

Address-Event-Representation (AER) was proposed in 1991 by Sivilotti [1] for transferring the state of an array of neurons from one chip to another. It uses mixed analog and digital principles and exploits pulse density modulation for coding information. The state of the neurons is a continuous time varying analog signal.

Fig. 1 explains the principle behind the AER. The emitter chip contains an array of cells (like, e.g., an imager or artificial retina chip) where each pixel shows a state that changes with a slow time constant (in the order of milliseconds). Each pixel includes an oscillator that generates pulses of minimum width (a few nanoseconds). Each time a pixel generates a pulse (called "event"), it communicates with the periphery and its address is placed on the external digital bus (the AER bus). Handshaking lines (Acknowledge and Request) are used for completing the communication.

In the receiver chip the pulses are directed to the pixels or cells whose address was on the bus. This way, pixels with the same address in the emitter and receiver chips will "see" the same pulse stream. The receiver cell integrates the pulses and reconstructs the original low frequency continuous-time waveform.

Transmitting the pixel addresses allows performing extra operations on the images while they travel from one chip to another. For example, inserting memories (e.g. EEPROM) allows transformations of images.

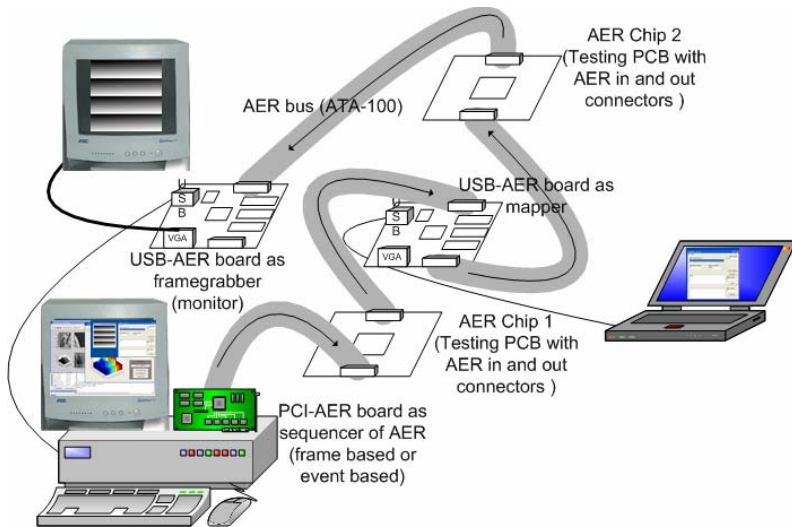


**Fig. 1.** AER inter-chip communication scheme

There is a growing community of AER protocol users for bio-inspired applications in vision and audition systems, as demonstrated by the success in the last years of the AER group at the Neuromorphic Engineering Workshop series [2]. The goal of this community is to build large multi-chip hierarchically structured systems capable of performing complicated array data processing in real time. To make these objectives possible it is essential to have a set of instruments that would allow to:

- Sequence: Produce synthetic AER event streams that can be used as controlled inputs while testing and adjusting a chip or set of chips.
- Monitor: Observe the output of any element in the system.
- Map: Alter the stream produced by an emitter and send the modified stream to a receiver

For these purposes we have designed and implemented two different instruments: a PCI board capable of sequencing and monitoring events at a rate of over 15Mevents/s and a versatile board that can be used for sequencing, monitoring and mapping. This last board can be used either in a stand alone mode or connected to an external



**Fig. 2.** AER tools usage scenario

computer through a USB bus. A possible scenario for these tools is shown in fig. 2 where a computer with a PCI-AER board produces output for AER chip1. The output from this chip is remapped by a USB-AER board and fetched to AER chip 2. The stream produced by chip 2 is monitored by another USB-AER board which can send its output directly to a VGA monitor or to a computer through USB bus.

## 2 Sequencing and Monitoring AER Events

To be useful for debugging an AER tool should be able to receive and also send a long sequence of events interfering as little as possible with the system under test. Let's start explaining the meaning of interfacing in the context.

As neurons have the information coded in the frequency (or timing) of their spikes, the pixels that transmit their address through an AER bus also have their information coded in the frequency of appearance of those addresses in the bus. Therefore, interspike-intervals (ISIs) are critical for this communication mechanism. Thus, a well designed tool shouldn't modify the ISIs of the AER.

The ISIs may be difficult to preserve depending on the nature of the emitter and/or receiver chips. Let's suppose the case of having an AER emitter chip connected to an AER receiver chip, and we want to debug their communication. In principle, there are two possibilities: connecting to the bus an AER sniffer element, or to introducing a new AER element in between the emitter and the receiver.

- The sniffer element will consist on an AER receptor that captures the address and stores it with a timestamp in memory for each request that appears on the AER bus. The problem in this case is that the speed of the emitter and receiver protocol lines could be faster than the maximum speed supported by the sniffer (15 ns per event in some existing chips), causing events to be lost. Another typical problem could be that the throughput of the AER bus (unknown in principle) would be so high that the interface memory cannot be downloaded to the computer's memory on time. This also implies that events are lost.
- The other possibility is to introduce a new AER element between the two chips. In this case the emitter sends the event to the AER element and the AER element sends the same event to the receiver chip. The problem now is that the new AER element will always introduce a delay in the protocol lines, and may also block the emitter if it is not able to keep up with its throughput. Therefore, ISIs are not conserved. But the behaviour will be the same than if we connect the emitter to a slower receiver.

The throughput problem requires using very fast PC interfaces and the problem of very fast emitter or receiver protocols can be reduced by using a very high frequency clock for the stages that interface with the AER protocols.

## 3 PCI-AER Interface

Before the development of our tools the only available PCI-AER interface board was developed by Dante at ISS-Rome (See [2]). This board is very interesting as it embeds

all the requirements mentioned above: AER generation, remapping and monitoring. Anyhow its performance is limited to 1Mevent/s approximately. In realistic experiments software overheads reduce this value even further. In many cases these values are acceptable but, currently many address event chips can produce (or accept) much higher spike rates.

As the Computer interfacing elements are mainly a monitoring and testing feature in many address event systems, the instruments used for these proposes should not delay the neuromorphic chips in the system. Thus, speed requirements are at least 10 times higher than those of the original PCI-AER board. Several alternatives are possible to meet these goals:

- Extended PCI buses.
- Bus mastering.
- Hardware based Frame to AER and AER to Frame conversion.

When the development of the CAVIAR PCI-AER board was started, using 64bit/66MHz PCI seemed an interesting alternative as computers with this type of buses were popular in the server market. When we had to make implementation decisions the situation had altered significantly. Machines with extended PCI buses had almost disappearing and, on the other hand, serial LVDS based PCI express [3] was emerging clearly as the future standard but almost no commercial implementations were in the market. Therefore, the most feasible solution was to stay with the common PCI implementation (32 bit bus at 33MHz) and consider PCI express for future implementations. Speed improvements, thus, should come from the alternative possibilities.

The previously available PCI-AER board uses polled I/O to transfer data to and from the board. This is possibly the main limiting factor on its performance. To increase PCI bus mastering is the only alternative. The hardware and driver architecture of a bus mastering capable board is significantly different, and more complex, than a polling or interrupt based implementation.

Hardware based frame to AER conversion doesn't increase PCI throughput but, instead, it reduces PCI traffic. First some important facts have to be explained. It is well known that some AER chips, especially grey level imagers where pulse density is proportional to the received light intensity, require a very large bandwidth. This is also the case of many other chips when they are not correctly tuned. For example let's consider a Grey level 128\*128 imager with 256 grey levels. In a digital frame based uncompressed 25fps format, it would require a bandwidth of  $128*128*25=0.39\text{MBytes/s}$ . The maximum requirements for an "equivalent" system that would output AER supposing the number of events in a frame period is equal to the gray level and considering the worst case where all pixels spike with maximum rate is:

$$2\text{bytes/event}*256\text{events/pixel}*number\ of\ pixels/\ frame\ period= 200\text{MBytes/s}$$

The meaning of this figure should be carefully considered. A well designed AER system, which produces events only when meaningful information is available, can be very efficient but, an AER monitoring system should be prepared to support the bandwidth levels that can be found in some real systems. These include systems that have not been designed carefully or that are under adjustment. Currently the available



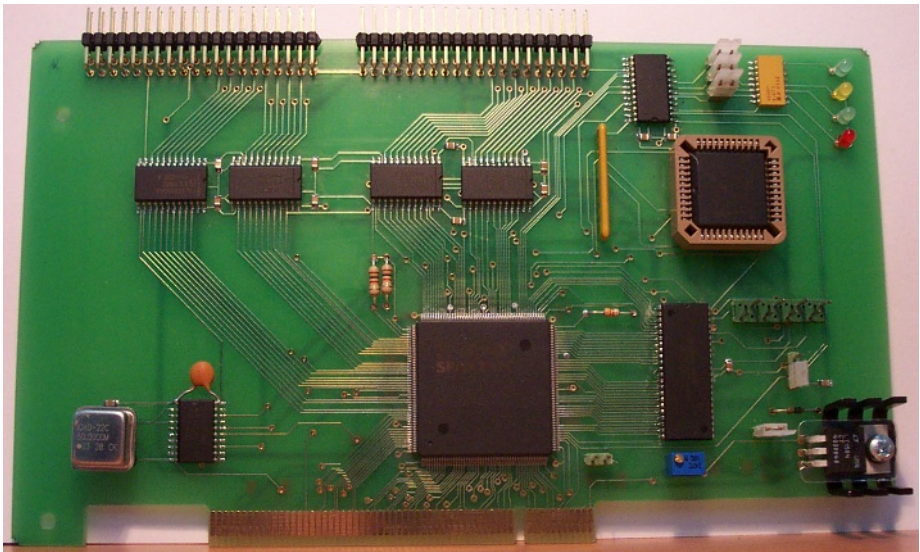
spike rates, even in these cases, are far from the value shown above but, some current AER chips may exceed the 40Mevent/s in extreme conditions.

The theoretical maximum PCI32/33 bandwidth is around 133Mbytes/s. This would allow for approximately 44Mevent/s considering 2 bytes per address and two bytes for timing information. Realistic figures in practice are closer to 20Mbyte/s. Thus, in those cases where the required throughput is higher a possible solution is to transmit the received information by hardware based conversion to/from a frame based representation. Although this solution is adequate in many cases, there are circumstances where the developers want to know precisely the timing of each event, thus both alternatives should be preserved.

Implementing AER to Frame conversion is a relatively simple task as it basically requires counting the events over the frame period. Producing AER from a frame representation is not trivial and several conversion methods have been proposed [4][5].

The theoretical event distribution would be that where the number of events for a specific pixel is equal to its associated grey level and those events are equally distributed in time. The normalized mean distance from the theoretical pixel position in time to the resulting pixel timing with the different methods is an important comparison criterion. In [6] it is shown that, in most circumstances, the behavior of the methods is similar and, thus, hardware implementation complexity is an important selection criterion. From the hardware implementation viewpoint random, exhaustive and uniform methods are especially attractive.

As a result of these considerations the design and implementation of the CAVIAR PCI-AER board was subdivided into a set of intermediate steps in which initially no mastering was implemented. Later Bus mastering was included and hardware based frame to AER conversion was included as a last step.



**Fig. 3.** CAVIAR PCI-AER board

The physical implementation of all the steps is equal. They differ in the VHDL FPGA code and in the operating system dependant driver. The first design was a VIRTEX based board which was completely redesigned after the first tests. It was established that most of the functionality, demanded by the users, could be supported by the larger devices in the less expensive SPARTAN-II family. The Spartan Version of the board is shown in fig. 3.

Currently a Windows driver that implements bus mastering is being tested. The Linux version with bus mastering is still under development. An API that is compatible, as much as permitted by the different functionality, with that used in the current PCI-AER board has been implemented. MEX files to control the board from MATLAB have also been developed.

### 4 USB-AER

The CAVIAR PCI-AER board can perform Address Event sequencing and monitoring functions but has no hardware mapping capabilities. Although software based mapping is feasible a specific device for this purpose is needed if we want to build AER systems that can operate without requiring any standard computer. This standalone operating mode requires to be able to load the FPGA and the mapping RAM from some type of non volatile storage that can be easily modified by the users. MMC/SD cards used in digital cameras are a very attractive possibility. However in the development stage the users prefer to load the board directly from a computer and, for this purpose USB seems the most suitable solution.

Many AER researchers would like to demonstrate their systems using instruments that could be easily interfaced to a laptop computer. This requirement can also be supported with the USB-AER board as it includes a relatively large FPGA that can be loaded from MMC/SD or USB, a large SRAM bank and two AER ports. Thus the board can be used also as a sequencer or a monitor. Due to the bandwidth limitations of full speed USB (12Mbit/s) Hardware based event to frame conversion is essential in this board for high, or even moderate, event rates.

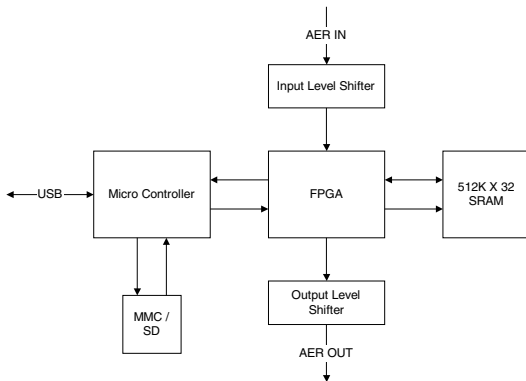


Fig. 4. USB-AER Block Diagram

The USB-AER board is based around a Spartan-II 200 Xilinx FPGA, with a 512K\*32 12ns SRAM memory bank. The board uses a Silicon Laboratories C8051F320 microcontroller to implement the USB and the MMC/SD interface. A simple VGA monitor interface is also provided to allow the board to act as a monitor (frame grabber). A block diagram of the board is shown in fig. 4.

The board will act as a different device according to the module that is loaded in the FPGA either through a MMC/SD card or from the USB bus. Currently the following Modes are implemented:

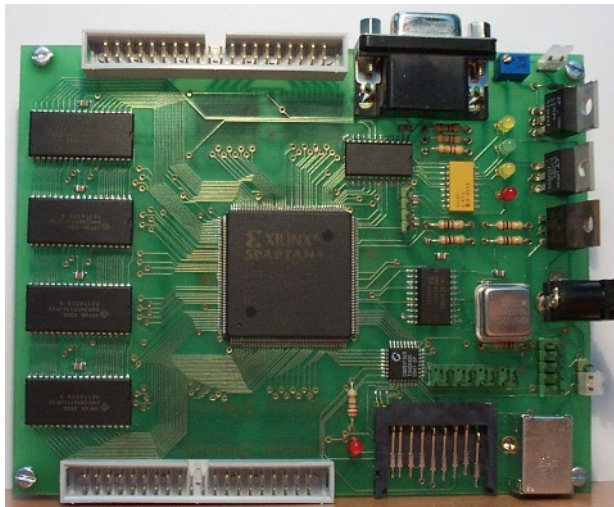
- Mapper: 1 event to 1 event and 1 event to several events.
- Monitor (frame-grabber): using either USB or VGA as output.
- Sequencer: based on hardware frame to AER conversion using the Random or Exhaustive method.

The power consumption is around 220mA (1.63W) working as random sequencer. And its size is 100x130 mm.

Modules are under development to capture sequences of up to 512K events with timestamps and to play these sequences back. These modules are very interesting when a researcher wants to use the output stream produced by a chip from another researcher as input to his or her chip.

This new board was interfaced in Telluride 04 [7] to the current version of the CAVIAR retina and to an imager developed at JHU. Later in the CAVIAR meeting in September 04 it was interfaced to the remaining project chips. The USB-AER board is shown in fig. 5.

A simple interface to control this board is available under windows. It allows loading modules into the FPGA, uploading or downloading data to the FPGA, and showing the received images when the board acts as a monitor.



**Fig. 5.** USB-AER Board

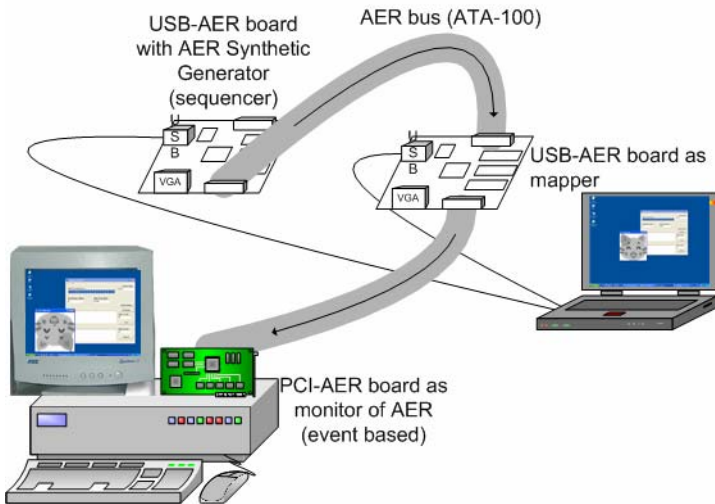
A Linux driver for the USB-AER is currently under test. With this driver the USB-AER board can be easily integrated with several MATLAB applications developed at INI [8].

## 5 Conclusions

A set of tools has been developed that allow efficient testing and demonstration of address event based systems. A demonstration scenario is shown in fig. 6. In this case a USB-AER board is generating a stream of events that correspond to the image shown in Fig. 6 in the laptop screen, which is rotated by a second USB-AER board used as a mapper to produce an AER event stream, which is captured by a PCI-AER board used as a monitor to produce the output shown in the desktop PC screen.

In this scenario only the presented tools are shown. In real world cases the tools are used to evaluate or tune neural chips. In the CAVIAR project the chips have been interfaced to two different retinas, a convolution chip, a winner take-all chip and a learning chip.

Although the AER protocol is asynchronous, the VHDL codes for both boards have been developed synchronously, due that both the PCI and the USB buses are synchronous. To avoid the metastability two cascade flip-flops have been used for the input protocol lines.



**Fig. 6.** Demonstration Scenario

**Acknowledgements.** This work was in part supported by EU project IST-2001-34124 (CAVIAR), and Spanish project TIC-2003-08164-C03-02 (SAMANTA).

## References

1. M. Sivilotti, Wiring Considerations in analog VLSI Systems with Application to Field-Programmable Networks, Ph.D. Thesis, California Institute of Technology, Pasadena CA, 1991.
2. A. Cohen, R. Douglas, C. Koch, T. Sejnowski, S. Shamma, T. Horiuchi, and G. Indiveri, Report to the National Science Foundation: Workshop on Neuromorphic Engineering, Telluride, Colorado, USA, June-July 2001. [[www.ini.unizh.ch/telluride](http://www.ini.unizh.ch/telluride)]
3. A. Wilen, J. Schade, R. Thornburg. "Introduction to PCI Express", Intel Press, 2003.
4. A. Linares-Barranco. Estudio y evaluación de interfaces para la conexión de sistemas neuromórficos mediante Address- Event-Representation. Ph.D. Thesis, University of Seville, Spain, 2003
5. A. Linares-Barranco, R. Senhadji-Navarro, I. García-Vargas, F. Gómez-Rodríguez, G. Jimenez and A. Civit. Synthetic Generation of Address-Event for Real-Time Image Processing. ETFA 2003, Lisbon, September. Proceedings, Vol. 2, pp. 462-467.
6. Linares-Barranco, A.; Jimenez-Moreno, G.; Civit-Ballcels, A.; Linares-Barranco, B.; On synthetic AER generation. ISCAS '04. Proceedings of the IEEE 2004 May 2004 Pages:V-784 - V-787 Vol.5
7. Avis Cohen, et.al., Report on the 2004 Workshop On Neuromorphic Engineering, Telluride, CO. June - July , 2004 [[www.ini.unizh.ch/telluride/previous/report04.pdf](http://www.ini.unizh.ch/telluride/previous/report04.pdf)]
8. M. Oster, Serverbased Software Architecture for AER systems [<http://www.ini.unizh.ch/~mao/AerSoftware/SoftwareOverview.pdf>]

# Automatic Generation of Bio-inspired Retina-Like Processing Hardware

Antonio Martínez<sup>1</sup>, Leonardo M. Reyneri<sup>3</sup>, Francisco J. Pelayo<sup>1</sup>, Samuel F. Romero<sup>2</sup>, Christian A. Morillas<sup>1</sup>, and Begoña Pino<sup>1</sup>

<sup>1</sup> Dept. Architecture and Computer Technology,  
C/Periodista Daniel Saucedo Aranda s/n E-18071, Spain  
`amartinez@atc.ugr.es`

<sup>2</sup> Dept. of Computer Science, School of Computing,  
Edif. A3, 133, University of Jaen. Campus las Lagunillas, s/n  
`sromero@ujaen.es`

<sup>3</sup> Dept. of Electronics, Politecnico di Torino,  
Corso Duca degli Abruzzi 24,  
10129 Turin, Italy  
`leonardo.reyneri@polito.it`

**Abstract.** This paper describes a tool devised for automatic design of bioinspired visual processing models using reconfigurable digital hardware. The whole system is indicated for the analysis of vision models, especially those with real-time requirements. We achieve a synthesizable FPGA/ASIC design starting from a high level description of a retina, which is made and simulated through an ad-hoc program. Our tool allows a thorough simulation of the visual model at different abstraction levels, from functional simulation of the visual specifications up to hardware-oriented simulation of the developed FPGA model. The main objective of this work is to build a portable and flexible system for a visual neuroprosthesis and to stimulate efficiently an array of intra-cortical implanted microelectrodes. A set of parameters can be adjusted in every step of the design flow in order to maximize the design flexibility of the model. Furthermore these parameters allow the different scientists who have to deal with the development to modify a well known characteristic.

## 1 Introduction

Visual rehabilitation of vision impaired people is one of the major challenges in biomedical engineering, and specially in the field of neuroengineering, more oriented to the development of neural prosthetic systems. The work presented here is being carried out within the EC project *CORTIVIS* (QLK6-CT-2001-00279) (see [1]), which has as a main objective the development of a cortical visual neuroprosthesis for the blind.

A project of this kind opens up a large interface where different sciences must coexist and interact with the design of the visual system. We present a work which deals with this large interface and performs a top-down design flow

(see fig. 1) taking into account the plurality of sciences and different levels of abstraction. Although available for a large number of computer vision schemes, we focus our attention to the development of a visual bio-inspired neuroprosthesis. The result is a general system for specification, simulation and testing of general-purpose computer vision models.

### 1.1 Justification of the Selected Implementation

The huge computing requirements for real-time retina-like digital processing of visual information forced us to work with the FPGA hardware technology in order to achieve the required real-time specifications of a bio-inspired visual system. FPGA technology allows an easy projection of a vision functional model onto reconfigurable hardware platforms. This technology also brings the opportunity of modeling our particular system adapting the real biological parallelism schemes for processing the visual information. The characteristics of a portable neuro-prosthesis ask for a very configurable system able to be adapted to every individual implanted. In that way, the possibility of changing the main processing parameters of the final hardware system is very important to syntonize with every patient peculiarity.

### 1.2 Modeling the Visual System

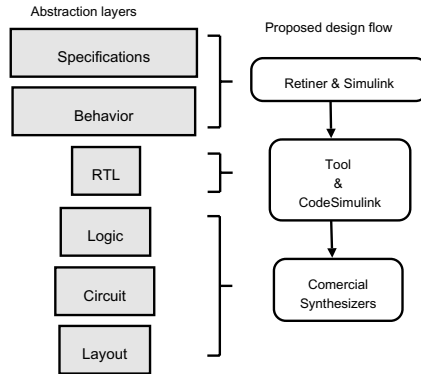
We have used *Retiner* (see [2]), a tool for modeling, simulating, testing and comparing vision models developed by some of the authors during the *CORTIVIS* project. This program is primarily written in *Matlab* and allows us to test the results of different computer vision models (therefore retina schemes) with many types of inputs, from static images (in standard formats like BMP, JPEG, PNG, TIFF, ...) up to live video (using a normal webcam) or a simple AVI video file. The main purpose of *Retiner* is to offer the physicians and neuro-physiologists a test-bench to essay retina models and check them against biological experimental data.

Our work in this paper has been to adapt *Retiner* to be inserted in the top-down design flow displayed in fig. 1, which spans from a functional specification to a real hardware representation of the system.

## 2 Tool Description

The software structure of the tool is designed following a component model in order to a better debugging and organization of every task. We present both, the simulation and synthesis schemes of our tool. In this paper we focus our attention in the medical purpose of the tool. We present the simulation and synthesis schemes that are used to perform our tasks. Fig. 1 represents the design flow that is being managed compared to the typical abstraction layers in the design of integrated circuits area.

We start the design from *Retiner* which allows us to easily make high level descriptions of image processing retina-like operations. Once we check that the



**Fig. 1.** Typical design flow using our tool

design is giving coherent results, we can proceed to run the tool that is described in this paper, to automatically obtain a hardware design in a standard description language (VHDL) that can be later used to program a real-time working circuit implementing our design, by means of third party commercial FPGA programming tools. The well-known AlphaData RC1000 PCI board, which contains a VirtexE 2000 FPGA chip, is being using to test the tool.

## 2.1 Synthesis Tasks

Fig. 2 represents the synthesis scheme that follows our tool.

Another example of applications that allow a high level specification of a system using *Simulink* blocks which can be translated into hardware can be consulted in [3, 4].

This work is in the context of a larger translation strategy (see [5]) but in this paper we are very interested in the way biomedical specialists from the project *CORTIVIS* can obtain a suitable artificial retina in reconfigurable hardware.

From a retina-like description defined by *Retiner* in a mathematical way we start the translation routines of our tool. After determining the desired mathematical behavior of the retina, a certain number of spatio-temporal filters can be selected to define any *Matlab*-compatible mathematical expression. Up to this state of the work, a linear weighted combination of the filters can be properly processed. The reference [6] can be considered for consulting the overall hardware scheme. A typical class of filters used to mathematically describe retinal functions is the *Difference of Gaussians (DoG)*:

$$DoG(X_1, X_2, Q_1, Q_2) \triangleq X_1 \otimes Q_1 - X_2 \otimes Q_2 \quad (1)$$

where  $\otimes$  indicates convolution, while  $Q_1$  and  $Q_2$  are two appropriate filter kernels. The *DoG* operator is available in *Retiner* software. In this way a general retina description could have this look:

$$F_1 = DoG(R + G, B; \dots) \quad (2)$$



$$F_2 = DoG(R + B, G; \dots) \quad (3)$$

$$F_3 = DoG(I, I; \dots) \quad (4)$$

$$\vdots \quad (5)$$

$$F_n = DoG(\dots) \quad (6)$$

$$Retina = a \times F_1 + b \times F_2 + c \times F_3 + \dots \quad (7)$$

with  $R, G, B, I$  standing for red, green and blue color channels,  $I$  standing for light intensity channel and  $a, b, c \in \mathfrak{R}$ .

The ellipsis stand for the necessary parameters that every filter function can have; in this case every difference of gaussians filter (1... $i$ ) will have  $\sigma_1$  and  $\sigma_2$  as the *sigma* parameter of every individual gaussian and a kernel mask of length  $l_i$  standing for the convolution kernel length.

Any linear combination of  $R, G, B$  and  $I$  is allowed in the kernel of every  $DoG$ . A more concrete example of a visual processing system is shown below

$$Model = \frac{3}{2}(4 DoG(2 R + G, B; \sigma_1, \sigma_2, l) + 2 DoG(2 R + G, 2 B; \sigma_1, \sigma_2, l)) \quad (8)$$

This is the main entry point to our translation tool. As is represented in fig. 2 this expression is given to the *HSM-RI* module, an application which starts the synthesis process. This module adapts the mathematical expression to minimize the hardware resources. The process is carried out by cancelling unsuitable products and divisions, building a list of the hardware primitives that will be generated taking into account which of them can be shared and rewriting the expression on the hardware side.

Once the equation 8 has been processed, it becomes:

$$Model = 9 \cdot Gauss(I_1; \sigma_1, l) - 6 \cdot Gauss(I_2; \sigma_2, l) - 3 \cdot Gauss(I_3; \sigma_2, l) \quad (9)$$

with  $I_1 \dots I_3$  the different inputs of the modules.

A simplification and sharing of resources has been done. Further optimizations of the models are possible, depending on the hardware partitioning of every individual primitive.

The corresponding scheme that is generated by the tool is shown in fig. 3. As it can easily be understood there was no need to calculate twice the  $I_1$  signal, so only one signal will feed the necessary primitives. In this context, eq. 9 involves only 3 gaussians instead of the 4 which would generate eq. 8.

At last, *HSM-RI* commanded by *CodeRetiner* module generates a *CodeSimulink*[7, 8, 9, 10] Matlab script which makes the interface with the *CodeSimulink* libraries and is able to generate both a *Simulink* model and a VHDL synthesizable project from its input. Every box in the *Simulink-CodeSimulink* model has a dual in VHDL which does the same, although some boxes can be hierarchical and join different boxes in a single one.

There's a feed-back between *CodeRetiner* and *Retiner* in order to control the process and enable a direct interaction in the synthesis flow. In this way, it is possible to stop the flow in the generation of the *Simulink* model (from *Retiner*)

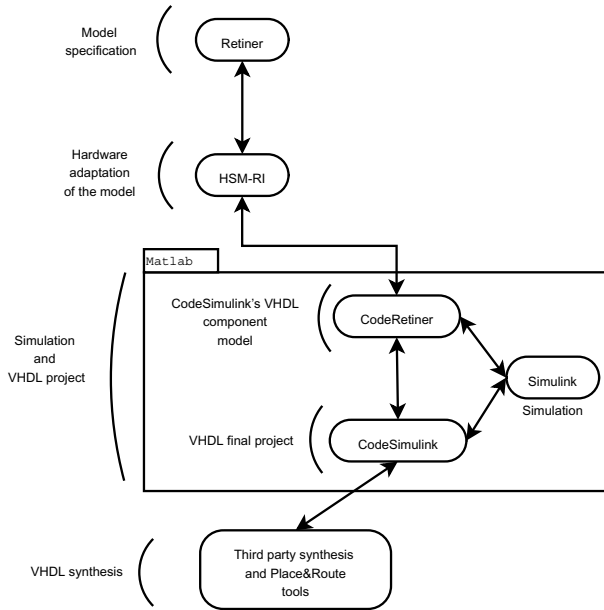


Fig. 2. Synthesis scheme of the tool

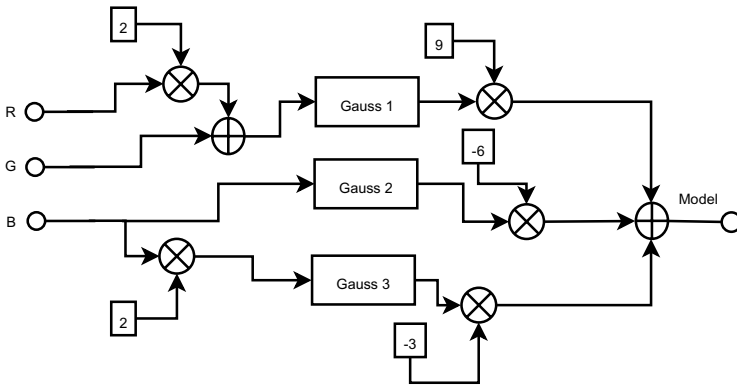


Fig. 3. Example of the structure inferred

for a full precision simulation and stop in the generation of the hardware oriented *Simulink* model in order to perform a hardware simulation.

An interesting characteristic of the tool is the automatic selection of the most appropriated digital hardware constraints in every VHDL module. This option is carried out by calculating the dynamic range of every *CodeSimulink* box from the model in successive simulations with different hardware variables (as the datawidth).

**Table 1.** *HSM-RI* processing steps

|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <ol style="list-style-type: none"> <li>1. Syntax analysis of the expression (<i>parsing</i>). <ul style="list-style-type: none"> <li>- Evaluation of the syntax of the mathematical expression.</li> <li>- Evaluation of the syntax of every individual function.</li> </ul> </li> <li>2. Signal extraction. <ul style="list-style-type: none"> <li>- Signals algebraic minimization.</li> <li>- Deletion of unnecessary signals.</li> </ul> </li> <li>3. Expansion of the mathematical expression.</li> <li>4. Reorganization of the computing primitives and signals. <ul style="list-style-type: none"> <li>- Inference of computationally lower cost functions.</li> <li>- Rewriting of the expression with well-known block primitives.</li> </ul> </li> </ol> |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

The table 2.1 schematizes *HSM-RI* processing steps carried out by *HSM-RI*.

As an example of a quantitative measurement of our results, a retina-like image processing system with the following characteristics:

- Parallel-Serial architecture.
- Custom made  $8 \times 8$  bits multipliers.
- Saturation in case of results exceeding the operand bit length target of the next module.
- Three *DoG* filters which generate six ( $5 \times 5$ ) convolution gaussian.
- External RAM access.
- Fixed point arithmetic.
- Final weighted sum of the three gaussian filters.

occupies the 31% (9465 slices) of a Xilinx VirtexE 2000 working up to 20Mhz.

## 2.2 Simulation of the System

Our tool is based on *Matlab* and *Simulink* (via *CodeSimulink*) to co-simulate the system. In fact, the *Simulink* model calculated by *Retiner* can be used to simulate the system. Three different simulation levels are allowed: The top level simulation is carried out by *Retiner*, where a full precision simulation of the whole system can be performed. Later, via the auto-calculated new model in *Simulink*, two new simulations are possible, a full precision of the adapted model which shouldn't differ from *Matlab* one and a simulation based on the hardware which will be inferred. The hardware simulation is performed using the *CodeSimulink* libraries.

## 3 Conclusions

A tool for synthesis, co-simulation and co-emulation of hybrid software/hardware general purpose vision systems has been presented. We have focused on the de-

velopment of bio-inspired vision systems. The tool is capable of producing a synthesizable VHDL model by interfacing with *CodeSimulink* libraries which is ready to be used with a *FPGA* or other integrated circuit technology. We believe that the top-down designing methodology is very useful considering the amount of different sciences which have to do in the development of the final system. By the meaning, this methodology dramatically reduces the design time and brings a general sight of the evolution of the system (specifications, proposed model, high-level simulation, hardware constraints, low-level simulation and synthesis).

*Acknowledgments.* This work has been supported by the EU under the european project *CORTIVIS* (Cortical Visual Neuroprosthesis for the Blind) QLK6-CT-2001-00279, by the National Spanish Grant (ref. DPI 2004-07032) and by the University of Granada (Plan Propio de Investigación).

## References

1. <http://cortivis.umh.es>: European project CORTIVIS main web page.
2. F.J. Pelayo, S. Romero, C. Morillas, A. Martínez, E. Ros, E. Fernández. Translating Image Sequences into Spike Patterns for Cortical Neuro-Stimulation, *Neurocomputing*, Volume 58-60 (June 2004), special issue on Computational Neuroscience: Trends in Research 2004, pp. 885-892, ISSN: 0925-2312.
3. Altera DSP Builder – Reference manual.: Altera Corporation, July 2003.
4. <http://www.xilinx.com/products/software/sysgen/features.htm>.: System Generator for DSP main page.
5. A. Martínez, F.J. Pelayo, C. Morillas, S. Romero, R. Carrillo, B. Pino. Generador automático de sistemas bioinspirados de visión en hardware reconfigurable. Proceedings of IV Jornadas de Computación Reconfigurable y Aplicaciones (JCRA'2004), ISBN: 84-688-7667-4, pp. 597-603 Barcelona, Spain. 2004.
6. F. J. Pelayo, C. Morillas, A. Martínez, S. Romero, B. Pino and E. Ros.: A reconfigurable machine to model the first stages of the human visual pathway. Proceedings of the III Jornadas sobre computación reconfigurable y aplicaciones (JCRA'2003). pp. 173-182. ISBN: 84-60099-28-8. Madrid, September 10-12, 2003
7. L.Lavagno, B. Pino, L. M. Reyneri, A. Serra.: A Simulink-based Approach to System Level Design and Architecture Selection. Proc. of the 26th EUROMICRO Conference. Vol. I, pp. 76-83 ISBN: 0-7695-0780-8. Maastricht, the Netherlands, September 5-7, 2000.
8. L.M. Reyneri, Implementation Issues of Neuro-Fuzzy Hardware: Going Towards HW/SW Codesign, *IEEE Transactions on Neural Networks*, January 2003.
9. E. Bellei et al., Simulink-based Codesign and Cosimulation of a Common Rail Injector Test Bench, *Journal of Circuits, Systems and Computers*, April 2003.
10. L.M. Reyneri, M. Chiaberge, L. Lavagno, B. Pino, A Simulink-based HW/SW Codesign of Embedded Neuro-Fuzzy Systems, *International Journal of Neural Systems*, June 2000.

# Two Hardware Implementations of the Exhaustive Synthetic AER Generation Method

F. Gomez-Rodriguez, R. Paz , L. Miro, A. Linares-Barranco,  
G. Jimenez, and A. Civit

Arquitectura y Tecnología de Computadores, Universidad de Sevilla,  
Av. Reina Mercedes s/n, 41012-Sevilla, Spain  
gomezroz@atc.us.es

**Abstract.** Address-Event-Representation (AER) is a communications protocol for transferring images between chips, originally developed for bio-inspired image processing systems. In [6], [5] various software methods for synthetic AER generation were presented. But in neuro-inspired research field, hardware methods are needed to generate AER from laptop computers. In this paper two real time implementations of the exhaustive method, proposed in [6], [5], are presented. These implementations can transmit, through AER bus, images stored in a computer using USB-AER board developed by our RTCAR group for the CAVIAR EU project.

## 1 Introduction

Sivilotti in 1991 proposed a new communications protocol [1] to interconnect neuro-inspired system, Address-Event-Representation (AER). The idea of AER is to provide an interconnection mechanism to communicate neuron-based chips. One neuron sends information to the next neurons layer sending pulses. The frequency of these pulses is proportional to the stimulus received by the neurons.

In biological systems, neurons are connected by point to point connections. In silicon based systems when we want to connect neurons located in different chips, point to point connections are not feasible (each chip can implement thousands of neurons communicate to a similar number of neurons in other chips). As it is not possible to interconnect chips using the required number connections lines, AER tries to solve this using a time multiplexed high speed digital bus.

An emitter chip, like a retina, is composed by cells that send pulse streams. The frequency of these pulses is proportional to the stimulus these cells are receiving. An arbiter combines all the streams and sends the address of the selected cell through the AER bus. At the receiver side, a decoder has to de-multiplex incoming events and send each event to the corresponding cell. This cell can integrate received pulses in order to reconstruct correct value.

To properly synchronise devices attached to this bus, a simple asynchronous handshake protocol based in two signals is used: a request signal (REQ) driven by the emitter device, and an acknowledge signal (ACK) driven by the receiver.

By connecting several AER devices, we can get a bio-inspired system able to make complex image or audio processing in real time. Many papers about AER have been presented like [2], [3], and [4].

In [5] and [6] some software methods to convert an image stored in a computer memory into an AER stream were proposed. AER system researchers need computer interfaces for testing and debugging purposes. On PCI based interfaces the bus bandwidth allows software to be used to produce individual events. If frame based videos have to be converted this can be done by software methods. This is not the case when using lower bandwidth USB connections, as in this case, it is necessary to transmit frames through the USB bus and perform frame to AER conversion in hardware. In this paper we present two hardware implementations of the Exhaustive method [5] [6] and we evaluate their behaviour.

## 2 Exhaustive Method for Synthetic AER Generation

The exhaustive method for synthetic AER generation has been presented in [5] and [6]. A. Linares-Barranco et al. present a software implementation of this method, this paper present two hardware implementations of the exhaustive method. Before presenting each implementation, let us see how the software exhaustive method is used to transmit an image through AER bus.

The time needed to transmit an image is called frame period. This algorithm divides the period in  $K$  slices of  $N \times M$  positions for an image of  $N \times M$  pixels with a maximum grey level of  $K$ . This method is based on the idea that each possible event (with maximum of  $N \times M \times K$  per frame period) has assigned one fixed position in the address event sequence. In thus way, for the slice  $k$ , an event of the pixel  $(i,j)$  is sent on the time  $t$  if the following condition is fulfilled:

$$(k \cdot P_{i,j}) \bmod K + P_{i,j} \geq K \quad \text{and} \quad N \cdot M \cdot (k - 1) + (i - 1) \cdot M + j = t \tag{1}$$

where  $P_{i,j}$  is the intensity value of the pixel  $(i,j)$ .

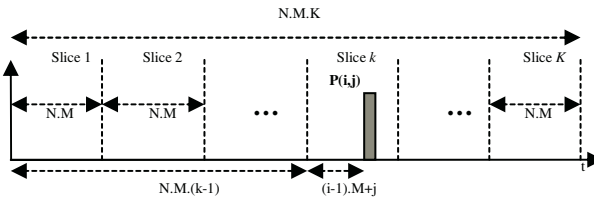


Fig. 1. The event distribution made by the Exhaustive method

The Exhaustive method tries to distribute the events of each pixel into the  $K$  slices at equal distances.

### 3 “Modulus” Exhaustive Method Implementation

The “modulus” implementation tries to emit each event as near as possible to the correct place. The idea of this implementation consists in dividing the period in slices; in each slice all the image is scanned. To decide if a pixel emits an event the following expression is used:

$$slice * p_i \bmod 256 < p_i \tag{2}$$

where:

- slice*: is a counter which provides the slice number
- $P_i$ : is the pixel value.

This formulation is very similar to formulation presented in [7]. The implementation tries to distribute the events in the best way possible. But the division of the period is sufficient to put all the events in their correct place. The exhaustive method shares the events, minimizing the errors.

Let us see a simple example of the exhaustive method approach. Consider an image with 8 grey levels, Table 1 shows how the method decides when a pixel has to emit an event. Columns represent grey levels (0 to 7) and rows represent slice; each cell is the value of  $window * p_i \bmod 8$ , in dark is marked the slices where each event has to emit depending on its value.

Fig. 2 shows a diagram of the system,

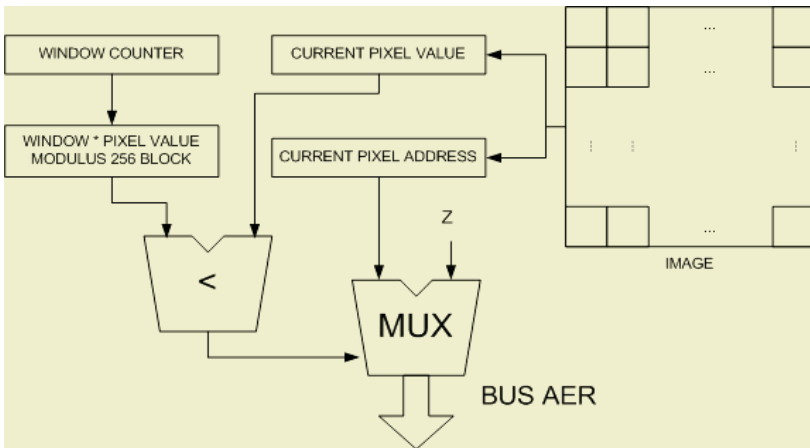


Fig. 2. “Modulus” system diagram

The size of the slice counter and pixel value are given by the number of grey levels, so  $size = \log_2 K$  where K is the number of grey levels, usually  $K = 256$ . The size of the pixel address depends on the size of the image, thus  $size\_address = \log_2 size\_image$ . For this implementation the size of the image is 1024 pixels.

**Table 1.** Events distribution using “modulus” exhaustive methods implementation with 8 grey levels

|        |   | Gray levels |   |   |   |   |   |   |   |
|--------|---|-------------|---|---|---|---|---|---|---|
|        |   | 0           | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Slices | 0 | 0           | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|        | 1 | 0           | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|        | 2 | 0           | 2 | 4 | 6 | 0 | 2 | 4 | 6 |
|        | 3 | 0           | 3 | 6 | 1 | 4 | 7 | 2 | 5 |
|        | 4 | 0           | 4 | 0 | 4 | 0 | 4 | 0 | 4 |
|        | 5 | 0           | 5 | 2 | 7 | 4 | 1 | 6 | 3 |
|        | 6 | 0           | 6 | 4 | 2 | 0 | 6 | 4 | 2 |
|        | 7 | 0           | 7 | 6 | 5 | 4 | 3 | 2 | 1 |

### 4 “Bitwise” Exhaustive Method Implementation

Bitwise implementation tries to get a very simple hardware approach to exhaustive method to minimize processing time and increase event rate. The idea of exhaustive method is to divide a frame time needed to send a whole image into slices. Each slice has a fixed time slot for each pixel. For a specified pixel, each slice has a slot and the number of slices that send an event is proportional to the bright level of the pixel. The distance between two events of a pixel should be constant. If a pixel has a value P, it should emit P times in K-1 slices (K is the number of values allowed, from 0 to K-1, so for 0 level no events are emitted and for K-1, events are sent it all slices).

Bitwise implementation is based on a binary counter driven by a periodic clock signal. This binary counter is divided into two parts. The least significant part is used to address the pixel, and the most significant part is used as “slice counter” to identify a slice in a period. A full run of the whole counter last a period: (K-1)\*N\*M clock cycles).

In order to determine if a pixel has to send an event, the slice number and pixel value have to be compared. If pixel level is greater than the bit-reversed slice counter, then the event has to be sent. This bit reversal is needed to homogenise event distribution. Bit reversal is very popular because it is used in FFT algorithm, and also in many ciphering algorithms to get maximum bit dispersion [8].

**Table 2.** Events distribution using “bitwise” exhaustive methods implementation with 8 grey levels

|        |   | Gray levels |   |   |   |   |   |   |   |
|--------|---|-------------|---|---|---|---|---|---|---|
|        |   | 0           | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Slices | 0 | 0           | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|        | 1 | 4           | 4 | 4 | 4 | 4 | 4 | 4 | 4 |
|        | 2 | 2           | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
|        | 3 | 6           | 6 | 6 | 6 | 6 | 6 | 6 | 6 |
|        | 4 | 1           | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
|        | 5 | 5           | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
|        | 6 | 3           | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
|        | 7 | 7           | 7 | 7 | 7 | 7 | 7 | 7 | 7 |

As it is show in Table 2 for K=8, we have K-1 slices from 0 to K-2. For maximum grey level (value=7) all slices are used. For grey level 1 one by each 7 is used. A pure



binary counter will count one more slice, adding an additional empty slice because no pixel is going to be emitted during slice number. This additional slice decreases performance, increasing period by  $T/(K-1)$ . This is not very important for high K values.

The whole system design is shown in Figure 3:

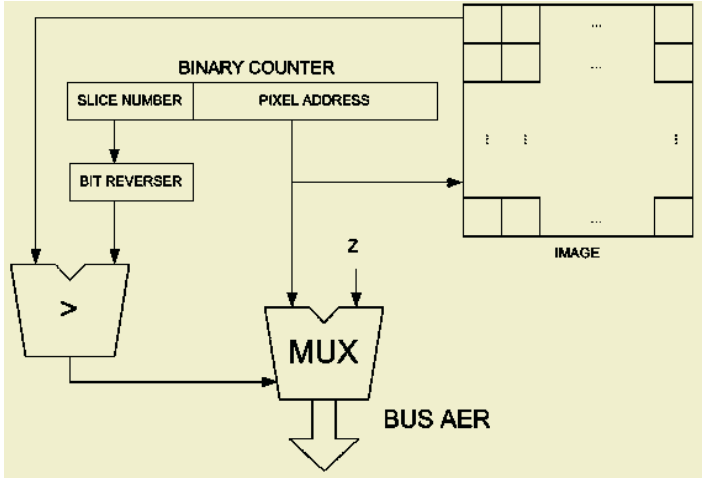


Fig. 3. "Bitwise" system diagram

### 5 Implementations Comparison

Both implementations have been programmed into board developed by RTCAR group at the University of Seville, called USB-AER board. USB-AER board has a FPGA that can be programmed for multiple applications. The implementations presented have been programmed in an USB-AER board in order to test their reliability.

In order to evaluate how both methods distribute the events in [6], an error measurement method is presented. We use this test to evaluate both implementations of the exhaustive method. The mathematical expression of the error measurement is:

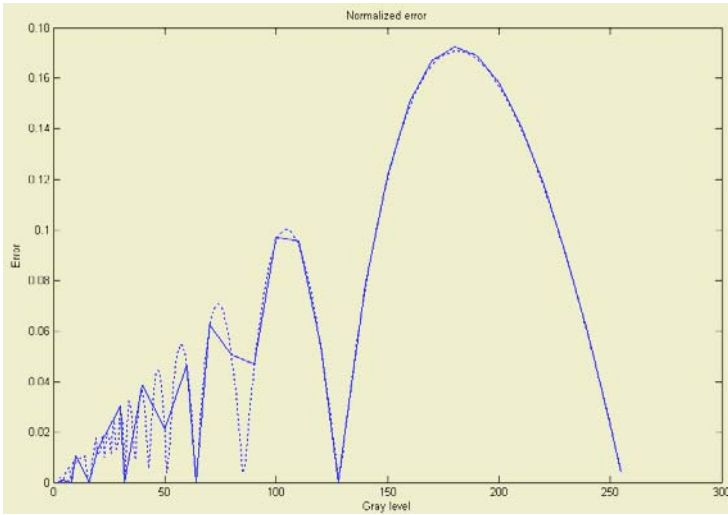
$$Normalized\_error = \frac{e}{m} \tag{3}$$

where:

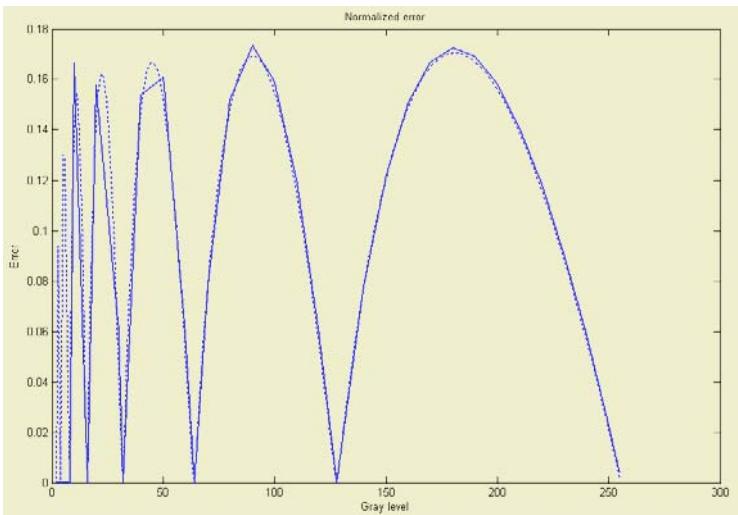
$$e = \frac{\sum_{k=1}^{ns} |D - d_k|}{ns} \tag{4}$$

$$me = 2 \cdot (D - 1) \cdot \left(1 - \frac{1}{P}\right) \text{ with } P \neq 1$$

- ns is the number of samplers.
- D is the theoretical event distance
- d is the real event distance
- p is the pixel value



**Fig. 4.** Normalized error for “modulus” Exhaustive method implementation



**Fig. 5.** Normalized error for "bitwise" Exhaustive method implementation

The test consists in transmitting an image with all pixels set to zero except one. This pixel changes its value, in order to test how implementation follows the theoretical event distribution for each grey level. The images have 256 grey levels. Figures 4 and 5 present the error distribution for each implementation.

Figure 4 and Figure 5 show, in solid line, the normalized error for “modulus” and “bitwise” hardware implementation respectively (taken 10 by 10 and power of 2 grey values); dotted line represents the theoretical implementation normalized error.

Both implementations present zero error when grey level is a power of 2, because in this case the period is divisible by these values. This is expected for all exhaustive method implementations.

The main differences between both implementations are for low pixel values. While “modulus” implementation presents smaller errors than “bitwise” implementation, this last implementation is simpler, because it does not need to compute any multiplication.

## 6 Conclusions

Two hardware implementations of the exhaustive method are presented. These implementations make possible the interconnection between computers and neuro-inspired systems, through an AER-BUS. Thus a computer can be used as an AER stream generator to develop other systems even with a relatively low bandwidth connection to the generator interface. This is typically the case of laptop computers with no PCI slots.

## Acknowledgements

This work was in part supported by EU through CAVIAR project (IST-2001-34124), and Spanish project SAMANTA (TIC-2003-08164-C03-02).

## References

1. M. Sivilotti, Wiring Considerations in analog VLSI Systems with Application to Field-Programmable Networks, Ph.D. Thesis, California Institute of Technology, Pasadena CA, 1991.
2. A. Cohen, R. Douglas, C. Koch, T. Sejnowski, S. Shamma, T. Horiuchi, and G. Indiveri, *Report to the National Science Foundation: Workshop on Neuromorphic Engineering*, Telluride, Colorado, USA, June-July 2001. [www.ini.unizh.ch/telluride]
3. Kwabena A. Boahen. “Communicating Neuronal Ensembles between Neuromorphic Chips”. *Neuromorphic Systems*. Kluwer Academic Publishers, Boston 1998.
4. Misha Mahowald. VLSI Analogs of Neuronal Visual Processing: A Synthesis of Form and Function. Ph.D. Thesis, California Institute of Technology Pasadena, California, 1992.
5. A. Linares-Barranco. Estudio y evaluación de interfaces para la conexión de sistemas neuromórficos mediante Address- Event-Representation. Ph.D. Thesis, University of Seville, Spain, 2003
6. A. Linares-Barranco, R. Senhadji-Navarro, I. García-Vargas, F. Gómez-Rodríguez, G. Jimenez and A. Civit. Synthetic Generation of Address-Event for Real-Time Image Processing. ETFA 2003, Lisbon, September. Proceedings, Vol. 2, pp. 462-467.
7. Alejandro Linares-Barranco, Gabriel Jimenez-Moreno, Antón Civit-Ballcells, and Bernabé Linares-Barranco. On Synthetic AER Generation. ISCAS 2004. Vancouver, Canada, May, 2004.
8. William H. Press, Saul A. Teukolsky, William T. Vetterling, Brian P. Flannery. *Numerical recipes in c: the art of scientific computing* ISBN 0-521-43108-5.1992. Cambridge University Press.

# On the Design of a Parallel Genetic Algorithm Based on a Modified Survival Method for Evolvable Hardware

Dong-Sun Kim<sup>1</sup>, Hyun-Sik Kim<sup>1</sup>, Youn-Sung Lee<sup>1</sup>, and Duck-Jin Chung<sup>2</sup>

<sup>1</sup> DxB-Communication Convergence Research Center,  
Korea Electronics Technology Institute,  
68 Yatop-Dong, Bundang-gu, SeongNam-Si, Gyeonggi-Do 463-816, Korea  
{[dskim](mailto:dskim@keti.re.kr), [hskim](mailto:hskim@keti.re.kr), [yslee](mailto:yslee@keti.re.kr)}@keti.re.kr

<sup>2</sup> Information Technology and Telecommunications, INHA University,  
253 Younghyun-Dong, Nam-Gu, Incheon 402-751, Korea  
[djchung@inha.ac.kr](mailto:djchung@inha.ac.kr)

**Abstract.** In this paper, we propose a Parallel Genetic Algorithm (PGA) based on a modified survival method and discuss its efficient implementation. For parallel computation, we use a hybrid distributed architecture based on the coarse-grain and fine-grain. Moreover, we propose a modified survival-based GA using tournament selection method. To show the validity of a proposed PGA, we evaluate its performance with optimization problems such as DeJong's functions, mathematical function, and set covering problem. In addition, we implement a PGA processor with ALTERA EP2A40672F FPGA device. The experimental results will be shown that proposed PGA remarkably improves the speed of finding optimal solution than single GAP.

## 1 Introduction

The Genetic Algorithm (GA) is one of evolutionary algorithm based on phenomenon of natural society and it has been known as a method of solving large-scaled optimization problems with complex constraints. However, GA needs a long computation time and has a local minima problem due to its iteratively adaptive process and evolution mechanism. By this reason, the parallel and distributed processing of GAs are mainly employed in recent works for reducing the evolution computation time [1,2,3,4]. In addition, the importance of hardware implementation is increased to use in real applications such as large-scaled optimization problems and real-time image processing [3,4,11]. Therefore, we propose a hardware-oriented parallel GA based on a modified survival method and discuss its efficient implementation. To show the validity of the proposed algorithm, we compare the proposed algorithm with the survival-based GA [5] and design a target board with ALTERA EP2A40672F device for estimating its performance with optimization problems such as DeJong's functions, mathematical function, and set covering problem.

## 2 Parallel Genetic Algorithm

The PGA can be categorized to three types: global, coarse-grained and fine-grained [6,7]. The coarse-grained algorithm has the advantage of simple parallel architecture implementation because it can be realized with several same GA processors. But it needs more concentration on designing data path to reduce the hardware complexity. In case of global parallelism, it needs main controller to take charge of scheduling the other processors. Such global method is more efficient to control parallel system but it has the limitation of improving the processing speed.

### 2.1 Proposed Parallel Genetic Algorithm

A proposed PGA is devised for efficient hardware implementation, which reduces the hardware complexity and increases the computational capability. For this purpose, we improve the steady-state model using a modified replacement method and propose a easily expandable architecture based on ring migration. Our selection scheme is that population replacement occur whenever the offspring's fitness is better than worse parent's. It has improved the selection scheme and the preservation of chromosome because the increment of chromosome with high possibility to attain a optimal solution give pressure to converge the solution as fast as possible compared to random selection of survival based GA. In addition, we modify an algorithm by adding migrator module for parallel and distributed processing. The proposed algorithm is mainly divided into two parts as below:

**The Sequential GA Part:** The proposed GA use a modified tournament selection and special survival condition. These enhanced selection algorithm is based on the partial update model that continuously replaces the population by means of update condition occurring, which the offspring's fitness is better than worse parent's. The modified tournament selection only needs two chromosomes from the population in order to generate two parents. On the other hand, the simplified tournament selection [11] needs four chromosomes. A worse chromosome of two parents is compared to the best chromosome between two offspring and the migrated chromosome from another processor. It selects better one between two chromosomes and detailed selection algorithm is described below pseudo-code.

**Migration Part of Hybrid PGA:** The important parameters of migration are migration interval, migration rate, selection method of migrating chromosome and topology type. Because migration method of the proposed PGA is based on the steady-state model [12] with tournament selection, migration interval of our PGA is performed as each generation. In addition, migration rate and migrating chromosome is continuously decided according to the proposed selection scheme. To consider on the complexity of hardware implementation, proposed topology for parallel architecture is basically organized with coarse-grained algorithm and each coarse-grained PGA is connected to other coarse-grained PGA group like fine-grained topology.

**Pseudo-Code of the Proposed PGA:** The Pseudo-code for the proposed parallel GA is following as

Procedure Parallel GA( ) Begin

```

1-- Initialize Population and Best_chrom and Parent_B
 For I = 0 to Np-1 do
 Chrom_data = Random(Ndata); Chrom_fit = Fitness (Chrom_data);
 Chrom = Chrom_data & Chrom_fit; Population(I) = Chrom;
 End For ;
 Best_chrom, Parent_B = all bits is initialized by zero;

2-- Start Genetic Algorithm
 While evolution(Best_chrom) do

3-- Select two chromosomes from population and decide parents
 Chrom_A = Selection(Population(address = RNG(Npa)));
 Chrom_B = Selection(Population(address = RNG(Npb)));
 Parent_A = Max. of Fitness (chrom_A, chrom_B);
 Parent_B = previous Parent_A;

4-- Decide Worse_data from two chromosomes using fitness value
 Worse_adrs = Min. of Fitness (chrom_A_adrs, chrom_B_adrs);
 Worse_fit = Min. of Fitness (chrom_A_fit, chrom_B_fit);
 Worse_data = Worse_adrs & Worse_fit;

5-- Do Crossover and Mutation and evaluate a fitness
 Crossover(A, B) = Choose[1-point, 2-point, uniform];
 Mutation (A, B) = Choose[single-point, multi-point]
 Offspring_data(A, B) = Crossover(Parent_A, Parent_B);
 Offspring_data(A, B) = Mutation (offspring_A_data, offspring_B_data);
 Offspring_fit(A, B) = Fitness (offspring_A_data, offspring_B_data);
 Offspring(A,B) = (offspring_A_data & A_fit, offspring_B_data & B_fit);

6-- Update a population after accept data from other GA
 Migr_chrom_in = Migrator(other GA);
 New_chrom = Max. of Fitness (offspring_A, offspring_B);
 Migr_chrom_out = Max. of Fitness (offspring_A, offspring_B);
 If New_chrom_fit > Worse_fit then
 If New_chrom_fit > Migr_chrom_in_fit then
 Population(Worse_adrs) = New_chrom;
 Else Population(Worse_adrs) = Migr_chrom_in;
 End if;
 Else
 If Worse_fit > Migr_chrom_in_fit then
 Population(Worse_adrs) = Worse_data;

```

```

 Else Population(worse_adrs) = Migr_chrom_in;
 End if;
 End if
 If New_chrom_fit > Best_chrom_fit then Best_chrom = New_chrom;
 End if;

7-- Migrate data to other GA
 If Migration='1' then
 Migrator(Migr_chrom_out);
 End if;
End While;

End procedure end.

```

As shown in above, we modified the random selection of survival-based GA to the tournament selection. In case of the steady-state GA model, it just replaces two previous parents with two offspring that generated by genetic operator. But the proposed PGA model compare a worse chromosome of two parents with the best chromosome among two offspring and the migrated chromosome from another processor. And then, it selects better one between two chromosomes and the probability of premature convergence increases relatively. This method only needs two chromosomes from the population in order to generate two parents. The proposed migration algorithm makes possible to interchange between populations of processors. As this module accepts good chromosomes from other processor, the entire average fitness value of population increased. As a result, the proposed parallel GA has faster convergence speed finding optimal solution than a single GA and it has merit that the probability of approaching optimal solution is high.

## 2.2 Performance Estimation of Proposed PGA

We compare the maximum value of each chromosome's fitness or the fitness mean of population about each generation to estimate the performance of GA and compare Goldberg's Simple GA [5] with the proposed sequential GA using MATLAB on Pentium PC (2.8 GHz). Because two algorithms have different update model of population, we compare two algorithms in view of finding optimal solution. The used fitness functions are DeJong's function [5] and Royal-road function [2] and Set covering problem (SCP)[13].

**SCP:** The SCP is a well-known combinatorial NP-hard optimization problem. The SCP can be defined the problem of covering the rows of zero-one matrix by a subset of the columns at minimal cost within m-row, n-column [13]. The SCP is important to the minimization of logic circuit or the optimization of the resource selection problem. For experiment, we chose fitness function that obtains minimal expression of Boolean function that composed of 19 rows 63 columns with non-unicost. The objective is to find a minimum-sized set of rows whose elements covered all of the 63 columns.

**DeJong’s Function:** The used Dejong’s function [8] is

$$DeJong(1) : f(x_1, x_2, x_3) = \sum_{i=1}^3 x_i^2 \quad x_i \subseteq [-5.12, 5.12] \quad (1)$$

$$DeJong(2) : f(x_1, x_2) = 100(x_1^2 - x_2)^2 + (1 - x_1)^2 \quad x_i \subseteq [-2.048, 2.048] \quad (2)$$

$$DeJong(3) : f(x_1, x_2, x_3, x_4, x_5) = \sum_{i=1}^5 |x_i| \quad x_i \subseteq [-5.12, 5.12] \quad (3)$$

**Mathematical Function Optimization:** The used optimization function is

$$f(x) = 21.5 + x \times \sin(4\pi x) + y \times \sin(20\pi y) \quad (4)$$

where  $0 \leq x \leq 12.1, 4.1 \leq y \leq 5.8$ . It is the problem that finds the maximum value in given range.

Experimental results by comparison of the speed finding optimal solution and fitness mean of each algorithm are shown in Table 1. For experiment, population size, mutation rate, crossover rate and mutation method are 256, 0.005, 0.7, and 1-point respectively. We can show that proposed algorithm normally finds optimal solution faster than that of Simple GA about 35%.

**Table 1.** Performance Comparison of Simple GA with Proposed GA

| Test Function | Simple GA[1]      |         | Proposed GA[2]    |         | [1]:[2] |
|---------------|-------------------|---------|-------------------|---------|---------|
|               | No. of generation | Time(s) | No. of generation | Time(s) |         |
| DeJong (1)    | 37                | 8.772   | 1,070             | 7.023   | 1:0.80  |
| DeJong (2)    | 868               | 152.393 | 11,069            | 82.083  | 1:0.54  |
| DeJong (3)    | 67                | 25.909  | 1,694             | 15.504  | 1:0.60  |
| Royal-Road    | 552               | 122.530 | 12,914            | 90.353  | 1:0.74  |
| SCP           | 18                | 2.194   | 704               | 1.236   | 1:0.56  |

### 3 Parallel Genetic Algorithm Processor

The effective hardware implementation of the PGA is to design an architecture that realizes a compromise between the architectural complexity and the performance of the PGA [4,14]. For this purpose, we divide into concurrent parallel groups and each parallel group is connected with pipeline and control signals.

#### 3.1 Pipelining and Parallelism

For parallel and pipeline architecture, it is important to divide into the concurrent and independent operation group based on the sequence of algorithm



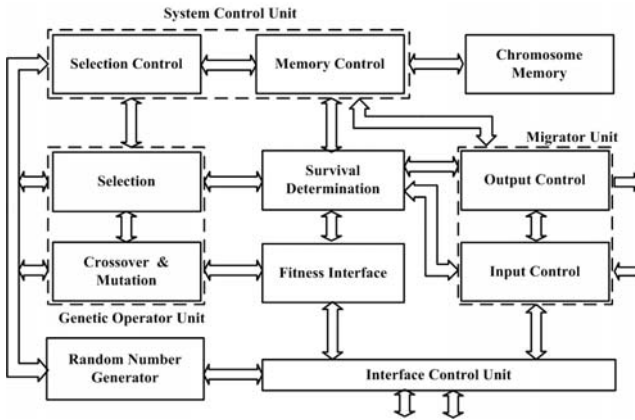


Fig. 1. Block diagram of the proposed parallel GAP

[8, 9]. As shown Figure 2, proposed PGA is divided to four parallel groups and two single modules consist of six stages pipelining. Because each parallel group has different computation time, we design pipelining architecture based on handshaking protocol for continuous update of population. Also, this technique makes it possible to achieve efficient computation because each parallel group can operate independently by clock signal. The used parallelism for the hardware per-

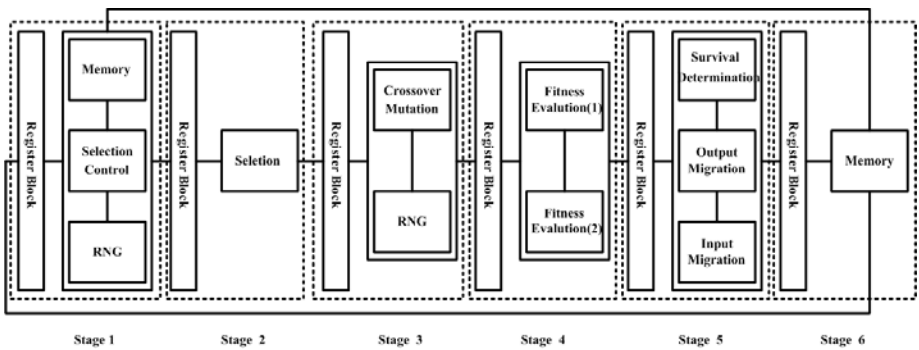


Fig. 2. Pipelining operation in Six stages

formance is hybrid architecture with fine-grained and coarse-grained. In other words, PGA execute concurrently with several sub processors that have sub population and one of sub processors can be export the population with the highest fitness value to another coarse-grained processor groups. In proposed processor, one or more than two modules evaluate the genetic operation of multiple individuals simultaneously because each module is designed as the pipelining based on handshaking protocol. Also, we use multiple fitness function and selective mi-

gration method for performing more efficient internal computation of processor and eliminating the bottleneck of population interchange between processors.

### 3.2 The Modules of Parallel GAP

**Selection Module:** Because two parents are obtained by only two fetched individuals from memory in our PGA, the modified tournament selection method is used for contributing to both a reduction of VLSI size and an increase of speed. In addition, it makes the solution converge into the optimal one as fast as possible because this scheme has higher selection pressure compared to random selection of survival-based GA. As shown in Figure 3, the proposed selection module is consisted of two major blocks which execute concurrently and it works as follow. The left block chooses two individuals randomly from the memory and it obtains better one between two individuals using a comparator, stores the tournament winner in the two registers of current parent [A] and [B]. The register of current parent [A] is replaced with the register of previous parent at the same time. As

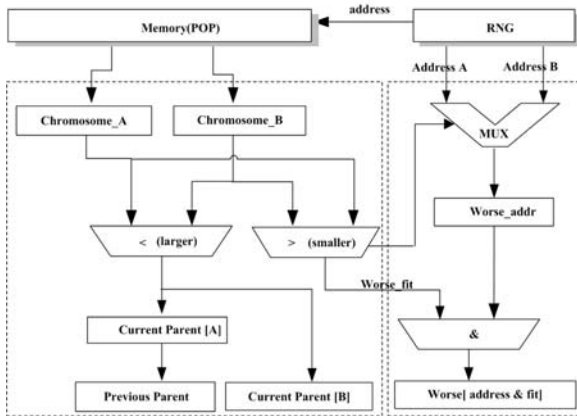


Fig. 3. Hardware structure of Selection Module

a results, two parents is chosen for next step, which is crossover and mutation. The right block computes the address of replacement and the fitness value of worse individual and the address of the worse individual is selected by control signal of multiplexer as shown in Figure 3.

**Crossover and Mutation Module:** The crossover and mutation module receives a selected pair of individuals A and B from the selection module, decides whether to perform crossover and mutation based on random values sent from the random number generator (RNG). This module has several methods of crossover and mutation for the generality in various problems. In case of crossover, it provides the four methods such as 1-point crossover, 2-point crossover, uniform crossover and no crossover. As shown in Figure 4, the crossover block is composed of one comparator, two register (a,b) and i MUX boxes including two

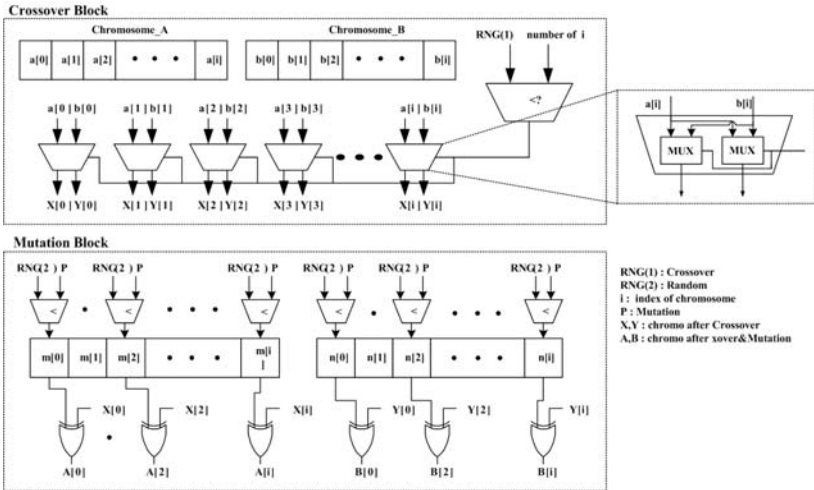


Fig. 4. Hardware structure of Crossover & Mutation Module

multiplexers. The  $i$  is the number of bits in the chromosome. In case of mutation, we use the string method. The mutation block is composed of  $i$  two-input XOR gates,  $i$  comparator and two register ( $m,n$ ) as shown in Figure 4.

**Survival Determination Module:** This module selects the chromosome among two offspring, worse parent, and the migrated chromosome from sub processor. After that, the selected chromosome is stored in memory and replaced finally for upgrade of population. We use three comparators to update chromosome and send migration chromosome to another processor.

**Migrator Module:** This module is a part of interface that takes charge of the migration of chromosome from one processor to another. As result, migrator module improve the speed of finding optimal solution and the average population fitness because it doesn't include the migrated chromosome unconditionally in sub population but compares offspring after accepting the migrated chromosome.

**Communication Topology:** For parallel architecture, communication topology is important to interconnect between sub processors. For efficient implementation of PGA, we use the ring type topology for the parallelism of coarse-grain and it can be expanded to fine-grained topology. Basic unit (BU) is here defined as a basic coarse-grained parallel GA group that is made up of some single GA chips and connected to each other in ring form. One of GA in BU, GAP 1, is export to another BUs for fast evolution and easily expand to fine-grained parallel GAP.

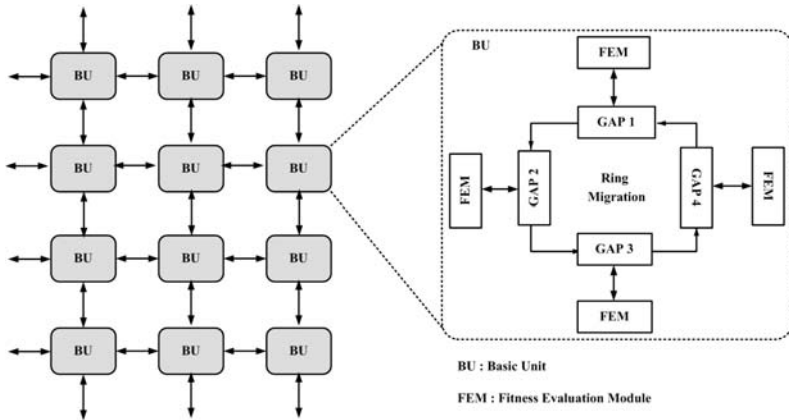


Fig. 5. Communication Type

### 4 Performance Evaluation of Proposed GAP

For experiment, we design a test board with ALTERA EP2A40672F device and proposed one parallel GAP occupies the 76% of 38400 logic elements. We compare with software GA for the performance evaluation of the proposed hardware GAP. Software GA is designed with C and executed on a Pentium PC (2.8 GHz). The used population size, crossover, and mutation probability is 256, 1, and 0.055, respectively. Table 2 shows the comparison result between software GAP and the proposed hardware GAP. The selection module of the proposed hardware GAP performs 100 times faster than that of software. In addition, crossover and mutation module is performed 100 times faster. Table 3 shows

Table 2. Software vs. Hardware (execution time per generation)

| Operation            | Software[us] | Hardware[us] | 1:SW/HW  |
|----------------------|--------------|--------------|----------|
| Selection            | 54.8         | 0.48(10)     | 1:114.2  |
| Crossover & Mutation | 54.8         | 0.48(10)     | 1:114.2  |
| Fitness Evaluation   | 1587.2       | 0.57(12)     | 1:2784.5 |
| 1-Generation         | 1829.0       | 3.6(80)      | 1:508.0  |
| SCP                  | 1235800      | 749.4(21588) | 1:1649.1 |

a comparison of the proposed single GAP (1-processor) and parallel GAP (2-processor). In case of the software simulation of SCP, the single GAP obtained optimal solution after performed 704 generations on average and the parallel GAP just found it after 398 generations. In the hardware simulation, we experimented with same device, same parameters condition for absolute comparison with two processors. The used clock rate, population size, crossover method, mutation rate and chromosome length are 25 MHz, 256, 1-point, 0.055, 19 bits

**Table 3.** Proposed SGAP vs. Proposed PGAP

|                |            | <b>Proposed SGAP</b><br>(1-Processor) [1] | <b>Proposed PGAP</b><br>(2-Processor) [2] |
|----------------|------------|-------------------------------------------|-------------------------------------------|
| Software       |            |                                           |                                           |
| Set-Covering   | Generation | 704                                       | 398                                       |
| Problem        | Time[us]   | 1235800                                   | 1749700                                   |
| Ratio([1]:[2]) |            | 1:0.706                                   |                                           |
| Hardware       |            |                                           |                                           |
| Set-Covering   | Generation | 21588                                     | 10703                                     |
| Problem        | Time[us]   | 749.4                                     | 325.8                                     |
| Ratio([1]:[2]) |            | 1:0.435                                   |                                           |

and the fitness function is the non-unicost SCP. This result also shows that the parallel GAP finds optimal solution faster than that of the single GAP same as software simulation result.

## 5 Conclusion

In this paper, we propose a Parallel Genetic Algorithm (PGA) based a modified survival method and discuss its efficient implementation. Our design is based on the steady-state model with modified tournament selection and special survival condition. Also, we propose a hybrid distributed architecture based on the coarse-grain and fine-grain to achieve a highly efficient parallel computation. The proposed hardware can be 500 times faster than software emulation on a 2.8 GHz PC. Also, the proposed parallel GAP (2-processor) can be increased the speed of finding optimal solution by about 40% more than the single GAP. Since it has shorter computation time, the proposed parallel GAP would be applied as a tool to get the optimal solution in various problems such a real time processing, the speech recognition and computer vision.

## References

1. Scott S.D., Samal A., Seth S.: HGA: A hardware-based genetic algorithm. Proceedings of ACM/SIMDA 3rd International Symposium on FPGA. (1995) 53–59
2. Cantu-Paz E.: A Survey of Parallel Genetic Algorithms. *iiiGAL R.* 97003. (1997)
3. Salami M.: Multiple genetic algorithm processor for hardware optimization. Proceedings of First International Conference. ICES'96 Evolvable System from Biology to Hardware. (1996) 249–259
4. Turton B.C.H., Arslan T., Horrocks D.H.: A Hardware Architecture for a Parallel Genetic Algorithm for Image Registration. Proceedings of IEE Colloquium on Genetic Algorithms. (1994) 11/1–11/6
5. Goldberg D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. (1989)

6. Pelikan M., Parthasarathy P., Ramraj A.: Fine-grained Parallel Genetic Algorithms in Charm++. *ACM Crossroad Magazine: Parallel Computing*. (2002)
7. Pereira C., Lapa C.: Coarse-grained parallel genetic algorithm applied to a nuclear reactor core design optimization problem. *Annals of Nuclear Energy*. (2003) 555–565
8. Michalewicz, Z.: *Genetic Algorithms + Data Structures = Evolution Programs*. 3rd edn. Springer-Verlag, Berlin Heidelberg New York (1996)
9. Berntsson J., Tang M.: A convergence model for asynchronous parallel genetic algorithms. *The 2003 Congress on Evolutionary Computation*. Vol. 4. (2003) 2627 – 2634
10. Gagne C., Parizeau M., Dubreuil M.: The Master-Slave Architecture for Evolutionary Computations Revisited. *Proceedings of the Genetic and Evolutionary Computation Conference*. (2003) 1578–1579
11. Yoshida N., Yasuaka T., Moriki T.: Parallel and Distributed Processing in VLSI Implementation of Genetic Algorithms. *Proceedings of the Third International ICSC Symposia on Intelligent Industrial Automation, On Soft Computnig*. (1999) 450–454
12. Konfrst Z.: *Parallel Genetic Algorithms: Advances, Computing Trends, Applications and Perspectives*. *Proceedings of the 18th International Parallel and Distributed Processing Symposium*. (2004) 26–34
13. Sandip S.: Minimal cost Set Covering using probabilistic methods. *Proceedings of ACM/SIGAPP Symposium on Applied Computing*. (1993) 157–164
14. Turton B.C.H., Arslan T.: A Parallel Gentic VLSI Architecture for a Combinatorial Real-Time Application-Disc Scheduling. *First International conference on Genetic Algorithms in Engineering Systems*. (1995) 493–498

# A Novel Approach for the Implementation of Large Scale Spiking Neural Networks on FPGA Hardware

B. Glackin, T.M. McGinnity, L.P. Maguire, Q.X. Wu, and A. Belatreche

Intelligent Systems Engineering Laboratory,  
Magee Campus, University of Ulster,  
Derry, Northern Ireland, BT48 7JL, UK  
Phone: +44-28-7137-5591, Fax: +44-28-7137-5570  
b.glackin@ulster.ac.uk

**Abstract.** This paper presents a strategy for the implementation of large scale spiking neural network topologies on FPGA devices based on the I&F conductance model. Analysis of the logic requirements demonstrate that large scale implementations are not viable if a fully parallel implementation strategy is utilised. Thus the paper presents an alternative approach where a trade off in terms of speed/area is made and time multiplexing of the neuron model implemented on the FPGA is used to generate large network topologies. FPGA implementation results demonstrate a performance increase over a PC based simulation.

**Keywords:** SNN, FPGA, Hardware, I&F.

## 1 Introduction

Recent trends in computational intelligence have indicated a strong tendency towards forming a better understanding of biological systems and the details of neuronal signal processing [1]. Such research is motivated by the desire to form a more comprehensive understanding of information processing in biological networks and to investigate how this understanding could be used to improve traditional information processing techniques [2]. Spiking neurons differ from conventional artificial neural network models as information is transmitted by the mean of spikes rather than by firing rates [3-6]. It is believed that this allows spiking neurons to have richer dynamics as they can exploit the temporal domain to encode or decode data in the form of spike trains [7].

Software simulation of network topologies and connection strategies provides a platform for the investigation of how arrays of these spiking neurons can be used to solve computational tasks. Such simulations face the problem of scalability in that biological systems are inherently parallel in their architecture whereas commercial PCs are based on the sequential Von Neumann serial processing architecture. Thus, it is difficult to assess the efficiency of these models to solve complex problems [8].

When implemented on hardware, neural networks can take full advantage of their inherent parallelism and run orders of magnitude faster than software simulations, becoming thus, adequate for real-time applications. Developing custom ASIC devices

for neural networks however is both time consuming and expensive. These devices are also inflexible in that a modification of the basic neuron model would require a new development cycle to be undertaken. Field Programmable Gate Arrays (FPGAs) are devices that permit the implementation of digital systems, providing an array of logic components that can be configured in a desired way by a configuration bitstream [3]. The device is reconfigurable such that a change to the system is easily achieved and an updated configuration bitstream can be downloaded to the device. Previous work has indicated that these devices provide a suitable platform for the implementation of conventional artificial neural networks [9, 10].

It is the aim of the authors current research to devise a strategy for the implementation of large scale SNNs on FPGA hardware. A primary objective of the research is to investigate how biological systems process information and as a result the system employs biologically plausible neuron models. Section two of this paper provides details on the SNN model used for the hardware implementation while the approach used to enable large scale networks to be implemented is described in section three. Results using this approach to implement a 1D co-ordinate transform application are detailed in section four while conclusions and future work are presented in section five.

## 2 SNN Model Implementation

Considerable research has been undertaken in developing an understanding of the behaviour of a biological neuron however there are less research results available on how large arrays of these interconnected neurons combine to form powerful processing arrays. The Hodgkin-Huxley spiking neuron model [11] is representative of the characteristics of a real biological neuron. The model consists of four coupled nonlinear differential equations which are associated with time consuming software simulations and would incur high “real-estate” costs when a hardware implementation is targeted [12]. Thus hardware implementations demand the application of a simpler neuron model. One such model is the Integrate-and-Fire (I&F) model which has been widely investigated by other researchers [12-14]. The model consists of a first order differential equation where the rate of change of the neuron,  $v$ , is related to the membrane currents. The I&F model used is,

$$c_m \frac{dv(t)}{dt} = g_l(E_l - v(t)) + \sum_j \frac{w^j g_s^j(t)}{A} (E_s^j - v(t)) \tag{1}$$

For a given synapse  $j$ , an action potential (AP) event at the presynaptic neuron at time  $t_{ap}$  triggers a synaptic release event at time  $t_{ap} + t_{delay}^j$ , a discontinuous increase in the synaptic conductance

$$g_s^j(t_{ap} + t_{delay}^j + dt) = g_s^j(t_{ap} + t_{delay}^j) + q_s^j \tag{2}$$



otherwise  $g_s^j(t)$  is governed by

$$\frac{dg_s^j(t)}{dt} = \frac{-1}{\tau_s^j} g_s^j(t) \quad (3)$$

The forward Euler integration scheme with a time step of  $dt=0.125ms$  was used to solve the I&F model equations. Using this method differential equation (1) becomes

$$v(t+dt) = v(t) + \frac{1}{c} ((g_l(E_l - v(t)) + \sum_j \frac{w^j g_s^j(t)}{A} (E_s^j - v(t)))) dt \quad (4)$$

and differential equation (3) becomes,

$$g_s^j(t+dt) = g_s^j(t) + \left(\frac{-1}{\tau_i} g_s^j(t)\right) dt \quad (5)$$

A description of the parameters from the above equations and the values used can be found in Table 1.

**Table 1.** I&F Model Parameters

| Param         | Common              | Pyramidal              | Inhibitory              | Description                  |
|---------------|---------------------|------------------------|-------------------------|------------------------------|
| $c_m$         | 8nF/mm <sup>2</sup> |                        |                         | membrane capacitance         |
| $E_l$         | -70mV               |                        |                         | membrane reversal potential  |
| $v_{th}$      | -59mV               |                        |                         | threshold voltage            |
| $v_{reset}$   | -70mV               |                        |                         | reset voltage                |
| $\tau_{ref}$  | 6.5ms               |                        |                         | refractory period            |
| $t_{delay}^j$ | 0.5ms               |                        |                         | propagation delay            |
| $g_l$         |                     | 1uS                    | 1uS                     | membrane leak conductance    |
| A             |                     | 0.03125mm <sup>2</sup> | 0.015625mm <sup>2</sup> | membrane surface area        |
| $E_s^j$       |                     | 0mV                    | -75mV                   | reverse potential of synapse |
| $\tau_s^j$    |                     | 1ms                    | 4ms                     | synapse decay time           |

The hardware implementation employed a fixed point coding scheme. 18 bits were used to represent the membrane voltage (4), 10 bits of which correspond to the fractional component. The synaptic conductance (5), was implemented using 12 bit precision for the fractional component. Multiplicand and divisor parameters were chosen as powers of 2 so that they could be implemented using binary shift thus utilising a smaller amount of logic than what would be required by a full multiplier or divider.

A biologically plausible Spike Timing Dependant Plasticity (STDP) algorithm, based on the Song and Abbott approach, was implemented in hardware to train the network[15-16]. Each synapse in an SNN is characterized by a peak conductance  $q$  (the peak value of the synaptic conductance following a single pre-synaptic action

potential) that is constrained to lie between 0 and a maximum value  $q_{max}$ . Every pair of pre and post-synaptic spikes can potentially modify the value of  $q$ , and the changes due to each spike pair are continually summed to determine how  $q$  changes over time. The simplifying assumption is that the modifications produced by individual spike pairs combine linearly. A pre-synaptic spike occurring at time  $t_{pre}$  and a post-synaptic spike at time  $t_{post}$  modify the corresponding synaptic conductance by  $q \rightarrow q + q_{max} F(\Delta t)$ , where  $\Delta t = t_{pre} + t_{post}$  and  $F(\Delta t)$  is defined by

$$F(\Delta t) = \begin{cases} A_+ \exp(\Delta t / \tau_+), & \text{if } \Delta t < 0 \\ -A_- \exp(\Delta t / \tau_-), & \text{if } \Delta t \geq 0 \end{cases} \quad (6)$$

The time constants  $\tau_+$  and  $\tau_-$  determine the ranges of pre- to postsynaptic spike intervals over which synaptic strengthening and weakening are significant, and  $A_+$  and  $A_-$  determine the maximum amount of synaptic modification in each case. The modification function for the STDP algorithm is shown in Fig. 1.

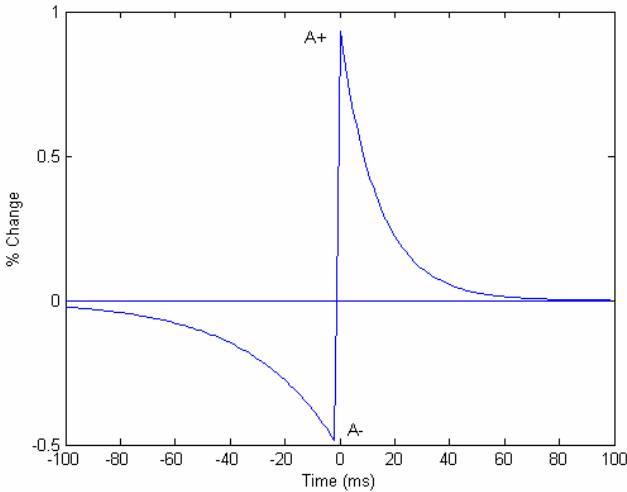


Fig. 1. STDP Modification Function

### 3 Large Scale Implementation

Despite the computational simplicity of the I&F model there is still a limit to the size of network that can be implemented on an FPGA. Consider the logic requirements to incorporate each SNN component on an FPGA as shown in Table 2.

Using a fully parallel implementation the number of neurons that can be implemented on an FPGA is generally limited to the number of embedded multipliers provided. Further multipliers can be generated from the logic, however this is area intensive and rapidly consumes the remainder of the resources, leaving little logic to implement STDP synapses. The largest device in the Xilinx Virtex II series of FPGAs is the XC2V8000 device which consists of 46,592 slices and 168 embedded

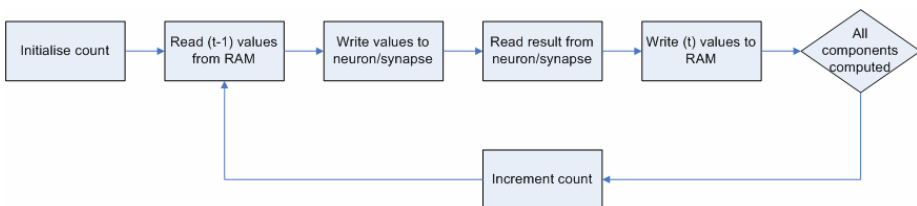
**Table 2.** SNN Logic Requirements

| SNN Component | Slices | Embedded Multipliers |
|---------------|--------|----------------------|
| Synapse       | 33     | 0                    |
| STDP Synapse  | 106    | 0                    |
| Neuron        | 63     | 1                    |

multipliers. Therefore using an STDP synapse to neuron connection ratio of 1:1 a total of 168 neurons and 168 STDP synapses could be implemented on a XC2V8000 device (using approximately 60% of the available resources for functional circuitry). Increasing the connection ratio of STDP synapses to neurons to a more realistic ratio of 10:1 enables a network consisting of 41 neurons and 410 synapses to be implemented. Finally using a more biologically plausible ratio of 100:1 indicates that the largest size network that could be implemented would be 4 neurons and 400 STDP synapses.

Utilising this fully parallel implementation with a clock speed of 100 MHz and a Euler integration time step of 0.125ms per clock period, a 1 second real time period of operation could be completed in 0.8ms. This provides a speed up factor of 12500 compared to real time processing. This compares very favourably with a software implementation where the data is processed serially and can take hours to run a several second real time simulation. The speed up factor indicates that there are possibilities for examining speed/area trade offs and determining alternative solutions that will lead to larger network capabilities. By partitioning the design between hardware and software a compromise can be made between the size of network that can be implemented and the computing speed. One approach is to use a controller that can time-multiplex a single neuron or synapse, which reduces the amount of logic required but increases the amount of time to compute. A flow diagram for this process is shown in Fig. 2.

The program running on the processor loops through the number of neurons/synapses to be computed. It reads the past time steps values for the neuron/synapse and sends them to the hardware component as inputs. The result of the hardware computation is read back and then written to RAM such that it can be used for the next time step computation. This approach has been adopted and a system has been developed which utilises the MicroBlaze soft processor core. MicroBlaze is a 32-bit soft processor core featuring a RISC architecture with Harvard-style separate 32-bit instruction and data busses. Synapse and neuron blocks are implemented in the logic along with the processor cores and can be accessed to perform custom SNN instructions for the processor. An overview of the system developed is shown in Fig. 3.



**Fig. 2.** Flow Diagram

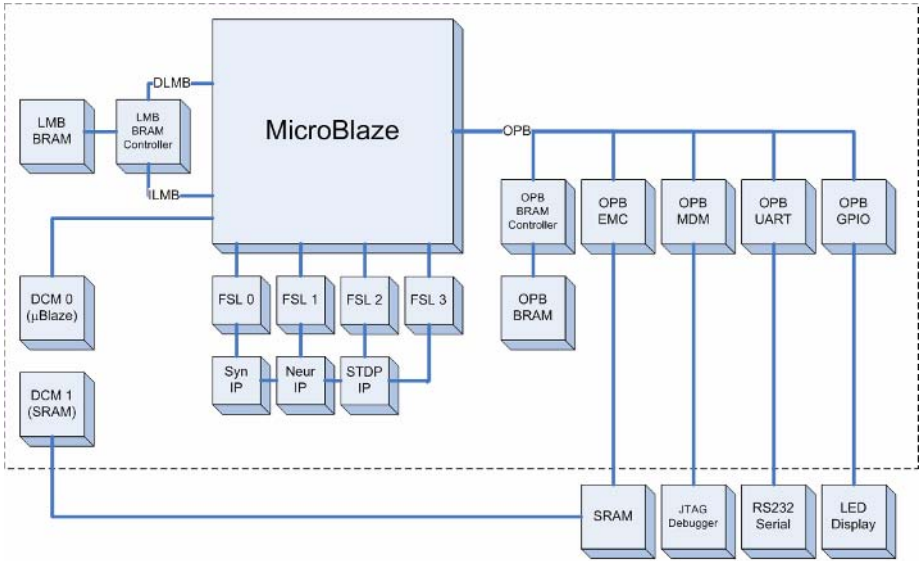


Fig. 3. MicroBlaze Overview

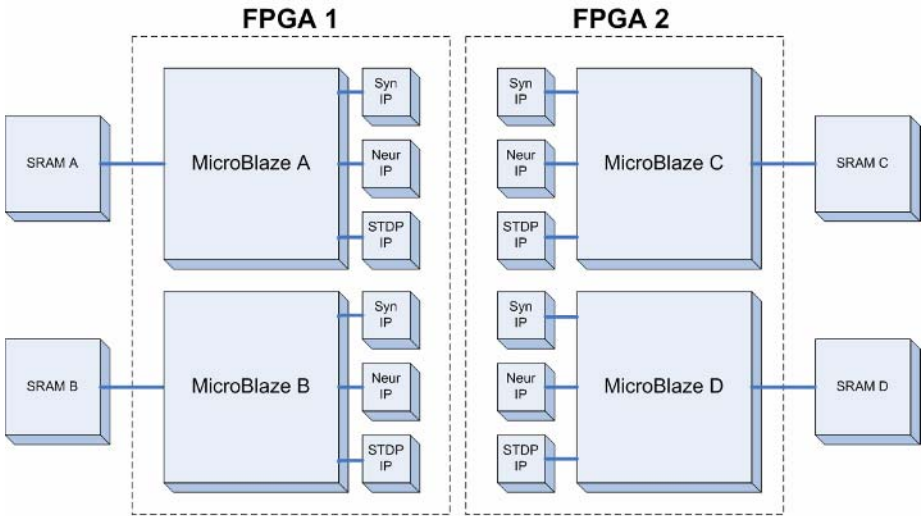


Fig. 4. Multiple Processor System

The main component in the system is the MicroBlaze soft processor core. The program running on this processor is contained in the Local Memory Bus (LMB) Block RAM (BRAM) and is accessed through the LMB BRAM controller. The neuron/synapse/STDP blocks are implemented in the FPGA logic and accesses by the processor using the Fast Simplex Links (FSL) interface which are uni-directional

point-to-point communication channel busses. The neural network data for the network is then held in SRAM which is external to the FPGA device. This memory is accessed over the On-chip Peripheral Bus (OPB) via the OPB External Memory Controller (EMC). As the memory is synchronous a clock-deskew configuration using two Digital Clock Managers (DCM) blocks with a feedback loop ensures that the clock used internal to the FPGA and that for clocking of the SRAM chips is consistent. Additional memory is also made available to the processor in the form of additional BRAM which is also accessed over the OPB bus. Internal status of the network such as the synapse weights or output spike trains can be displayed on a host PC via the serial port which is provided by the OPB User Asynchronous Receive Transmit (UART). Finally the OPB MDM provides an interface to the JTAG debugger allowing the system to be debugged if problems arise. Multiple instances of the soft processor core can be instantiated on a single FPGA thus a degree of parallelism can be maintained in the network while a number of FPGAs can also be designated to operate in parallel. Thus the system can be extended by increasing the number of processors in the system as shown in Fig. 4.

## 4 Co-ordinate Transform

In order to test the system described in the previous section, a spiking network designed to perform one dimensional co-ordinate transformation was implemented. Co-ordinate transformation is used in the SenseMaker system [17], to convert arm angles of a haptic sensor to (x,y) co-ordinates. These co-ordinates are then used to generate a haptic image representation of the search space. A biologically plausible strategy was developed based on the principle of the basis function network proposed by Deneve et al [18] and was further extended by incorporating the STDP algorithm described in section 2 to train the network to perform the function  $y = x - c$ .

An input layer with 100 neurons is fully connected to the output layer with synapses for which weights are determined by STDP. The training layer with 100 neurons is connected to the output layer with fixed weights within a receptive field. The output layer receives spikes from both the input and the training layers during the training stage. Only the input layer is used during the testing stage.

Using this approach a network was trained to perform the function  $y = x - 72^\circ$ . A Gaussian distributed spike train was presented at each input neuron while a similarly distributed spike train was applied at the appropriate training neuron. For example if a spike train centred at neuron 50, i.e.  $0^\circ$ , was applied to the input layer, a spike train centred at neuron 30, i.e.  $-72^\circ$  would be applied at the training layer. Using this approach for the complete network produced the weight distribution matrix as shown in Fig. 6. This figure shows, in a graphical manner, the value of the synaptic weights connecting the input and output layers converged after training has been completed. It can be seen that the weight distribution reflects the function itself, i.e. a shift  $y = x - 72^\circ$ . The figure shows the fixed weight interpretation of the synapse weights. As a 12 bit coding scheme was used, the actual weight can be determined by dividing by  $2^{12}$ , e.g.  $48 \div 2^{12} = 0.01171875$ .

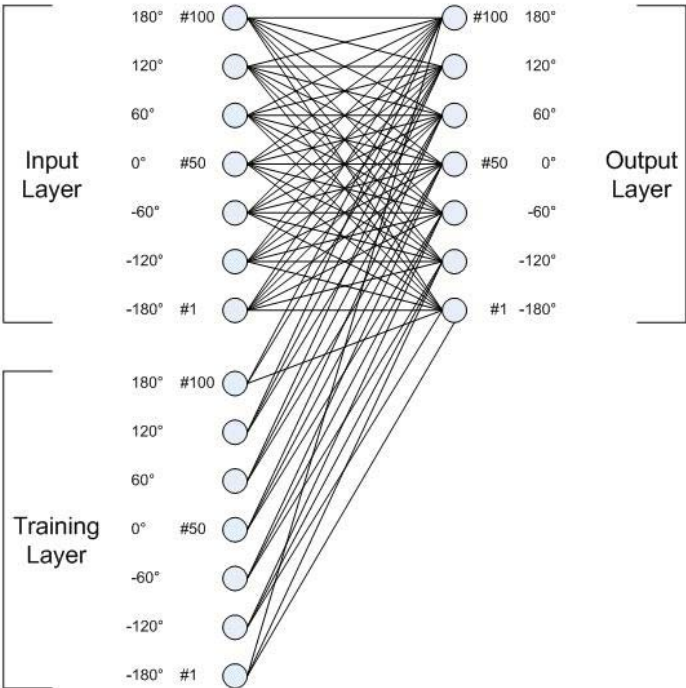


Fig. 5. SNN Trained by STDP for  $y=x-c$

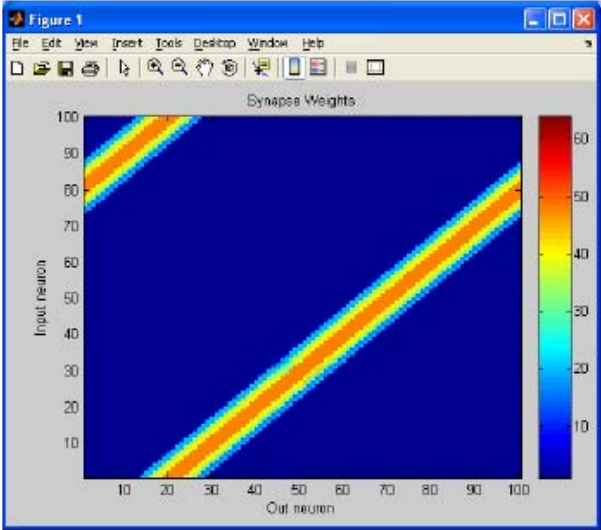
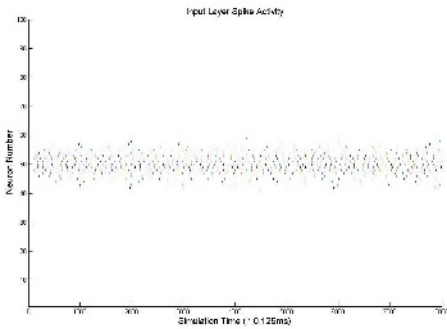


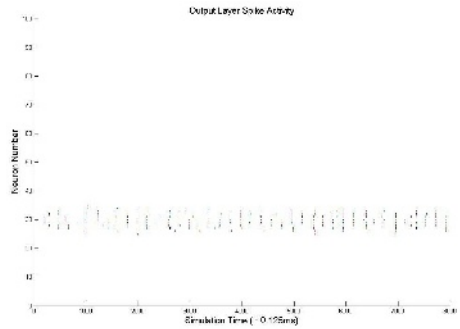
Fig. 6. Weight Distribution Matrix

After training, the network was tested by applying input spike trains and observing the output spike train produced by the network implemented on the FPGA. These tests indicated that the network possessed good generalisation capabilities, with an input spike train centred on neuron  $x$ , an output spike train could be observed centred on neuron  $x - 72^\circ$ . Fig. 7 and Fig. 8 display the input and output spike trains for one such test where an input centred at neuron 50 ( $0^\circ$ ) was applied and an output centred at neuron 30 ( $-72^\circ$ ) can be observed. These results were obtained from a single processor system as depicted in Fig. 3.

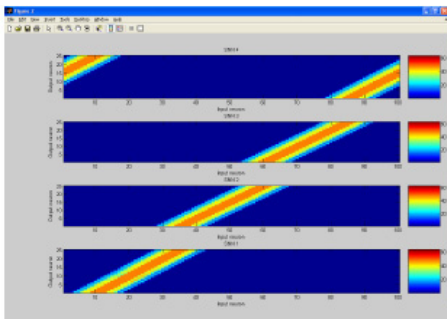
Experiments have also been undertaken to evaluate the performance of the multi-processor system when applied to the co-ordinate transformation application. For this experiment the network structure was divided up over the 4 processors, as depicted in Fig. 4, such that each processor was responsible for 25 of the 100 output neurons. Using this approach the weight matrices for the individual processors were obtained and are shown in Fig. 9. These individual matrices can then be combined to form the single matrix for the overall network depicted in Fig. 10.



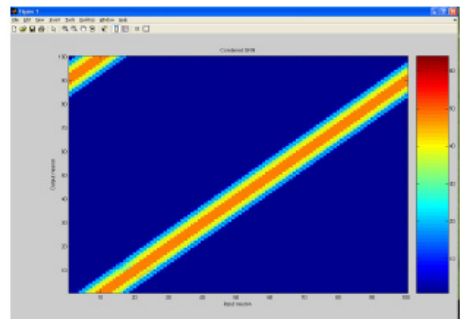
**Fig. 7.** Input Layer Spike Activity



**Fig. 8.** Output Layer Spike Activity



**Fig. 9.** Multi-Processor Weight Matrices



**Fig. 10.** Combined Weight Matrix

To test the maximum capability of this system the largest network possible was also trained on the FPGA system. The network structure consisted of 1400 input

synapses, training synapses and output synapses with each processor responsible for 350 of these output neurons. In total there are 1,964,200 synapses and 4200 neurons for this network. The results obtained are the same as for the smaller versions of the network.

The FPGA implementation for the experiments detailed in this section took place on the BenNuey system developed by Nallatech which provides a multi-FPGA platform. Two of the FPGAs in this platform were utilised for the 4 MicroBlaze system, an XC2V8000 and an XC2V4000 device. The platform can accommodate up to 7 FPGAs, thus this system could be extended further by configuring additional FPGAs in the system with additional processors. The times taken to perform a 1 second real time simulation using the approaches above are shown in Table 3.

**Table 3.** Simulation Performance

| <b>Network Size</b> | <b>PC (Matlab)</b> | <b>1 <math>\mu</math>Blaze</b> | <b>4 <math>\mu</math>Blaze</b> |
|---------------------|--------------------|--------------------------------|--------------------------------|
| 100*100*100         | 2,917.2s           | 115.7s                         | 27.2s                          |
| 1400*1400*1400      | 454,418s           | N/A                            | 4,237s                         |

The multi-processor FPGA approach achieves a performance improvement factor of 107.25 over the Matlab simulation which was implemented on a Pentium 4 2GHz PC.

## 5 Conclusions and Further Development

The results presented in this paper displays that the existing system is a viable platform for the implementation of large scale biologically plausible neural networks on FPGA devices. The current system outperforms a Matlab software simulation by over a factor of 100. Further improvements could be gained by increasing the number of processors in the system thus distributing the processing power and introducing a higher degree of parallelism.

Further enhancements however have also been identified which may further improve performance of the system. It is a goal of this work to benchmark the MicroBlaze soft processor core against the PowerPC core which is integrated on Xilinx Virtex II Pro devices to determine possible gains from moving to this technology.

The work to date has identified that the combination of multiple processor instances with dedicated SNN blocks interfaced to the processor can give performance gains over software simulations. A target of this work is also however to improve the performance of the system such that a large amount of neurons and synapses can be simulated on the FPGA system in real time. From analysing the implementations to date it has been identified that large portions of the network are regularly inactive for large periods of time. With the current approach however for each time step of the simulation each synapse and neuron value is computed regardless. A more efficient approach would be to only compute neuron and synapse when necessary, i.e. when input activity for that component is present. This event based simulation is the next stage of this work which aims to further improve the



computational performance of the hardware implementations. Future work is also investigating the applicability of this platform to replicate the perceptual capabilities of humans which may demand networks of the order of  $10^5$  neurons and  $10^8$  synapses.

## Acknowledgements

The authors acknowledge the financial and technical contribution of the SenseMaker project (IST-2001-34712) which is funded by the EC under the FET life like perception initiative.

## References

- [1] U. Roth, A. Jahnke and H. Klar, "Hardware Requirements for Spike-Processing Neural Networks", From Natural to Artificial Neural Computation (IWANN), edited by J. Mira and F. Sandoval, Springer Verlag Berlin, 1995, Page(s): 720 -727.
- [2] Wolfgang Maass (Editor), Christopher M. Bishop (Editor), Pulsed Neural Networks, MIT Press
- [3] Andrés Upegui, Carlos Andrés Peña-Reyes, and Eduardo Sanchez. "A methodology for evolving spiking neural-network topologies on line using partial dynamic reconfiguration", ICCI - International Conference on Computational Intelligence. Medellin, Colombia. November 2003.
- [4] A. Perez-Uribe. Structure-adaptable digital neural networks. PhD thesis. 1999. EPFL.
- [5] E. Ros, R. Agis, R. R. Carrillo E. M. Ortigosa. Post-synaptic Time-Dependent Conductances in Spiking Neurons: FPGA Implementation of a Flexible Cell Model. Proceedings of IWANN'03: LNCS 2687, pp 145-152, Springer, Berlin, 2003.
- [6] Haykin. Neural Networks, A Comprehensive Foundation. 2 ed, Prentice-Hall, Inc, New Jersey, 1999.
- [7] D. Roggen, S. Hofmann, Y. Thoma and D. Floreano, "Hardware spiking neural network with run-time reconfigurable connectivity in an autonomous robot," NASA/DoD Conference on Evolvable Hardware July 2003.
- [8] A. Delorme, J. Gautrais, R. VanRullen and S.J. Thorpe, "SpikeNET: A simulator for modeling large networks of integrate and fire neurons." Neurocomputing, 26- 27, 989-996 1999.
- [9] J.J. Blake, T.M. McGinnity, L.P. Maguire, The Implementation Of Fuzzy Systems, Neural Networks and Fuzzy Neural Networks Using FPGAs, Information Sciences, Vol. 112, No. 1-4, pp. 151-68, 1998.
- [10] B. Glackin , L. P. Maguire, T. M. McGinnity, Intrinsic and extrinsic implementation of a bio-inspired hardware system, Information Sciences, Volume 161, Issues 1-2 , April 2004, Pages 1-19
- [11] Hodgkin, A. L. and Huxley, A. F. (1952) "A Quantitative Description of Membrane Current and its Application to Conduction and Excitation in Nerve" Journal of Physiology 117: 500-544
- [12] Gerstner W, Kistler W, Spiking Neuron Models: Single Neurons, Populations, Plasticity, Cambridge University Press, 2002
- [13] Koch C, Biophysics of Computation: Information Processing in Single Neurons, Oxford University Press, 1999

- [14] Dayan P, Abbott LF, Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems, MIT Press, Cambridge, 2001
- [15] Song, S., Miller, K. D., and Abbott, L. F. (2000). Competitive Hebbian learning through spike-timing dependent synaptic plasticity. *Nature Neuroscience*, 3:919-926.
- [16] Song, S. and Abbott, L.F. (2001) Column and Map Development and Cortical Re-Mapping Through Spike-Timing Dependent Plasticity. *Neuron* 32:339-350
- [17] <http://sensemaker.infm.ulst.ac.uk/>
- [18] Deneve S., P. E. Latham and A. Pouget, Efficient computation and cue integration with noisy population codes, *Nature Neuroscience* 4:826-831, 2001.

# A Quaternary CLB Design Using Quantum Device Technology on Silicon for FPGA Neural Network Architectures

P.M. Kelly, T.M. McGinnity, L.P. Maguire, and L.M. McDaid

Intelligent Systems Engineering Laboratory, Faculty of Engineering,  
University of Ulster Derry, Northern Ireland, BT48 7JL, UK

Phone: +44-28-7137-5293, Fax: +44-28-7137-5570

pm.kelly@ulster.ac.uk

**Abstract.** Field Programmable Gate Arrays (FPGAs) are being used as platforms for the digital implementation of intelligent systems. Binary digital systems provide an accurate, robust, stable performance that is free from the drift and manufacturing tolerances associated with analogue systems. However binary systems have a much lower functional density than their analogue counterparts resulting in inefficient use of silicon surface area. A design for a novel Configurable Logic Block (CLB) is presented which retains the robust qualities of digital processing whilst providing increased functional density. The circuit design uses Si/SiGe Inter-band Tunneling Diodes (ITDs) and NMOS/CMOS transistors to create quaternary memory cells in a topology and architecture suited to the implementation of neural networks. The performance of the CLB is simulated in HSPICE and the results are presented.

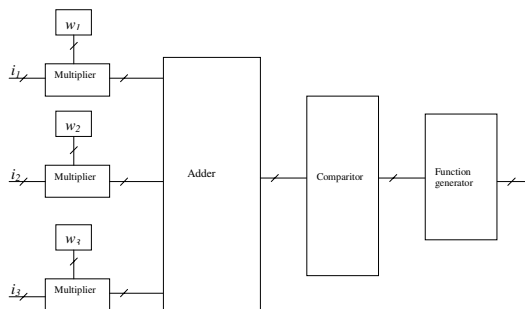
## 1 Introduction

FPGAs are general purpose templates which can be programmed to carry out specific functions. Their re-configurable array based architectures and re-programmable functionality along with their on chip memory capabilities make them suited to the hardware implementation and training of neural networks. The robust, noise tolerant, accurate performance offered by FPGA-based digital systems is highly desirable but comes at a cost. In the context of neural networks, existing binary FPGA systems are inefficient in terms of functional density when compared to analogue implementations. On the other hand analogue systems suffer from the problems of drift and manufacturing tolerances which can have a negative effect on their performance. Combining the desirable qualities offered by analogue and digital systems requires the use of devices and signal systems that will offer increased functional density and robust, accurate performance. Normally, reduced feature sizes of transistors is the means by which increased functional density and capacity in integrated circuits are achieved. However, quantum devices known as resonant tunneling devices (RTDs) and the circuits based on them demonstrate an inherent increased functionality due to the Negative Differential Resistance (NDR) characteristics and the effect of the resulting current-voltage inversion.

This paper proposes a CLB design that uses a combination of RTDs and a Multi-valued Logic (MVL) signal system, both of which have been shown to increase functional density in logic circuits [1,2]. As the CLB is the core functional element in FPGAs it follows that the functional density of neural networks implemented on these devices will be improved. The CLB design uses Si/SiGe inter-band tunneling diodes (ITDs) and NMOS transistors as the fundamental components because they possess characteristics that make them suited to use within a VLSI system such as an FPGA. The circuits that follow were simulated on HSPICE using models based on data obtained from manufactured devices.

## 2 Neural Network Implementation in an FPGA

FPGA based neural networks benefit from the robust and repeatable performance that binary systems offer. The reprogrammable nature of SRAM based FPGAs allows flexibility in circuit architectural configuration. However neural networks place a heavy demand on the resources of the devices. The diagram shown in figure 1 shows one method of implementing a neuron. As this represents a single neuron it illustrates the demands that would be placed on an FPGA realization of many neurons and several layers.

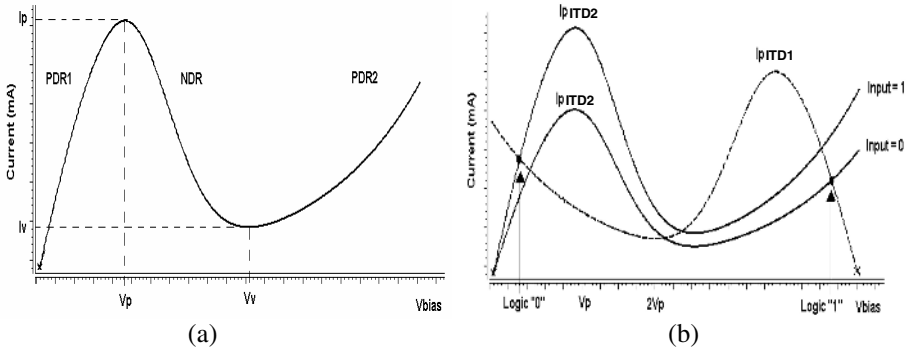


**Fig. 1.** Illustration of how a neuron could be implemented in an FPGA

The diagram in figure 1 represents a single neuron but there will be a need for many of these in the vertical direction and two or more in the horizontal direction to allow input, output and hidden layers within the neural network architecture. The bit width and the number of inputs will be critical in determining how many of neurons can be created on the FPGA. The circuits proposed here offer the possibility of reducing the number of CLBs required to create a given neuron by using an MVL digital system instead of a binary system. Thus an MVL FPGA would retain the advantages offered by binary whilst increasing its functional density.

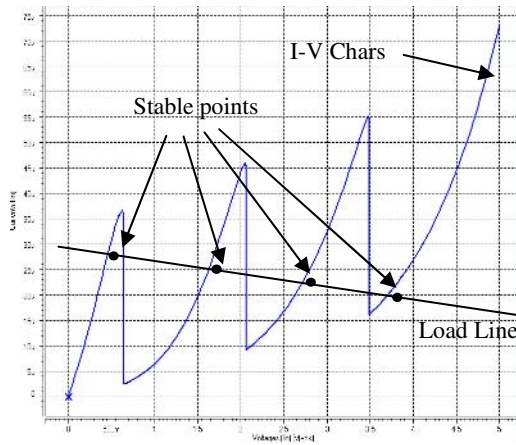
## 3 ITD Characteristics

The fundamental circuit component used in the designs presented in this paper is the ITD. The current voltage characteristics of ITDs display negative differential resistance due to the resonant tunneling effect that determines their behavior.



**Fig. 2.** (a) Typical I-V characteristic curve for an ITD (b) Load line depiction of a RTD series pair operation

A typical I-V characteristic is shown in figure 2(a). This shows that a single device has three main regions: a positive-differential-resistance region (PDR1), a negative-differential-resistance region (NDR) and a second positive-differential-resistance region (PDR2). The I-V curve demonstrates that the device has two positive stable states, one in PDR1 and one in PDR2.



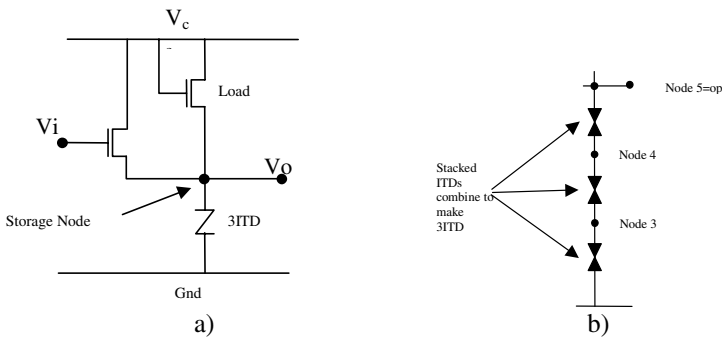
**Fig. 3.** I-V characteristics of a 3ITD stack with resistive load line running through stable points

The simulated devices used in this research have characteristics that display a peak current when the bias voltage is 0.2 volts and a valley current occurs when the bias voltage is increased to 0.55 volts. When the devices are deployed in the TLGs the resulting high and low logic levels can be latched with a holding current of approximately  $30\mu A$  or less. By connecting two RTDs in series and then applying a clocked bias to the series chain, one of the RTDs can latch to PDR2. The diagram in

figure 2(b) uses the load line of ITD1 to illustrate this. The ITD with the lowest peak current will latch to the stable point in its PDR2 region. The other ITD returns to the stable point in its PDR1 region. If this concept is extended so that there are three ITDs in series acting as drivers and transistor connected as a load it is possible to obtain a total of four stable states and this is illustrated in figure 3. By combining the stacked ITDs and the load transistor the characteristics are ‘stretched’ resulting in the four stable states becoming spread between 0.5 and 3.6 volts. The voltage level of each of the states can be assumed to represent a logic level in a quaternary system. These characteristics provide the core functionality for the circuits that are combined to create the MVL CLB design.

### 4 Si/SiGe ITD Based Quaternary Memory Circuit

The diagram shown in figure 4 shows the circuit diagram for the quaternary memory used in the CLB design [1]. The circuit operates off a clocked power supply and during the leading edge of the supply the amount of current flowing in the ITDs determines which device/devices will switch to their PDR2 regions. Any of the stable states shown in figure 2 can be latched to if the appropriate value of current is injected via the input transistor into the storage/output node of the driver ITDs. The value of the load resistor/transistor as a holding element is important as it must not by itself be capable of allowing any of the ITDs to switch, otherwise they would always do so at the leading edge of the clocked power supply. When the injection current is turned off the ITDs ‘hold’ in their various states due to the current provided via the load. The circuit was simulated using Si/SiGe ITDs and NMOS transistors. The performance in terms of stability and fan-in/fan-out capability under load conditions is greatly improved compared to III/V based device implementations.



**Fig. 4.** a) Quaternary memory cell with load transistor and a three ITD stack as driver b) Expanded view of 3ITD to show nodes between devices in the stack

This is due to the isolation provided by the high impedance of the NMOS between input and output gate. Given also that the output node is likely to be connected to the next stage via the gate of an NMOS transistor the amount of current taken from the node is greatly reduced compared to a III/V based HFET resulting in stable circuit operation [3,4].

The performance of the quaternary memory cell was assessed through HSPICE simulations by using the models developed for the ITD and NMOSFET from the measured data of manufactured devices. The input/programming and stored waveforms for the cell are shown in figure 5. The waveforms demonstrate the operation of the cell showing that there are four distinct logic states that can be latched to by input voltages that fall within a certain range. This very important quality means that a certain amount of input signal attenuation will be tolerated by the cell which is a requirement of devices and circuits within VLSI systems. These circuits have a rise time of 1 to 1.5 nS, and a power consumption of 91  $\mu$ W.

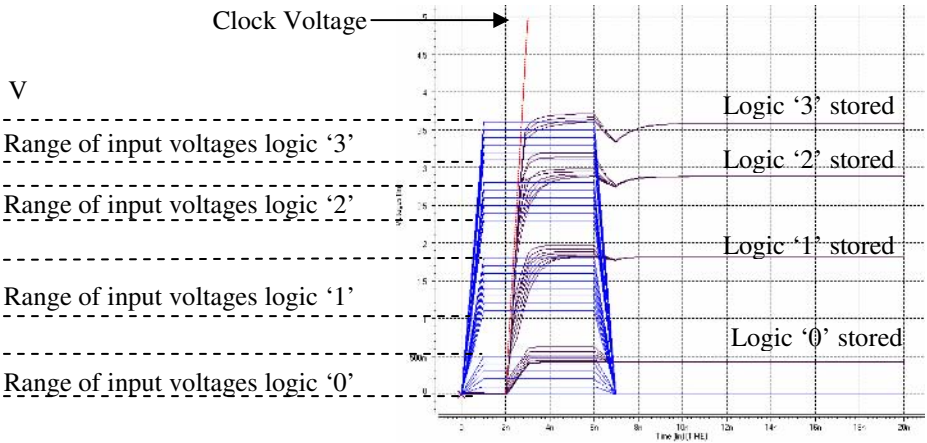


Fig. 5. Input signals and stored logic levels for quaternary cell

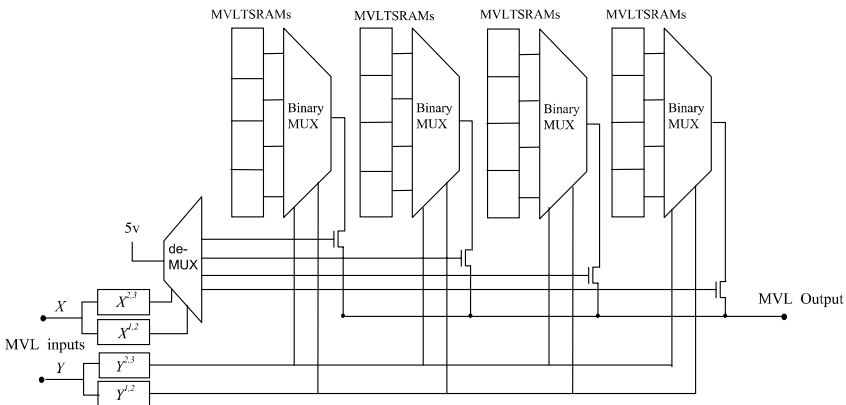


Fig. 6. Quaternary LUT architecture

The quaternary cells were assembled first into groups of four to create a universal gate and then four of these were used in the design of a two input MVL look up table

(LUT). The LUT is the major component of the CLB proposed in this paper as it is the re-programmable function generator at the core of the design. The memory cells were accessed via NMOS transistor based multiplexers which were addressed using binary signals derived from radix conversion circuits based on literal gates with functions  $X^{03}$ ,  $X^{01}$  and  $Y^{03}$ ,  $Y^{01}$ . The inputs to the conversion circuits comprised two MVL lines carrying radix 4 signals and the resulting outputs were 4 lines of radix 2. The circuit architecture shown in Figure 6 was simulated and was found to operate correctly when programmed with randomly chosen MVL functions.

The simulated graphs shown in figure 7 show the waveforms at outputs of the radix conversion circuits and output of the LUT. These confirm that the LUT is operating correctly by outputting the known function stored in the quaternary memory cells in the correct order for an input of two radix 4 variables incrementing from 00 to 33 arithmetic order.

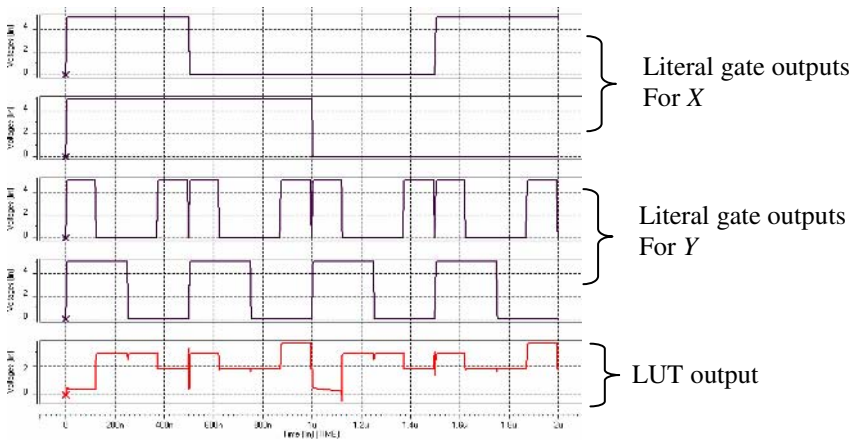


Fig. 7. Output waveforms for the LUT and the radix conversion circuits

### 5 Si/SiGe ITD-Based MVL CLB

The MVL LUT is combined with an MVL latch (comprising a single quaternary memory cell) along with multiplexers to produce the MVL CLB shown in figure 7. The CLB provides the option of registered and unregistered output as well as choice of data source in. Its design has been kept relatively simple for the purposes of this research work and is intended to demonstrate the potential of MVL in FPGAs. The CLBs were simulated in an array based architecture and interconnected to each other to perform functions such as addition and multiplication as these are key mathematical processes associated with the implementation of neural networks.

Both binary and quaternary LUTs are capable of representing 16 possible input variable combinations. In the binary case the number of potential input combinations will be  $x = r^n = 2^4 = 16$  whilst in the quaternary case the number will be  $x = r^n = 4^2 = 16$ . However the functional capacities of the respective LUTs are very



different as in the binary case the number of possible output functions is given by  $N = r^{r^n} = 2^{2^4} = 2^{16}$  and in the quaternary case this is  $N = r^{r^n} = 4^{4^2} = 2^{32}$ . In both cases 16 memory cells will be required to cover all the possible input combinations. Thus whilst it is true to say that a two-input quaternary LUT equates to a four-input binary LUT at their inputs the same is not true for the outputs of the systems because the quaternary LUT effectively stores two bits per cell.

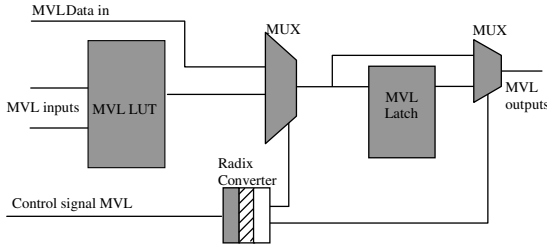


Fig. 8. Basic MVL CLB

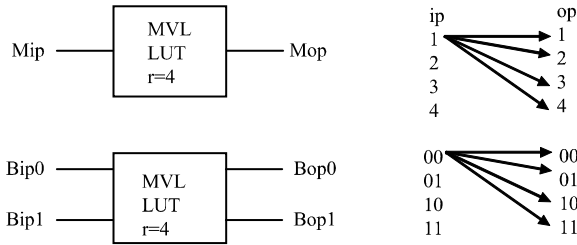


Fig. 9. Single input quaternary and two input binary cell

Consider briefly the single-input quaternary LUT shown in figure 9 this is effectively a universal gate which allows a mapping of any quaternary input state to any quaternary output state. This equates to two binary digits at both the input and output. In that case it's clearly possible to get two completely different single digit functions out for each two-input combination albeit in radix 4 numbers. So for example if MVL state '2' is placed at the input this equates to binary digits  $(x_1, x_2)$  i.e. (1,0) and these two bits can be operated on simultaneously by completely different functions for example  $AND(x_1, x_2) = 0$  and  $OR(x_1, x_2) = 1$ . Thus at the output the results of both operations can be represented by the quaternary number '1' which equates to binary '01'. To carry out the same operations using binary LUTs would require the parallel connection of two two-input LUTs. This implies, that by using MVL memory, reductions can be made not only in the number of lines required but also in the number of LUTs required for certain processing and a given level of functionality. Thus for architectures with highly parallel connection the number of binary CLBs used can be replaced by half the amount of quaternary CLBs whilst retaining the same functionality. This increase in functional density has many beneficial effects such as a

reduction in; device count, interconnection lines and interconnection resources. Alternatively the functional capacity of the FPGA could be doubled whilst using the same number of quaternary CLBs. Because neuronal circuits quickly use up resources when implemented on FPGAs, doubling the capacity of the device will make available twice the resources and hence increase its ability to deploy the neural networks. In terms of the area saved on chip this will be dependant on the physical size of the ITDs. However, if they can be stacked on the drain or source of a FET they will consume no area above that required for the transistor. The ITDs described are currently in micron dimensions as they are prototypes but these would be scaled for manufacture to fit within the drain/source of a transistor. The fabrication difficulties that this might create is outside the scope of this paper.

## 6 Summary

The resource intensive applications of neural networks quickly consume the functionality and connectivity of FPGAs. Thus it is necessary either to create FPGAs specific to the implementation of neural networks or to increase the resource capability of existing designs. A Si/SiGe ITD based MVL CLB design was presented that demonstrated an increase in functional density and a doubling of functional capacity for a quaternary system. All of the circuits presented were simulated and found to exhibit correct functionality and robust circuit operation. The inherent low complexity of ITD based circuits along with the MVL signal system were responsible for the gains in functional density and capacity.

## Acknowledgements

The investigation was funded by the EU QUDOS project IST 2001-32358.

## References

1. Van Der Wagt "JPA Tunnelling-Based SRAM" Proceedings of the IEEE vol. 87, No. 4, April 1999.
2. A. Seabaugh, B. Brar, T. Broekaert, G. Frazier, F. Morris, P. van der wagt, and E. Beam III, "Resonant tunneling Circuit Technology: Has it Arrived?" Gallium Arsenide Integrated Circuit (GaAs IC) Symposium, 1997. Technical Digest 1997, 19th Annual, 15-17 Oct 1997 Page(s): 119 –122.
3. P.M. Kelly, C.J. Thompson, T.M. McGinnity, and L.P. Maguire, "Investigation of a Programmable Threshold Logic Gate Array," IEEE International Conference Electronics Circuits and Systems, proceedings Vol. II, pp 673-67, Sept 2002.
4. W. Prost, U. Auer, F-J. Tegude, C. Pacha, K. F. Goser, G. Janssen and T. van der Roer, "Manufacturability and Robust Design of Nanoelectronic Logic Circuits Based on Resonant Tunnelling Diodes," Int. J. Circ. Theor. Appl. 2000; 28:537-552.

# A Dynamically-Reconfigurable FPGA Platform for Evolving Fuzzy Systems

Grégory Mermoud<sup>1</sup>, Andres Upegui<sup>1</sup>, Carlos-Andres Peña<sup>2</sup>,  
and Eduardo Sanchez<sup>1</sup>

<sup>1</sup> Ecole Polytechnique Fédérale de Lausanne - EPFL,  
Logic Systems Laboratory - LSL,  
Lausanne, Switzerland

{gregory.mermoud, andres.uegui, eduardo.sanchez}@epfl.ch

<sup>2</sup> Novartis Institutes for Biomedical Research - NIBR, Basel, Switzerland  
carlos\_andres.pena@pharma.novartis.com

**Abstract.** In this contribution, we describe a hardware platform for evolving a fuzzy system by using Fuzzy CoCo — a cooperative coevolutionary methodology for fuzzy system design — in order to speed up both evolution and execution. Reconfigurable hardware arises between hardware and software solutions providing a trade-off between flexibility and performance. We present an architecture that exploits the dynamic partial reconfiguration capabilities of recent FPGAs so as to provide adaptation at two different levels: major structural changes and fuzzy parameter tuning.

## 1 Introduction

Nature has long inspired scientists from many disciplines, but it is only very recently that technology is allowing the physical implementation of bio-inspired systems. Nowadays a non negligible part of computer science is devoted to building and developing new bio-inspired systems and most of them yield quite good performance, but often even their creators do not know why and how such systems work since they perform opaque heuristics. Fuzzy systems are an exception among these approaches since they might provide both good results and interpretability of them. Nevertheless, the construction of fuzzy systems is a hard task involving a lot of correlated parameters, which are often constrained to satisfy linguistic criteria. Evolutionary algorithms fit well to such a task [4]. *Fuzzy CoCo* is an evolutionary technique, based on cooperative coevolution, conceived to produce accurate and interpretable fuzzy systems [3].

Three approaches to implement fuzzy systems exist: microprocessor-based (or software), dedicated ASIC and FPGA-based solutions. Maximum flexibility can be reached with a software specification of the full system; however, fuzzy systems are highly parallel and microprocessor-based solutions perform poorly compared to their hardware counterparts. Dedicated ASIC is the best solution for achieving good performance, but such an approach dramatically reduces the

adaptability of the system [1]. Finally, FPGA-based systems provide both higher performance for parallel computation than software solutions and enhanced flexibility compared to ASIC thanks to their dynamic partial reconfiguration (DPR) feature [7]. They thus constitute the best candidate for evolving hardware. Moreover, their run-time reconfiguration features can be used to reduce execution time by hardwiring computationally intensive parts of the algorithm [2].

In this paper we propose a hardware platform for evolving fuzzy systems by using Fuzzy CoCo in order to speed up both evolution and execution while offering equivalent performance. The rest of this section presents an introduction to Fuzzy CoCo and a brief description of dynamic partial reconfiguration on FPGAs. Then, Section 2 describes our hardware platform. In Section 3 we describe the genome used to encode our system. Section 4 presents the experimental setup and results of the simulated platform. Finally, Section 5 contains a discussion about the possibilities and limitations of the platform, gives some directions for further work and concludes.

### 1.1 Fuzzy CoCo

Fuzzy CoCo is a Cooperative Coevolutionary approach to fuzzy modeling, wherein two coevolving species are defined: database (membership functions, MFs hereafter) and rule base. In Fuzzy CoCo, the fuzzy modeling problem is solved by two coevolving cooperative species. Individuals of the first species encode values which define completely all the MFs for all the variables of the system. Individuals of the second species define a set of rules of the form:

**if** ( $v_1$  is  $A_1$ ) **and** ... ( $v_n$  is  $A_n$ ) **then** (*output is C*),

where the term  $A_v$  indicates which linguistic label of the fuzzy variable  $v$  is used by the rule. The two evolutionary algorithms used to control the evolution are instances of a simple genetic algorithm. The genetic algorithms apply fitness-proportionate selection to choose the mating pool and apply an elitist strategy with an elitism rate  $E_r$  to allow a given proportion of the best individuals to survive into the next generation. Standard crossover and mutation operators are applied with probabilities  $P_c$  and  $P_m$ , respectively.

An individual undergoing fitness evaluation establishes cooperation with one or more representatives of the other species, i.e. it is combined with individuals from the other species to construct fuzzy systems. The fitness value assigned to the individual depends on the performance of the fuzzy systems it participated in. Representatives, or *cooperators*, are selected both fitness-proportionally and randomly from the last generation since they have already been assigned a fitness value. In Fuzzy CoCo,  $N_{cf}$  cooperators are selected according to their fitness and  $N_{cr}$  cooperators are selected randomly from the population. For a more detailed exposition of Fuzzy CoCo see [3].

### 1.2 Dynamic Partial Reconfiguration on FPGAs

FPGAs [5] are programmable logic devices that allow, by software reconfiguration, the implementation of digital systems. They provide an array of logic cells

that can be configured to perform a given logic function by means of a configuration bitstream. Some FPGAs allow performing partial reconfiguration, where a reduced bitstream reconfigures only a given subset of internal components. Dynamic Partial Reconfiguration (DPR) is done while the device is active: certain areas of the device can be reconfigured while other areas remain operational and unaffected by the reprogramming [7]. For Xilinx's FPGA families Virtex, Virtex-E, Virtex-II, Virtex-II Pro (applicable also for Spartan-II and Spartan-IIE) there are two documented flows to perform DPR: Module Based and Difference Based.

With the *Difference Based* flow the designer must manually edit low-level changes such as: look-up-table equations, internal RAM contents, I/O standards, multiplexers, flip-flop initialization and reset values. A partial bitstream is generated, containing only the differences between the before and the after designs [8].

The *Module Based* flow allows the designer to split the whole system into modules. For each module, the designer generates a configuration bitstream starting from an HDL description and goes through the entire implementation independently of other modules. A complete initial bitstream must be generated before partial bitstreams are generated for each reconfigurable module. Hardwired Bus Macros must be included. These macros guarantee that each time partial reconfiguration is performed the routing channels between modules remain unchanged, avoiding contentions inside the FPGA and keeping inter-module connections correct.

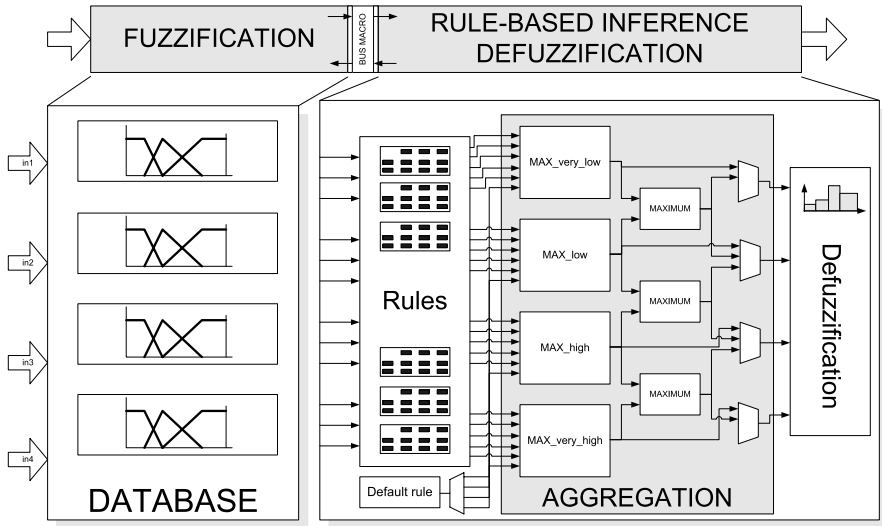
## 2 Our Evolvable FPGA Platform

The proposed platform consists of three parts: a hardware substrate, a computation engine and an adaptation mechanism, as described in [6].

The *hardware substrate* supports the computation engine. It must provide good performance for real-time applications and enough flexibility to allow fuzzy system evolution through the adaptation mechanism. The substrate must allow to test different possible modular layers in a dynamic way. As aforementioned, programmable logic devices such as FPGAs appear as the best solution because they provide high performance thanks to their hardware specificity and a high degree of flexibility given their dynamic partial reconfigurability.

The *computation engine* constitutes the problem solver of the platform. We have chosen fuzzy systems due to their ability to provide not only accurate predictions, but interpretability of the results as well. Other computational techniques such as filters, oscillators, or neural networks are not excluded. Further details are presented in Section 2.2.

The *adaptation mechanism* allows to modify the function described by the computational part. Two types of adaptation are available: major structural modification and parameter tuning. We keep our architecture modular in order to allow structural adaptation as described in detail in [6]. Herein, we concentrate in parameter tuning.



**Fig. 1.** Schematic of the evolvable fuzzy platform. The fuzzy architecture presented at the bottom consists of three layers: (1) fuzzification, (2) rule-based inference and (3) defuzzification. These blocks are described in detail in Section 2.3. The two grey boxes on the top represent the boundaries of the two dynamically reconfigurable layers defined for evolution purposes

## 2.1 The Adaptation Mechanism

In our architecture parameter tuning implies modifying lookup table (LUT) functions. We used the Difference-based reconfiguration flow since only small modifications are performed. This choice provides two advantages: (1) minimization of the reconfiguration bitstream size and hence the reconfiguration time and (2) allowing the possibility of automatically generating the bitstream. To achieve that, we created three hard macros using LUTs for each evolvable part of our platform: the input MFs parameters, the inference rules, the aggregation configuration and the output MFs parameters. By using hard macros location constraints, we can locate each LUT and hence modify it by using Difference-Based reconfiguration as described in [7]. As aforementioned, we use a coevolutionary approach, Fuzzy CoCo, as adaptation strategy. Each Fuzzy CoCo's species is thus implemented as a dynamically reconfigurable layer (see Figure 1), which facilitates separate evolution.

## 2.2 The Fuzzy Computation Engine

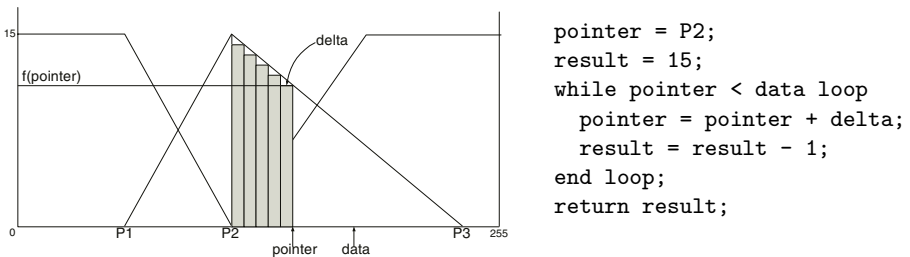
Our fuzzy architecture consists of three layers: fuzzification, rule-based inference and defuzzification. *Fuzzification* transforms crisp input values into membership values. The *rule-based inference*, which computes the firing of each fuzzy rule, provides an activation level for one of the four output MFs. As several rules can propose the same action – i.e. the same output MF – the output fuzzy values are

aggregated by using an aggregation operator (e.g. *maximum*). *Defuzzification* produces a crisp output from the resulting aggregated fuzzy set. We merge inference and defuzzification into a single physical module since the latter is static. Figure 1 shows a top level view of the platform.

### 2.3 Fuzzy Model Setup

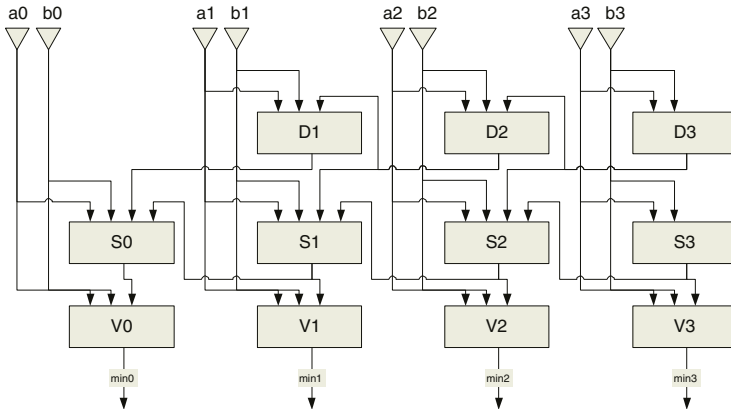
Our implementation has 4 input variables with 3 triangular MFs each. The inference layer contains 20 rules that take up to 4 fuzzy input values from different input variables. The system is easily scalable with respect to the number of inputs or rules. For the sake of interpretability, we add a *default rule*, whose effect is important when the other rules are not very active. In our implementation, the default rule has a fixed activation level encoded by the genome. One of the most commonly used defuzzification methods is the Center of Areas (COA), which is computationally expensive since it includes division. We propose an iterative method and the use of rectangular output MFs for this stage. More details on these issues are provided below.

**Fuzzification.** Taking into account semantic criteria, consecutive MFs of a given input variable are orthogonal [3]. The whole variable is thus defined by means of three parameters, say  $p_1$ ,  $p_2$  and  $p_3$ , defining the function edges as shown in Figure 2. Each parameter represents a key point and is taken from the LUTs. To compute the fuzzy membership value, we propose an iterative approach. The graphic and the pseudocode shown in Figure 2 describe an example of fuzzification, for the second MF of a variable, of an input value between  $p_2$  and  $p_3$ .

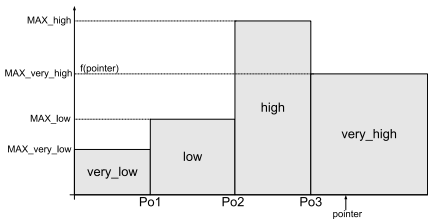


**Fig. 2.** Fuzzification algorithm: graphic representation and pseudocode. In the example, the iterative procedure computes the resulting membership value, *result*, approximating the decreasing linear function with a decreasing counter

**Rules.** For maximum flexibility, we require rules to be able to include fuzzy and and or operators (i.e. respectively *minimum* and *maximum*). As explained in Section 2.1, we have created a hard macro that uses only LUTs to compute any combination of and and or operators on 4 fuzzy values chosen among 16 input values. Figure 3 shows an implementation of the minimum between two 4-bit values  $a$  and  $b$ .



**Fig. 3.** Implementation of a 4-bits minimum operator. Each rectangle represents a LUT taking 4 bits as input. The macro is made up of three layers (D, S and V) and four stages (one per bit). The layer D indicates to the next stage whether a decision can be made. Once a decision is made, further D units transmit this fact. The layer S indicates which value, *a* or *b*, is chosen by the multiplexer V



```

area = 0;
pointer = 0;
while area < totalArea/2 loop
 pointer = pointer + 1;
 area = area + aggr(pointer);
end loop;
return pointer;

```

**Fig. 4.** Rectangular defuzzification MFs and pseudocode for the second step of the defuzzification algorithm. Note that, in the pseudocode, **aggr()** is a function that returns the activation level of the current output linguistic value. Moreover, **totalArea** was previously computed by the first step of the algorithm. The two steps differ only in their end criteria

**Aggregation.** As mentioned before, the activation level of each output MF corresponds to the aggregation of all the rules proposing such MF as output. As shown in Figure 1, the maximum number of rules for each output MF is five plus the default rule. However, we allow the merging of two consecutive output MFs, which has the double effect of increasing the maximum number of rules for the merged MF and decreasing the number of possible output MFs. For example, *very-low* and *low* might be merged to constitute a single MF, called *low*, by forcing them to have the same membership value. In that case, only three output MFs are available instead of four. On the other hand, up to 10 rules plus the default rule might use *low* as consequent.



**Defuzzification.** In our architecture, we consider 4 rectangular output MFs, as those shown in Figure 4. This form, intermediate between singletons and triangular MFs, allows the use of an iterative algorithm to approximately compute the center of areas. Although this method increases latency, it reduces logic and can be efficiently pipelined. The defuzzification process is made up of two steps: the first step computes the total area and the second one, illustrated by the pseudocode in Figure 4, iterates until reaching half the total area.

### 3 Genome Encoding

Figure 5 illustrates our genome encoding. Individual 1 is used to define the database of input MFs while individual 2 is used to describe the rule set and output MFs. The  $i$ -th input variable is defined by three 8-bit parameters:  $P_{i1}$ ,  $P_{i2}$  and  $P_{i3}$  (Section 2.3). For purposes of simplicity, we have pre-assigned five rules to each output linguistic value. The genome must describe the connections between the input MFs and the rules. Encoding the  $k$ -th rule requires five parameters: four 2-bit antecedent values  $A_{kj}$  to choose the applicable MF and one one-bit value  $t_k$  for the type of operator. The *default rule* is encoded by two parameters: one 4-bit value  $dr$  for its activation level and one 2-bit value  $dr_a$  for its consequent. The *aggregation* needs four 2-bit parameters  $M_l$  with  $l = 1, 2, 3, 4$  that indicate the value to be chosen for each output linguistic value among the original and the merged results (Section 2.3). The *output MFs* are completely encoded by three 8-bit values  $Po_1, Po_2$  and  $Po_3$  that represent their boundaries (see Figure 4). The genome of the first individual, encoding 4 input variables, is 96 bits long. The genome of the second individual, encoding 20 active rules, the default rule, 4 aggregated MFs and an output variable, is 218 bits long.



**Fig. 5.** A schematic view of the genome describing the controller

## 4 Platform Simulation and Results

### 4.1 Setup

The experimental setup consists of two parts : (1) a Matlab simulation of the migration of a fuzzy system individual (FSI) from software to hardware implementation, (2) a Matlab simulation of the evolved hardware implementation.

*Migration of an FSI from software to hardware implementation:* by using Fuzzy CoCo evolution, we ran 50 evolutions using the software fuzzy system in order to generate 50 FSIs using at most 10 rules. Then, we compared performance of both the hardware and the software fuzzy system for all these individuals.

*Performance of evolved hardware implementation:* in this case, the performances are compared on the basis of two different individuals, each of them being specially evolved either for the software or the hardware. The evolutionary algorithm is implemented in software in both cases. We ran 48 evolutions with different parameter combinations.

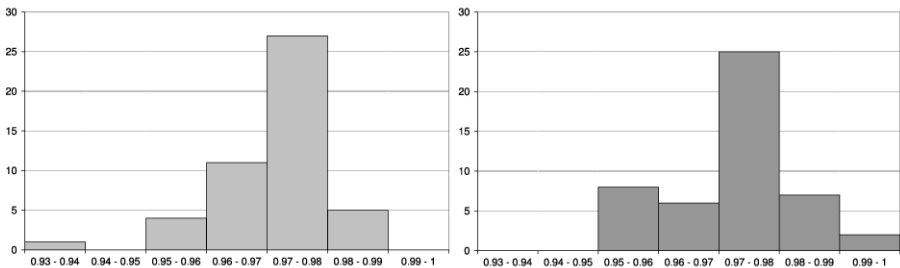
We chose the Iris problem as benchmark since it was already used for evaluating Fuzzy CoCo [3]. Fisher’s Iris data is a well-known classification problem consisting of feature measurements for the speciation of iris flowers. In order to test our system performances, we reused the Matlab Fuzzy CoCo simulation, but another fuzzy system was implemented. This system takes into account all the constraints imposed by the hardware implementation, as described in Section 2.3, except the default rule activation level that is fixed to 2 in the simulation. Therefore, one could consider the real system as being more flexible than the simulated one. We consider the overall percentage of correctly classified cases over the entire database as performance metric.

### 4.2 Results

*Migration of an FSI from Software to Hardware Implementation.* The overall performance loss is about 8.4%, but it should be observed that the standard deviation of these results is high (6.85%). Some individuals perform the same in both implementations while others make the hardware system lose 20% of accuracy, as shown in Table 1.

*Performance of Evolved Hardware Implementation.* The experiment shows that the hardware can reach almost the same accuracy as the software implementation. Table 2 shows the experimental results.

The mean values are almost the same although the software implementation performs slightly better with its best and worst individuals. One may notice that the evolution allows a great reduction of the hardware standard deviation. Figure 6 provides a synthetical view of both implementations performances.



**Fig. 6.** Summary of results of 48 evolutionary runs on both evolved implementations (hardware on the left, software on the right). The histogram depicts the number of systems exhibiting a given mean accuracy value on the complete database

**Table 1.** Comparison between software and hardware implementation performance for the same individual. The best and worst cases are given according to the hardware performance in comparison with the software using the same individual. 50 trials were performed

|         | Software | Hardware | Loss |
|---------|----------|----------|------|
| Mean    | 97.6     | 89.5     | 8.31 |
| Best    | 96.7     | 96.7     | 0.0  |
| Worst   | 97.3     | 75.3     | 22.6 |
| Std dev | 0.9      | 6.3      | 6.6  |

**Table 2.** Comparison between evolved software and hardware implementation performances after 100 generations. The best and worst cases are the overall best and worst performances of both implementations. 48 trials were performed

|         | Software | Hardware | Loss |
|---------|----------|----------|------|
| Mean    | 97.6     | 97.4     | 0.14 |
| Best    | 99.3     | 98.7     | 0.66 |
| Worst   | 95.4     | 94       | 1.4  |
| Std dev | 1        | 0.9      | 1.14 |

## 5 Conclusions and Further Work

In this paper, we have presented a fuzzy hardware platform intended to be evolved by using Fuzzy CoCo. We described the three parts of the platform: the hardware substrate, the computation engine and the adaptation mechanism and how they can be merged. We presented experimental results that show that our platform can reach almost the same performance as a software implementation of Fuzzy CoCo.

Our promising results have incited us to engage in further investigation of this approach. We are currently pursuing three lines of research: (1) refining the implementation in order to allow on-chip evolution, (2) implementing more challenging applications, especially by increasing our platform size, (3) experimenting hybrid systems (e.g. fuzzy neural networks). An *on-chip evolution* would require a processor inside the device being reconfigured, but it would make our platform completely autonomous. Such a capability combined with a larger platform would allow more challenging applications, especially in the robotics field.

## References

1. A. Costa, A. De Gloria, P. Faraboschi, A. Pagni, and G. Rizzotto. Hardware solutions for fuzzy control. *Proceedings of the IEEE*, 83:422–434, 1995.
2. D. Kim. An implementation of fuzzy logic controller on the reconfigurable FPGA system. *IEEE Transactions on Industrial Electronics*, 47(3):703–715, 2000.

3. Carlos Andres Peña Reyes. *Coevolutionary Fuzzy Modeling*, volume 3204. Springer, Berlin, lecture notes in computer science edition, 2004.
4. Y. Shi, R. Eberhart, and Y. Chen. Implementation of evolutionary fuzzy systems. *Fuzzy Systems, IEEE Transactions on*, 7:109–119, 1999.
5. S.-M. Trimberger. *Field-Programmable Gate Array Technology*. Kluwer Academic Publishers, Boston, 1994.
6. Andres Upegui, Carlos Andres Peña Reyes, and Eduardo Sanchez. An FPGA platform for on-line topology exploration of spiking neural networks. *Microprocessors and microsystems*, In press.
7. Xilinx. Two flows for partial reconfiguration: Module based or difference based. Application Note 290, Xilinx, 2004.
8. Xilinx. Virtex series configuration architecture user guide. Application Note 151, Xilinx, 2004.

# FPGA Implementation of Hopfield Networks for Systems Identification\*

Hafida Boumeridja<sup>1</sup>, Miguel Atencia<sup>2</sup>, Gonzalo Joya<sup>1</sup>, and Francisco Sandoval<sup>1</sup>

<sup>1</sup> Departamento de Tecnología Electrónica, Universidad de Málaga

<sup>2</sup> Departamento de Matemática Aplicada, Universidad de Málaga  
Campus de Teatinos, 29071 Málaga, Spain  
matencia@ctima.uma.es

**Abstract.** This contribution presents the hardware implementation of a neural system, which is a variant of a Hopfield network, modified to perform parametric identification of dynamical systems, so that the resulting network possess time-varying weights. The implementation, which is accomplished on FPGA circuits, is carefully designed so that it is able to deal with these dynamic weights, as well as preserve the natural parallelism of neural networks, at a limited cost in terms of occupied area and processing time. The design achieves modularity and flexibility, due to the usage of parametric VHDL to describe the network. The functional simulation and the synthesis show the viability of the design, whose refinement will lead to the development of an embedded adaptive controller for autonomous systems.

## 1 Introduction

In contrast to software simulations, hardware implementation of neural networks preserves parallelism and, besides, it allows for the construction of application specific neural modules. On one hand, these *neural circuits* will outperform software when attempting computationally intensive tasks, such as pattern recognition. On the other hand, they can be successfully embedded in autonomous robots, due to their reduced dimensions, weight and power consumption. Among the architectures proposed for hardware implementation of ANNs, Field Programmable Gate Arrays (FPGAs) provide some advantages, such as rapid prototyping, adaptation, reduced cost and simplicity of design [4]. However, until a few years ago, FPGAs have been unable to reproduce the behaviour of ANNs when the solution of complex, high-dimensional problems was attempted and real-world tasks were involved. The difficulty to deal with a large number of weights that must be stored as floating-point high precision real numbers was a major weakness. Another drawback stemmed from the need of performing multiplications, a basic operation of neural processing. Only recent advances, such

---

\* This work has been partially supported by the Spanish Ministerio de Ciencia y Tecnología (MCYT), Project No. TIN2004-05961. H. Boumeridja acknowledges the support of a fellowship of the Agencia Española de Cooperación Internacional (AECI).

as the appearance of reconfigurable FPGAs [1], have considerably tempered the impact of these limitations. Consequently, the study of implementation issues is currently a worthwhile task that deserves further research [2], in order to determine the viability of neural hardware systems.

The main aim of this work is ascertaining the feasibility of FPGA implementation of a recurrent neural system. The studied ANN is based upon continuous Hopfield networks, which are modified in order to solve the problem of on-line parametric identification of a dynamical system. The problem solution will consist in an estimation of the parameters that appear as coefficients in the Ordinary Differential Equations (ODEs) that model the system. This estimator is part of a wider research project, which is intended to provide an embedded adaptive controller of an autonomous robot, in the presence of uncertain and/or time-varying environmental parameters. In this context, several observations suggest that FPGA is a promising approach: i) concurrency, which is connatural to Hopfield networks, is best reproduced by hardware implementation, when compared to software simulations; ii) in general, the Hopfield networks that are applied to parametric identification comprise a reduced number of neurons and, consequently, the number of weights is not beyond the capabilities of current FPGAs; iii) autonomous systems will often require an identification and control module, which should be built on a small and light circuit, in order to permit its embedding into the autonomous system. Despite these favourable properties, some issues must be studied in order to implement the proposed architecture. An important difference of the *neural estimator*, compared to conventional Hopfield networks, is the fact that its weights vary with time. This variability introduces an additional difficulty with respect to other reported FPGA implementations of ANNs, due not only to the architecture restrictions that are imposed, but also to the eventual implications on the system convergence and stability. The design that is here presented focuses on supporting time-varying weights, while preserving as much as possible the natural parallelism of the network. The construction of a working FPGA-based architecture will allow for determining the practical convergence conditions of the estimator, depending both on the data precision and the permitted processing speed. Then, these requirements can be compared to those of the theoretical stability analysis [3], which was established considering unlimited precision real numbers.

The rest of the paper is organized as follows. Section 2 concisely describes the adopted model of the Hopfield network, which is the Abe formulation, and its application to parametric identification. In Section 3, the architecture of the network is presented, focusing on the constitutive elements of the design and the degree of parallelism achieved. The VHDL description and the signal representation format are presented in Section 4. The synthesis proceeds in Section 5, together with a discussion of the simulation and implementation results. Finally, Section 6 presents some concluding remarks on the main features of the proposed design. The paper ends with a summary of the future directions of research aimed at attaining an identification module that can be embedded in an adaptive controller of an autonomous robotic system.

## 2 Parametric Identification with Hopfield Networks

The Abe formulation [5] of continuous Hopfield networks has been shown to provide promising results in optimization [6]. Its dynamics is defined by the following systems of ODEs (see [7] and references therein):

$$\frac{dp_i}{dt} = \sum_j w_{ij} s_j - b_i; \quad s_i(t) = \tanh\left(\frac{p_i(t)}{\beta}\right) \tag{1}$$

where  $s_i$  is the state of neuron  $i$  and the value of  $\beta$  adjusts the slope of the hyperbolic tangent. In order to formalize the problem of parametric identification, consider a dynamical system whose behaviour is modelled by an ODE, which we assume to be *linear in the parameters* (LIP):

$$\mathbf{y} = \mathbf{A} (\boldsymbol{\theta}_n + \boldsymbol{\theta}) \tag{2}$$

where both the vector  $\mathbf{y}$  and the matrix  $\mathbf{A}$  can depend on the states  $\mathbf{x}$ , the inputs  $\mathbf{u}$  and the state derivatives  $\frac{d\mathbf{x}}{dt}$ . The vector  $\boldsymbol{\theta}$  of parameters is unknown and possibly time-varying, whereas an a-priori estimation  $\boldsymbol{\theta}_n$  is known. Despite the restriction to LIP models, still Equation (2) comprises a wide variety of nonlinear systems, in particular, in the context of robotics, since every rigid mechanical system can be cast into the LIP form [8]. The problem of parametric identification [9] can be defined as determining an estimation  $\hat{\boldsymbol{\theta}}$  that minimizes the prediction error  $\mathbf{e} = \mathbf{y} - \mathbf{A} (\boldsymbol{\theta}_n + \hat{\boldsymbol{\theta}})$ . Hence, identification can be regarded as an optimization problem, whose solution is attained by the network given by Equation (1), as long as the weights are defined as:

$$\mathbf{W} = -\mathbf{A}^\top \mathbf{A} \quad \mathbf{b} = \mathbf{A}^\top \mathbf{A} \boldsymbol{\theta}_n - \mathbf{A}^\top \mathbf{y} \tag{3}$$

The states  $\mathbf{s}$  of the resulting network represent at each instant the obtained estimation  $\hat{\boldsymbol{\theta}}$  (see [3] for further details).

The design process is illustrated with the implementation of a mechanical system, namely an ideal single link manipulator, modelled by the following ODE:

$$y = \frac{d^2 x}{dt^2} = -\frac{g}{l} \sin x - \frac{v}{ml^2} \frac{dx}{dt} + \frac{1}{ml^2} u \tag{4}$$

where  $x$  is the angular position,  $l$  is the manipulator arm length,  $v$  is the friction coefficient,  $m$  is the mass transported by the arm,  $u$  is the control torque produced by a motor and  $g$  is the gravity, assumed constant. In order to cast the model into the LIP form, the formal parameter vector  $\boldsymbol{\theta}$  is defined as:  $(\boldsymbol{\theta}_n + \boldsymbol{\theta})^\top = (-g/l, -v/ml^2, 1/ml^2)$ . Then, the matrix  $\mathbf{A}$  can be defined as  $\mathbf{A} = (\sin x, \frac{dx}{dt}, u)$  so that Equation (2) holds. The corresponding neural estimator comprises three neurons, since the size of the parameter vector is three.

### 3 Digital Architecture of the Hopfield Network

This section describes the architectural design of the continuous Hopfield network given by Equation (1) or, rather, its discretization by the Euler rule, assuming  $\beta = 1$  for simplicity. Then, the model of the individual  $i$ -th neuron is built upon the following components (Figure 1):

- A memory component (RAM) where the values of the synaptic weights are stored. At this stage of the design, the calculation of the weights according to Equation (3) is performed by an external software module.
- A Multiply-Accumulate (MAC) component that computes the weighted sum  $\sum w_{ij} s_j$ . The weights  $w_{ij}$  that correspond to successive values of  $j$  are sequentially presented by appropriately addressing the RAM, whereas the corresponding states  $s_j$  are selected by a multiplexer from the output vector.
- A subtractor component that computes the difference between the result of the MAC and the bias signal  $b_i$ .
- A multiplier component that computes  $\Delta t$  times the result of the subtractor.
- An accumulator component that adds the result of the preceding multiplication to the previous value of  $p_i$ .
- A Look-Up Table (LUT) that implements the activation function  $\tanh$ .

The usage of the MAC element results in a proper trade-off between cost and processing speed. On one hand, it provides a substantial saving of hardware, compared to a completely parallel computation of the weighted sum. On the other hand, a reasonable degree of parallelism is preserved, since the number of neurons is moderate in our application.

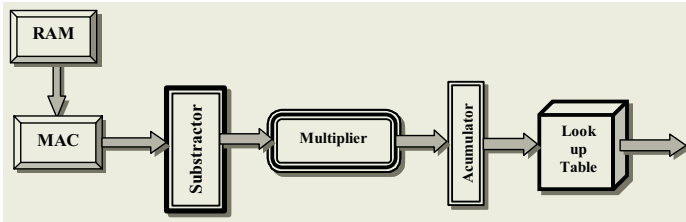


Fig. 1. Hardware model of an individual neuron

The whole network is constructed by joining several (three in our application) neurons together, each of which is modelled as explained above. The global architecture involves some remarkable peculiarities regarding the design of the hardware Hopfield model:

- Only one MAC per neuron is used to compute the product sum.
- Each MAC has its own RAM where the weights are dynamically.
- The neurons of the Hopfield network perform their computations in parallel.
- Whenever a new computation is completed, corresponding to a  $\Delta t$  time advance, the outputs of the LUTs are serially introduced into each MAC.



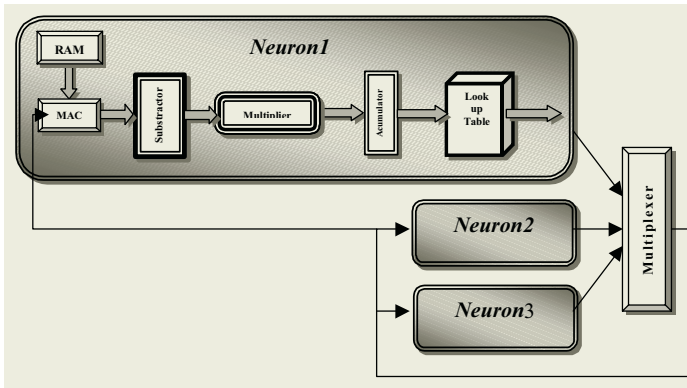


Fig. 2. Global architecture of the Hopfield network with three neurons

The resulting Hopfield architecture is schematically shown in Figure 2. This architecture exhibits a high degree of parallelism and regularity. Besides, the recurrence that is an intrinsic feature of Hopfield networks is implemented in a natural way, contrarily to sequential software simulations.

## 4 VHDL Description of the Hopfield Network

In order to implement the architecture described in the previous section, a VHDL program has been produced. The VHDL syntax is used at register transfer level, since this style is best suited to the synthesis that is later performed. The description of our design is hierarchical, starting from the creation of a component that models an individual neuron. Then, this modular component is replicated so as to construct the architecture of the whole network. As stated above, the component neuron comprises a RAM, a MAC, a subtractor, a multiplier, an accumulator and a LUT. The VHDL description of a neuron is summarized in Figure 3.a). The network consists of a set of neurons, together with a multiplexer that selects the appropriate output for feedback at each cycle. In Figure 3.b) the VHDL description of the network is sketched. In order to achieve design flexibility, the word size (`nb_bits`) and the memories depth (`nb_add`) are defined as generic parameters. The designer can adjust the performance of the network simply by selecting different values.

The signals are coded as signed 2's complement binary integers, as is usual in FPGA-based (i.e. digital) implementation of ANNs. Actually, a virtual decimal point allows for dealing with fixed-point rational numbers. The location of the virtual decimal point can be decided by the designer, since it has no effect on the mathematical hardware. For our application, a word length of 8 bits has been selected, reserving 3 bits for the fractional part, hence the signals have the form `SIIII:FFF`, where `S` is the sign bit, `I` is an integer bit and `F` is a fractional bit.

```

entity neuron is
 -- word size
 generic (nb_bits:integer);
 port (wr1, ...);
end neuron;
architecture neuron_description
 of neuron is
 component RAM
 generic(nb_add:integer;
 nb_bits:integer);
 ...
 component MAC ...
 component Subtractor ...
 component Accumulator ...
 component LUT
 generic(nb_add:integer;
 nb_bits:integer);
 ...
 component Multiplier ...
 ...
 end neuron_description;
end neuron;
a)

entity network is
 generic (nb_bits, k:integer);
 port (clk, ...);
end network;
architecture network_description
 of network is
 component neuron1 ...
 component neuron2 ...
 component neuron3 ...
 component mux_8
 generic(nb_bits:integer;
 k:integer);
 ...
 end network_description;
b)

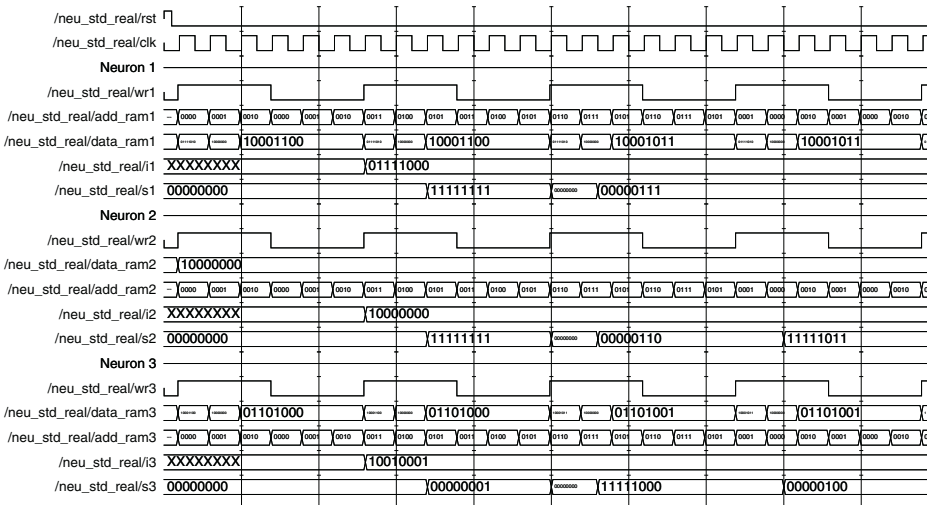
```

**Fig. 3.** Structure of the VHDL description of: a) one neuron; b) a Hopfield network with three neurons

## 5 Synthesis and Simulation

In order to perform the synthesis correctly, the network must be simulated until it meets the functional specifications. With this aim, the *Modelsim SE PLUS 5.7g* simulator of *Modeltech* is used. The success of the functional simulation is illustrated by the resulting chronogram, which is partially shown in Figure 4. Once the functionality has been verified, the VHDL-RTL code is used as a source for synthesizing the proposed architecture, by means of the synthesis tool *XILINX ISE 6.3i*. The Xilinx Virtex family is used as the target technology.

A summary of the synthesis results is reproduced in Table 1. The most usual flaws of FPGA implementations stem from the reduced area and the limited frequency. In our neural network, the percentage of used components is small, except for the number of input/outputs. When considering the eventual scaling of the circuit, it is remarkable that the number of neurons will not grow dramatically in usual applications, due to the way the identification method has been designed. However, the simulation results still deserve further analysis in order to determine whether the discretization errors are too severe, since real values are stored as fixed-point numbers. If the precision has to be incremented, eventually a more advanced FPGA family should be used. Contrarily, frequency does not seem to be a significant issue and the permissible frequency is valid for most applications.



**Fig. 4.** Extract of the functional simulation results of the Hopfield network. Observe the sequential MAC operation in three consecutive cycles

**Table 1.** Synthesis Results

| Selected Device                  | v100bg256-5         |
|----------------------------------|---------------------|
| Number of Slices                 | 446 out of 1200 37% |
| Number of Slice Flip Flops       | 300 out of 2400 12% |
| Number of 4 input LUTs           | 743 out of 2400 30% |
| Number of bonded IOBs            | 146 out of 184 79%  |
| Timing Summary Minimum period    | 20.922ns            |
| Maximum Frequency                | 47.797MHz           |
| Maximum combinational path delay | 14.550ns            |

## 6 Conclusions

In this paper, we present a successful FPGA synthesis for digital implementation of Hopfield networks, modified to perform parametric identification. The main feature of the design is the ability to deal with time-varying weights, whose values are presented to the FPGA at predetermined intervals. The results of the functional simulation support the design, proving that current FPGAs are able to work fast enough to provide on-line solution to complex tasks. Besides, the synthesis summary suggests that the area usage is moderate.

The main direction of future research is the analysis of the results provided by the estimator along an extended time period. The aim is to determine whether the convergence of the estimation is preserved despite both the limited precision of the data representation and the approximated implementation of the transfer function. On the other hand, the usage of more powerful FPGA families will allow for including dedicated multipliers, as well as incrementing the bit length

of data. The validation of these results will pave the way for the construction of an adaptive controller, available for embedding into an autonomous robotic system.

## References

1. Porrman, M., Witkowski, U., Kalte, H., Rückert, U.: Dynamically Reconfigurable Hardware - A New Perspective for Neural Network Implementations. In Glesner, M., Zipf, P., Renovell, M., eds.: *Lecture Notes in Computer Science (FPL 2002)*. Volume 2438., Springer-Verlag (2002) 1048–1057
2. Zhu, J., Sutton, P.: FPGA Implementations of Neural Networks - A Survey of a Decade of Progress. In Cheung, P.Y., Constantinides, G.A., de Sousa, J.T., eds.: *Lecture Notes in Computer Science (FPL 2003)*. Volume 2778., Springer-Verlag (2003) 1062–1066
3. Atencia, M.A., Joya, G., Sandoval, F.: Parametric Identification of Robotic Systems with Stable Time-varying Hopfield Networks. *Neural Computing & Applications* **13** (2004) 270–280
4. Izeboudjen, N., Farah, A., Titri, S., Boumeridja, H.: Digital implementation of artificial neural networks: From VHDL description to FPGA implementation. In Mira, J., Sanchez-Andres, J., eds.: *Lecture Notes in Computer Science (IWANN 1999)*. Volume 1607., Springer-Verlag (1999) 139–148
5. Abe, S.: Theories on the Hopfield Neural Networks. In: *Proc. IEE International Joint Conference on Neural Networks*. Volume I. (1989) 557–564
6. Joya, G., Atencia, M.A., Sandoval, F.: Hopfield Neural Networks for Optimization: Study of the Different Dynamics. *Neurocomputing* **43** (2002) 219–237
7. Atencia, M.A., Joya, G., Sandoval, F.: Dynamical Analysis of Continuous Higher Order Hopfield Networks for Combinatorial Optimization. *Neural Computation* (2005) in press
8. Spong, M.W., Vidyasagar, M.: *Robot dynamics and control*. John Wiley & Sons (1989)
9. Slotine, J.J., Li, W.: *Applied Nonlinear Control*. Prentice Hall (1991)

# An FPGA-Based Adaptive Fuzzy Coprocessor

Antonio Di Stefano and Costantino Giaconia

Dipartimento di Ingegneria Elettrica, Elettronica e delle Telecomunicazioni,  
Università di Palermo, Viale delle Scienze, I-90128, Italy  
distefano@diepa.unipa.it, costantino.giaconia@unipa.it

**Abstract.** The architecture of a general purpose fuzzy logic coprocessor and its implementation on an FPGA based System on Chip is described. Thanks to its ability to support a fast dynamic reconfiguration of all its parameters, it is suitable for implementing adaptive fuzzy logic algorithms, or for the execution of different fuzzy algorithms in a time sharing fashion. The high throughput obtained using a pipelined structure and the efficient data organization allows significant increase of the computational capabilities strongly desired in applications with hard real-time constraints.

## 1 Introduction

Fuzzy logic [1] based algorithms are extensively applied in a wide range of applications involving the treatment of imprecise data or management of complex processes that can hardly be described by a mathematical model. Fuzzy logic is also viewed as a convenient method to embed knowledge of human experts in applications. Some examples of the most advanced fields in which fuzzy logic has gained an increasing success are the visual guidance of robots or autonomous vehicles [2] [4], analysis of sensors data and data fusion techniques [5][6]; braking systems and stability control in automotive environment [7]. Many of such applications require real-time processing capability in order to guarantee satisfactory performances or even to meet basic safety requirements. To achieve this goal a number of fuzzy logic processors have been proposed [8]-[10], as well as dedicated hardware implementations [11]-[15]. The main disadvantage of these solutions is their relatively low flexibility compared to a software implementation, since in many cases fuzzy logic processors lack of general purpose computational capabilities. Moreover, hardware implementations usually employ techniques, such as the use of look-up tables to store the Membership Function (MF) shapes, that prevent a fast modification of the fuzzy parameters. This makes difficult, or even impossible, the execution of fast adaptive fuzzy algorithms, since to change a membership function parameter (shape, position, base width, etc.) it is necessary to compute all its values and to store them in the LUT memory at each iteration [16]. To overcome all these weaknesses a new fuzzy coprocessor was developed, mainly intended to be implemented on FPGA-based Systems on Chip. The coprocessor was optimized for the execution of fast adaptive algorithms, by

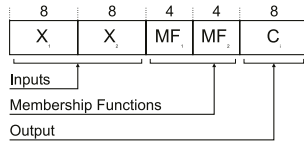


Fig. 1. Rule word data format

minimizing the number of clock cycles required to changes MF and rules parameters. This capability can also be conveniently exploited to execute different fuzzy algorithms in the same coprocessor, using a time shared scheme. In order to achieve good speed performances a pipelined architecture was chosen, and a convenient data organization was employed to minimize the communication overhead with the main processor.

## 2 Coprocessor Architecture

The coprocessor was designed to be easily connected to a generic 32 bit processor through the system bus. Its architecture and operations was chosen in order to maximize its dynamic configurability as well as its generality. The coprocessor works as a general purpose fuzzy logic controller provided with two 8 bit inputs and one 8 bit output. It is possible to specify up to 16 membership function (i.e. fuzzy subsets) for each of the two inputs. There is not a fixed rule matrix as in other fuzzy processors, instead the rules to be evaluated are provided on the fly by the processor along with input data. This approach, even requiring a slightly increased CPU effort, results in an easier adaptation to different algorithms and allows an enhanced ability to support adaptive algorithms employing a dynamically changing set of rules. Each rule and input data are packed in a 32 bit word, and is passed to the coprocessor using a single bus transaction. The word format is shown in figure 1. As it can be seen, rules format is inspired to the usual linguistic form, in fact each word specify either the antecedent part of the rule (the two crisp input and the index of the two membership function to use), either the consequent part, consisting in an 8 bit value specifying the rule output (as a fuzzy singleton).

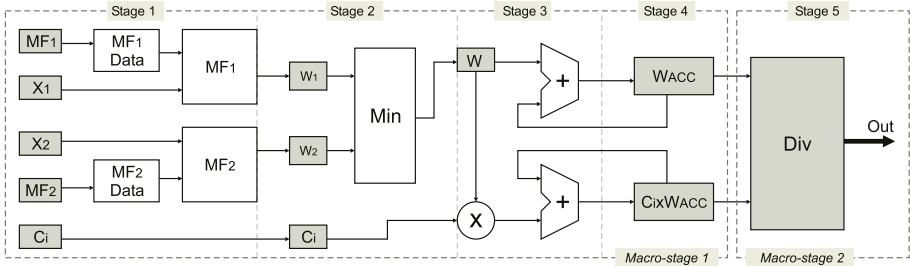
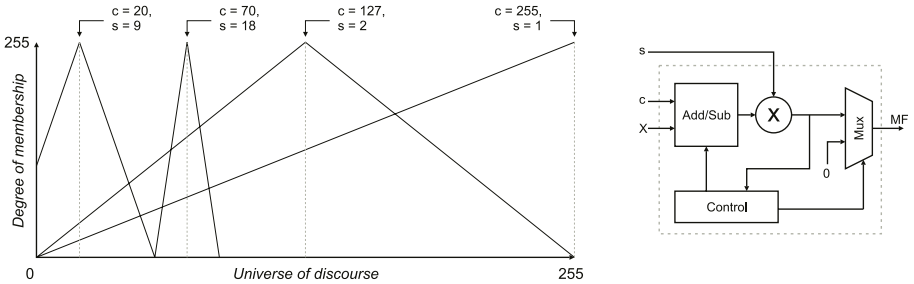


Fig. 2. Architecture of the fuzzy coprocessor

To effectively process the rules, the coprocessor has a pipelined structure composed by the five stages shown in figure 2. A detailed description of operations and hardware implementation of the various stages is given in the following subsections.

### 2.1 Membership Function Evaluation

In order to evaluate the rules premise, it is necessary to compute the degree of membership of each input to a specified membership function. Many proposed fuzzy processors make use of look-up tables to store membership function shapes. Even if this approach allows to easily create arbitrary shaped functions, it has some serious limiting factors: 1) it requires a significant amount of memory resources (not always available in a System on Chip implementation); 2) it makes difficult to quickly change the membership function characteristics; 3) usually a maximum of two overlapping functions are allowed [9]. To overcome all these limitations the proposed design employs a combinatorial circuit to evaluate on line, and within a single clock cycle, the value of a membership function. Symmetric triangular MFs are used, since they allow an efficient arithmetic computation of their values, and each membership function can be completely described by only two parameters (figure 3 left): its center, and its slope (that is inversely proportional to the width of its base).



**Fig. 3.** Example of membership functions obtained varying the two parameter center and slope (left). Structure of the membership function evaluation block (right)

Only these parameter are stored in a small RAM block (using only 16 word of 16 bit), instead of the entire MF shapes. Using these data, MFs evaluation blocks process 8 bit wide input data, and return an 8 bit degree of membership. This is calculated using the following expression 1:

$$MF(x, c, s) = \begin{cases} 255 - |x - c| \cdot s & \text{when } MF > 0 \\ 0 & \text{when } MF < 0 \end{cases} \quad (1)$$

where  $x$  is the input,  $c$  is the center coordinate (8 bit), and  $s$  is the slope (8 bit). This expression is implemented in hardware using an adder/subtractor, an 8x8 bit multiplier and a clipping unit (figure 3 right). The coprocessor employs

two separate membership function generators in order to evaluate concurrently the degree of membership of the two input (figure 2). If the use of the 8x8 bit multiplier is not convenient (e.g. it is not available or it requires too many logic resources to be implemented) it is possible to use a 3 bit barrel shifter instead of it. In this case only MFs with 8 possible slopes can be obtained. In most cases this is not a serious limitation.

## 2.2 Rule Evaluation and Inference

Once the degree of membership of the two input is tested, the fuzzy AND is computed applying the MIN operator to the two 8 bit membership values. The output of the minimum block ( $W_i$ ), representing the rule firing strength, is passed to the next pipeline stage along with the rule output value ( $C_i$ , 8 bit). In order to perform the inference and defuzzification process, all rules have to be evaluated and the partial results accumulated. Since rule outputs are expressed by fuzzy singleton (i.e. numerical values), it is convenient to employ the Fuzzy-Mean (FM) aggregation/defuzzification method [17]. This implies performing the aggregation together with the defuzzification process, without any additional cost. Partial products for the defuzzification stage are generated in the third and fourth pipeline stages, which contain an 8x8 bit multiplier to evaluate the product  $W_i C_i$  and two 16 bit accumulators to add up this product and  $W_i$  respectively. The last pipeline stage computes the crisp 8 bit output by using the Center Of Gravity (COG) defuzzification method. The crisp output is obtained using the following formula:

$$O = \frac{\sum_{i=1}^n W_i C_i}{\sum_{i=1}^n W_i} \quad (2)$$

where  $W_i$  are the rule firing strengths, and  $C_i$  the rules' outputs. For this purpose a 16 bit integer divider, based on the serial restoring algorithm is used.

## 3 Pipelining

A pipeline structure was chosen and the various stages are built in order to maximize the overall throughput. First of all the pipeline was balanced with respect to the propagation delay of each block, thus maximizing the operating clock frequency. This aspect is slightly influenced by the particular technology used; the proposed solution takes into account an implementation in an FPGA provided with embedded RAM blocks and multipliers. In order to exploit the whole pipeline capacity a bus capable of performing a read or write operations in a single clock cycle should be used. This condition is usually verified in most of System on Chip buses, where pipelined or burst operations are supported. Finally, the minimization of pipeline stalls needs an additional optimization of the overall architecture. The last pipeline stage is represented by the integer divider, that needs at least 16 clock cycles to perform a division, and is activated only after all the rules have already been evaluated. From the other side the



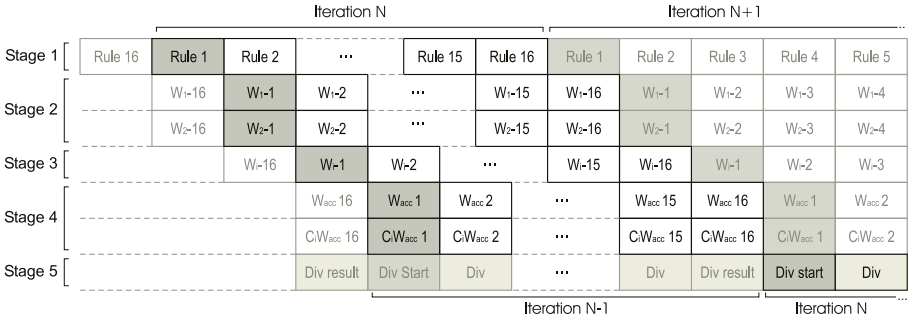


Fig. 4. Detailed pipelined operation

rules evaluation process usually requires a comparable number of clock cycles to be completed, this condition suggests to split the first four pipeline stages from the last one, hence obtaining two independent macro-stages. This simple precaution makes possible to carry out the division while the next set of rules are evaluating. The maximum throughput can be achieved when the number of rules to be evaluated are exactly 16, in this case in fact there are not stalls in the pipeline as showed in figure 4.

### 4 Data Organization and Programming

The coprocessor is programmed by means of simple "load" and "store" assembler instructions or similar read or write operations directed to its memory mapped registers (shown in figure 5).

Typically, the first operation performed by the CPU is the initialization of MFs parameters. This is done by writing in the appropriate registers a 16 bit words describing their centers and base width. Since a maximum of 16 MFs per input can be specified, this phase requires a maximum of 32 instructions to be performed (i.e. 32 clock cycles in most cases). This operation can also

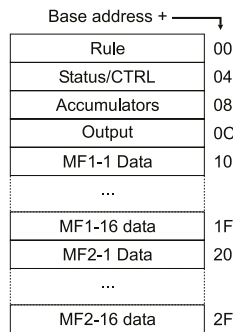


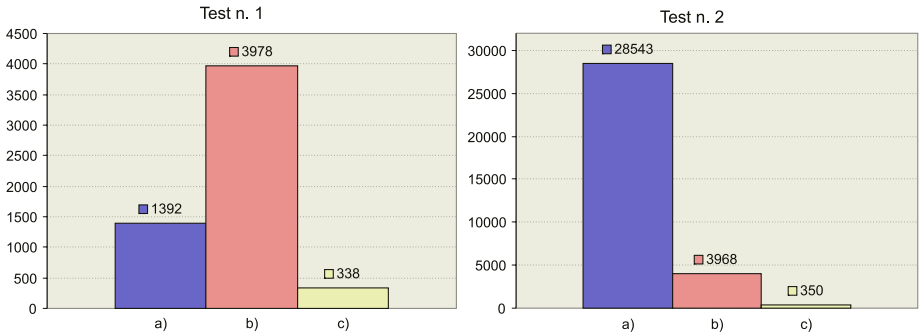
Fig. 5. Memory map of the coprocessor registers

be performed at run time and can even be directed to single MFs. Rules are passed to the coprocessor with a single 32 bit write operation as discussed before. Once the coprocessor receives a rule, it is immediately processed by the first four pipeline stages, and the results accumulated. When all the rules have been evaluated the CPU can start the divider unit to obtain the crisp output. A new set of rules can be immediately evaluated, while the divider is still running. The passed rules and their numbers depend on the particular fuzzy algorithm executed: it is possible to have a fixed set of rules, repeated at each iteration (changing the input values if needed); or a dynamic one, where the rules and their numbers can vary according to some adaptive algorithm. Another interesting possibility is to use the coprocessor to execute more than one fuzzy algorithm in a time-shared manner. This can be useful in applications performing several tasks, each requiring a different fuzzy algorithm (like the one found in automotive or robotics field [2][7]). To switch from one fuzzy context to another it is necessary to change the MF parameters and to use a different set of rules. This operation can be performed in few clock cycles as discussed before, since it doesn't require the computation of new MF values and/or to update look-up tables content; moreover, it can be implemented with compact and modular code. If the whole number of MFs used is less or equal to 16 per input, it is even possible to store all the parameters belonging to different fuzzy algorithms in the MF memory at the beginning and only for once; and referencing the appropriate MF in the rules simply by its index number.

## 5 Experimental Results

The coprocessor was described in VHDL language, and was implemented in a Xilinx Spartan3-200 FPGA. This device is equipped with 1920 logic blocks (slices), 216Kbit of internal memory, available as Block RAMs, and 12 embedded 18x18 bit multiplier. The design required about 267 slices (13%) and only 64 byte of RAM. It was capable of running at a clock frequency of 73MHz, after a fully automatic Place & Route process. The coprocessor was tested by implementing it in a System on Chip composed by a Xilinx MicroBlaze 32 bit RISC processor core [18], running at a clock frequency of 66MHz. The coprocessor was attached to the CPU On-chip Peripheral Bus (OPB) [19]. Since the MicroBlaze processor is a Harvard machine, it was configured to use the dedicated Local Memory Bus (LMB) to access the instruction memory.

The system performances were tested measuring the execution time of an ANSI C program (running on the MicroBlaze processor) implementing a generic fuzzy controller, employing 3 MFs per input, and 9 rules. The execution time was measured by means of an hardware timer, implemented on the system, counting the clock cycles. The first test measured the execution time of a single iteration of the basic fuzzy controller. The controller was implemented: a) in software, using LUT-based MFs; b) in software, using the same arithmetic MF evaluation algorithm used by the coprocessor; c) using the coprocessor. The results are shown in figure 6. As it can be seen the number of clock cycles



**Fig. 6.** Number of clock frequency required in the three experimental cases for the standard (Test 1) algorithm, and the adaptive one (Test 2)

required using the coprocessor (c) are about 4 times smaller than the software-only implementation (a). It has to be noted that the coprocessor would require the same execution time even if the number of rules to evaluate were up to 16. A second test measured the execution time of the adaptive version of the same fuzzy controller, where for sake of simplicity, only one MF was allowed to change at each iteration. As expected the coprocessor allowed significantly faster execution (from 10 to about 80 times). Interestingly, this spread could be greatly increased in both cases, if the program is written in assembler instead of C: in the above mentioned conditions it required only 31 to 35 clock cycles for each iteration (i.e. about  $0.4\mu\text{s}$  @ 66MHz).

## 6 Conclusion

We described the architecture and the FPGA implementation of an adaptive fuzzy coprocessor. The experimental results show a remarkable increase (up to 1000 times) in the execution speed of an adaptive fuzzy algorithm, when compared with an optimized software implementation. Finally, it is worth to note that the proposed architecture is fully scalable in terms of number of inputs by linearly increasing the number of MFs evaluation blocks and slightly modifying the control logic and/or the memory map.

## References

1. L. A. Zadeh: "Fuzzy sets", *Information & Control*, vol. 8, no. 69, June 1965.
2. M. Elarbi-Boudiher, A. Rahmoun, D. Ziou: "A fuzzy control system for visual guidance of an autonomous mobile robot", *Proc. of the IEEE/IEEJ/JSAI Intl. Conf. on Intelligent Transportation Systems*, 5-8 Oct. 1999.
3. J. Urzelai, J.P. Uribe, M. Ezkerra: "Fuzzy controller for wall-following with a non-holonomous mobile robot", *Proc. of the 6th IEEE Intl. Conf. on Fuzzy Systems*, vol. 3, 1-5 July, 1997.

4. W. Li: "'Perception-action' behavior control of a mobile robot in uncertain environments using fuzzy logic", Proc. of the IEEE/RSJ/GI Intl. Conf. on Intelligent Robots and Systems '94, vol. 1, 12-16 Sept. 1994.
5. R.E. Gibson, D.L. Hall, J.A. Stover: "An autonomous fuzzy logic architecture for multisensor data fusion", Proc. of the IEEE Intl. Conf. On Multisensor Fusion and Integration for Intelligent Systems, Las Vegas, 1994.
6. E. Benoit, L. Foulloy, S. Galichet, G. Mauris: "Fuzzy sensor for the perception of colour", Proc. of the 3rd IEEE Conf. on Fuzzy Systems, vol. 3, 26-29 June 1994.
7. H. Eichfeld, A. Mertens, et al.: "Applications of SAE 81C99x fuzzy coprocessors", Proc. of the 1998 IEEE Intl. Conf. on Fuzzy Systems, vol. 1, 4-9 May 1998.
8. M. Sasaki, F. Ueno, T. Inoue: "7.5 MFLIPS fuzzy microprocessor using SIMD and logic-in-memory structure", Proc. Of the 2nd IEEE International Conference on Fuzzy Systems, vol. 1, 28 March-1 April 1993.
9. V. Salapura: "A fuzzy RISC processor", IEEE Transactions on Fuzzy Systems, vol. 8, no. 6, Dec. 2000.
10. H. Eichfeld, M. Klimke, et al. : "A general-purpose fuzzy inference processor", IEEE Micro, vol. 15, issue 3, June 1995.
11. T. Yamakawa: "A survey on fuzzy information processing hardware systems", Proc. of the IEEE Intl. Symposium on Circuits and Systems, vol. 2 , 28 April-3 May 1995.
12. H. Watanabe, W. D. Dettloff, K. E. Yount: "A VLSI fuzzy logic controller with reconfigurable, cascadable architecture", IEEE Journal of Solid-State Circuits, vol. 25, no. 2, April 1990.
13. D. Kim: "An implementation of fuzzy logic controller on the reconfigurable FPGA system", IEEE Transactions on Industrial Electronics, vol. 47, no. 3, June 2000.
14. M. A. Manzoul, D. Jayabharathi: "FPGA for fuzzy controller", IEEE Transactions on Systems, Man, and Cybernetics, vol. 25, no. 1, Jan. 1995.
15. S. Singh, K. S. Rattan: "Implementation of a fuzzy controller on an FPGA using VHDL", Proc. of the 22nd Intl. Conf. of the North American Fuzzy Information Processing Society, 24-26 July 2003.
16. J. M. Jou, P. Y. Chen, S. F. Yang: "An adaptive fuzzy logic controller: its VLSI architecture and applications", IEEE Transactions on Very Large Scale Integrations (VLSI) Systems, vol. 8, no. 1, Feb. 2000.
17. Jager, Ren "Fuzzy Logic in Control" - Ph.D. thesis Delft University of Technology, Department of Electrical Engineering, Control Laboratory, pp. 312. Delft, The Netherlands. ISBN 90-9008318-9.
18. Xilinx Corporation, "MicroBlaze processor reference guide", version 3.2, April 2001.
19. IBM Corporation, "On-Chip Peripheral Bus - Architecture Specifications", version 2.1, April 2001.

# Cascade Ensembles

N. García-Pedrajas<sup>1</sup>, D. Ortiz-Boyer<sup>1</sup>, R. del Castillo-Gomariz<sup>1</sup>,  
and C. Hervás-Martínez<sup>1</sup>

University of Córdoba,  
Campus Universitario de Rabanales,  
Córdoba, Spain  
{npedrajas, dortiz, rcastillo, chervas}@uco.es

**Abstract.** Neural network ensembles are widely use for classification and regression problems as an alternative to the use of isolated networks. In many applications, ensembles has proven a performance above the performance of just one network.

In this paper we present a new approach to neural network ensembles that we call “cascade ensembles”. The approach is based on two ideas: (i) the ensemble is created constructively, and (ii) the output of each network is fed to the inputs of the subsequent networks. In this way we make a cascade of networks.

This method is compared with standard ensembles in several problems of classification with excellent performance.

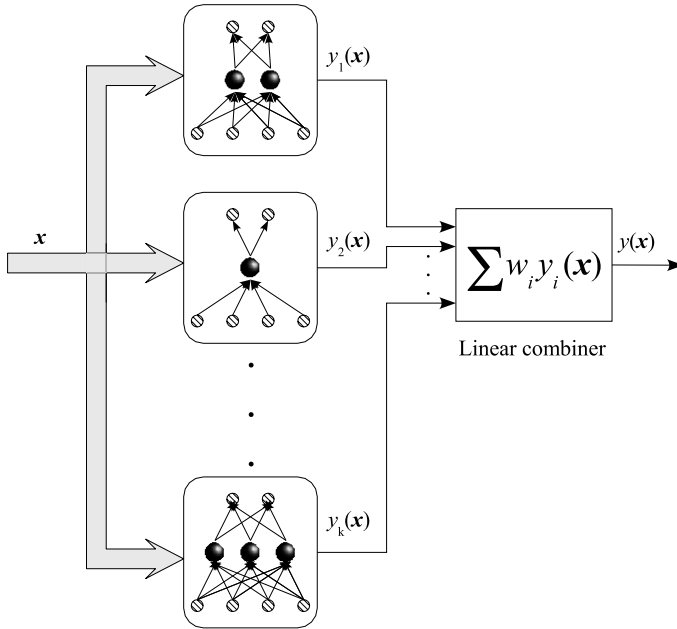
## 1 Introduction

Neural network ensembles[11] are receiving increasing attention in recent neural network research, due to their interesting features. They are a powerful tool especially when facing complex problems. Network ensembles are usually made up of a linear combination of several networks that have been trained using the same data (see Figure 1, although the actual sample used by each network to learn can be different. Each network within the ensemble has a potentially different weight in the output of the ensemble. Several works have shown [11] that the network ensemble has a generalisation error generally smaller than that obtained with a single network and also that the variance of the ensemble is lesser than the variance of a single network. If the networks have more than one output, a different weight is usually assigned to each output. The ensembles of neural networks have some of the advantages of large networks without their problems of long training time and risk of over-fitting. For more detailed descriptions of ensembles the reader is referred to [2] [14] [3] [10] [5].

Although there is no clear distinction between the different kinds of multinet networks [7] [1] [6], we follow the distinction of [13]. In an ensemble several redundant approximations to the same function are combined by some method, and in a modular system the task is decomposed into a number of simpler components.

This combination of several networks that cooperate in solving a given task has other important advantages such as [8] [13]:

- They can perform more complex tasks than any of their subcomponents [15].
- They can make an overall system easier to understand and modify, as the whole system is decomposed in smaller parts.
- They are more robust than a single network.



**Fig. 1.** Standard ensemble of neural networks

In most cases, neural networks in an ensemble are designed independently or sequentially, so the advantages of interaction and cooperation among the individual networks are not exploited. Earlier works separate the design and learning process of the individual networks from the combination of the trained networks. In this work we propose a framework for designing ensembles, where the training and combination of the individual networks are carried out together, in order to get more cooperative networks and more effective combinations of them.

The design of neural network ensembles implies making many decisions that have a major impact on the performance of the ensembles. The most important decisions that we must face when designing an ensemble are the following:

- The method for designing and training the individual networks.
- The method of combining the individual networks, and the mechanism for obtaining individual weights for each network if such is the case.
- The measures of performance of the individual networks.

- The methods for encouraging diversity among the members of the ensembles and how to measure such diversity.
- The method of selection of patterns that are used by each network to learn.
- Whether to include regularization terms and their form.

Techniques using multiple models usually consist of two independent phases: model generation and model combination [10]. The disadvantage of this approach is that the combination is not considered during the generation of the models. With this approach the possible interactions among the trained cannot be exploited until the combination stage [8], and the benefits that can be obtained from this interactions during the learning stage are lost.

However, several researchers [16][9] have recently shown that some information about cooperation is useful for obtaining better ensembles. This new approach opens a wide field where the design and training of the different networks must be interdependent.

In this paper, we present a new model for constructively making the ensemble. Our basic aim is improving the combination of networks. In this way we created an ensemble where the  $i$ -th network receives as inputs the outputs of the  $i - 1$  already trained networks. This ensemble is so-called *cascade ensemble*. In this way, some of the ideas of the constructive cascade-correlation networks are applied to ensemble construction [4].

This paper is organised as follows: Section 2 describes our model of cascade ensembles; Section 3 shows the experimental results of our model and its comparison with standard ensembles; finally Section 4 states the conclusions of our work.

## 2 Cascade Ensembles

The main advantage of cascade ensembles is that each time a new network is added to the ensemble, that network knows the outputs of the previously trained networks. In this way, the construction of the ensemble is not separated in two stages: training and combination of the networks.

In the proposed model, each network tries to refine the classification carried out by the previous models. For a pattern  $\mathbf{x}$  the input to  $k$ -th network is  $(\mathbf{x}, y_1, y_2, \dots, y_{k-1})$ , where  $y_i$  is the output of network  $i$  (see Figure 2).

Cascade ensembles can also be defined in other ways that are under development. The main advantages of this approach are:

1. The ensemble is made constructively. In this way, the complexity of the ensemble can match the problem, as the addition of new networks can be stopped when the required performance is achieved.
2. The stages of training and combination are not separated. The new networks have the knowledge acquired by previous networks.
3. There is no need of an additional combination scheme and the subsequent optimisation algorithm needed to set its parameters.

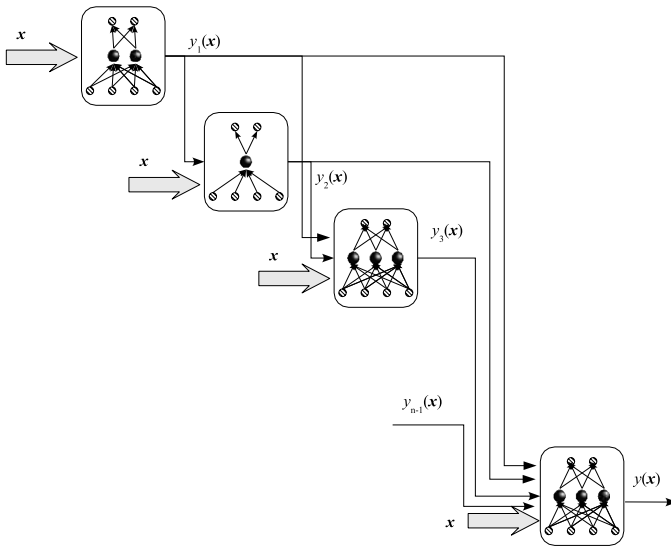


Fig. 2. Cascade ensemble of neural networks

### 3 Experiments

The experiments were carried out with the objective of testing our model against an standard ensemble. We have applied our model and standard ensemble method to several real-world problems of classification. These problems are briefly described in Table 1. These nine datasets cover a wide variety of problems. There are problems with different number of available patterns, different number of classes, different kind of inputs, and of different areas of application. Testing our model on this variety of problems can give us a clear idea of its performance.

The tests were conducted following the guidelines of L. Prechelt [12]. Each set of available data was divided into three subsets: 50% of the patterns were used for learning, 25% of them for validation and the remaining 25% for testing

**Table 1.** Summary of data sets. The features of each data set can be C(continuous), B(binary) or N(nominal). The Inputs column shows the number of inputs of the network as it depends not only on the number of input variables but also on their type

| Data set  | Cases |      | Classes | Features |    |    | Inputs |
|-----------|-------|------|---------|----------|----|----|--------|
|           | Train | Test |         | C        | B  | N  |        |
| Anneal    | 674   | 224  | 5       | 6        | 14 | 18 | 59     |
| Glass     | 161   | 53   | 6       | 9        | -  | -  | 9      |
| Heart     | 202   | 68   | 2       | 6        | 1  | 6  | 13     |
| Hepatitis | 117   | 38   | 2       | 6        | 13 | -  | 19     |
| Horse     | 273   | 91   | 3       | 13       | 2  | 5  | 58     |
| Pima      | 576   | 192  | 2       | 8        | -  | -  | 8      |
| Sonar     | 104   | 104  | 2       | 60       | -  | -  | 60     |
| Promoters | 80    | 26   | 2       | -        | -  | 57 | 114    |
| Vehicle   | 635   | 211  | 4       | 18       | -  | -  | 18     |



**Table 2.** Results for the classification of the described problems for cascade and standard ensemble. For each problem we show the averaged training and test error, the standard deviation, and the best and worst result

| Problem   | Model | Training |        |        |        | Generalisation |        |        |        | <i>t</i> -test |
|-----------|-------|----------|--------|--------|--------|----------------|--------|--------|--------|----------------|
|           |       | Mean     | SD     | Best   | Worst  | Mean           | SD     | Best   | Worst  |                |
| Anneal    | Casc  | 0.0113   | 0.0140 | 0.0000 | 0.0460 | 0.0250         | 0.0122 | 0.0134 | 0.0580 | –              |
|           | Std   | 0.0064   | 0.0056 | 0.0015 | 0.0179 | 0.0106         | 0.0064 | 0.0015 | 0.0223 | 0.0000         |
| Glass     | Casc  | 0.1201   | 0.0183 | 0.0870 | 0.1553 | 0.2692         | 0.0325 | 0.2075 | 0.3396 | –              |
|           | Std   | 0.1660   | 0.0147 | 0.1429 | 0.1925 | 0.2943         | 0.0236 | 0.2453 | 0.3396 | 0.0011         |
| Heart     | Casc  | 0.1229   | 0.0102 | 0.0990 | 0.1436 | 0.1373         | 0.0156 | 0.1029 | 0.1618 | –              |
|           | Std   | 0.0726   | 0.0056 | 0.0545 | 0.0792 | 0.1539         | 0.0120 | 0.1324 | 0.1912 | 0.0000         |
| Hepatitis | Casc  | 0.0499   | 0.0129 | 0.0342 | 0.0855 | 0.1140         | 0.0222 | 0.0789 | 0.1842 | –              |
|           | Std   | 0.0051   | 0.0043 | 0.0000 | 0.0085 | 0.1289         | 0.0174 | 0.1053 | 0.1579 | 0.0054         |
| Horse     | Casc  | 0.0223   | 0.0054 | 0.0110 | 0.0366 | 0.2934         | 0.0252 | 0.2527 | 0.3516 | –              |
|           | Std   | 0.0289   | 0.0036 | 0.0220 | 0.0330 | 0.3018         | 0.0160 | 0.2637 | 0.3297 | 0.1274         |
| Pima      | Casc  | 0.2215   | 0.0086 | 0.2066 | 0.2500 | 0.2061         | 0.0196 | 0.1667 | 0.2552 | –              |
|           | Std   | 0.2218   | 0.0048 | 0.2118 | 0.2326 | 0.2049         | 0.0137 | 0.1823 | 0.2396 | 0.7819         |
| Promoters | Casc  | 0.0000   | 0.0000 | 0.0000 | 0.0000 | 0.1526         | 0.0189 | 0.1154 | 0.1923 | –              |
|           | Std   | 0.0000   | 0.0000 | 0.0000 | 0.0000 | 0.1538         | 0.0000 | 0.1538 | 0.1538 | 0.7109         |
| Sonar     | Casc  | 0.0295   | 0.0182 | 0.0000 | 0.0577 | 0.1766         | 0.0137 | 0.1442 | 0.2115 | –              |
|           | Std   | 0.0000   | 0.0000 | 0.0000 | 0.0000 | 0.1779         | 0.0090 | 0.1635 | 0.2019 | 0.6703         |
| Vehicle   | Casc  | 0.1546   | 0.0136 | 0.1276 | 0.1858 | 0.1919         | 0.0212 | 0.1517 | 0.2417 | –              |
|           | Std   | 0.1796   | 0.0072 | 0.1669 | 0.2016 | 0.1962         | 0.0190 | 0.1611 | 0.2464 | 0.4157         |

the generalization error. There is an exception, Sonar problem, as the patterns of this problem are prearranged in two subsets due to their specific features. The ensembles are made up by 10 networks, each one with 10 hidden nodes with logistic transfer function. The training parameters for the back-propagation algorithm are  $\eta = 0.01$  and  $\alpha = 0.01$ .

The training and generalisation results are shown in Table 2. The table shows also the *p*-values of paired *t*-test for generalisation error. The results show that the cascade algorithm performs better than standard ensembles in 7 problems (in three of them with statistical significance). This is interesting as the cascade models avoids the use any method for combining the outputs.

## 4 Conclusions and Future Work

In this paper we have introduced a new approach to neural network ensembles called *cascade ensembles* where the ensemble is created constructively. The preliminary results showed in this paper are very promising and open a wide field of study for this approach.

As work for the future, we are working in two ideas. Firstly, the application of sampling methods, such as bagging and boosting, to our model. Secondly, the modification of the cascade inputs, as there different ways for feeding the outputs of the trained networks to the new networks.

## Acknowledgments

This work has been financed in part by the project TIC2002-04036-C05-02 of the Spanish CICYT and FEDER funds.

## References

1. R. Avnimelech and N. Intrator. Boosted mixture of experts: an ensemble learning scheme. *Neural Computation*, 11(2):483–497, 1999.
2. T. G. Dietterich. Ensemble methods in machine learning. In J. Kittler and F. Roli, editors, *Proceedings of the First International Workshop on Multiple Classifier Systems*, pages 1–15. Springer-Verlag, 2000.
3. S. Dzeroski and B. Zenko. Is combining classifiers with stacking better than selecting the best one? *Machine Learning*, 54:255–273, 2004.
4. S. E. Fahlman and C. Lebiere. The cascade-correlation architecture. In D. S. Touretzky, editor, *Advances in Neural Information Systems 2*, pages 524–532, San Mateo, CA, 1990. Morgan Kaufman.
5. A. Fern and R. Givan. Online ensemble learning: An empirical study. *Machine Learning*, 53:71–109, 2003.
6. G. Giacinto and F. Roli. Dynamic classifier selection. In *Multiple Classifier Systems 2000*, volume 1857 of *Lecture Notes in Computer Science*, pages 177–189, 2000.
7. J. Hansen. Combining predictors: Comparison of five meta machine learning methods. *Information Science*, 119(1–2):91–105, 1999.
8. Y. Liu, X. Yao, and T. Higuchi. Evolutionary ensembles with negative correlation learning. *IEEE Transactions on Evolutionary Computation*, 4(4):380–387, November 2000.
9. Y. Liu, X. Yao, Q. Zhao, and T. Higuchi. Evolving a cooperative population of neural networks by minimizing mutual information. In *Proc. of the 2001 IEEE Congress on Evolutionary Computation*, pages 384–389, Seoul, Korea, May 2001.
10. C. J. Merz. Using correspondence analysis to combine classifiers. *Machine Learning*, 36(1):33–58, July 1999.
11. M. P. Perrone and L. N. Cooper. When networks disagree: Ensemble methods for hybrid neural networks. In R. J. Mammone, editor, *Neural Networks for Speech and Image Processing*, pages 126–142. Chapman – Hall, 1993.
12. L. Prechelt. Proben1 – A set of neural network benchmark problems and benchmarking rules. Technical Report 21/94, Fakultät für Informatik, Universität Karlsruhe, Karlsruhe, Germany, September 1994.
13. A. J. C. Sharkey. On combining artificial neural nets. *Connection Science*, 8:299–313, 1996.
14. G. I. Webb. Multiboosting: A technique for combining boosting and wagging. *Machine Learning*, 40(2):159–196, August 2000.
15. X. Yao and Y. Liu. Making use of population information in evolutionary artificial neural networks. *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics*, 28(3):417–425, June 1998.
16. Z-H. Zhou, J. Wu, and W. Tang. Ensembling neural networks: Many could be better than all. *Artificial Intelligence*, 137(1–2):239–253, May 2002.

# Ensembles of Multilayer Feedforward: Some New Results<sup>1</sup>

Joaquín Torres-Sospedra, Carlos Hernández-Espinosa,  
and Mercedes Fernández-Redondo

Universidad Jaume I. Dept. de Ingeniería y Ciencia de los Computadores,  
Avda Vicente Sos Baynat s/n, 12071 Castellon, Spain  
{espinosa, redondo}@icc.uji.es

**Abstract.** As shown in the bibliography, training an ensemble of networks is an interesting way to improve the performance with respect to a single network. However there are several methods to construct the ensemble. In this paper we present some new results in a comparison of twenty different methods. We have trained ensembles of 3, 9, 20 and 40 networks to show results in a wide spectrum of values. The results show that the improvement in performance above 9 networks in the ensemble depends on the method but it is usually low. Also, the best method for a ensemble of 3 networks is called “Decorrelated” and uses a penalty term in the usual Backpropagation function to decorrelate the network outputs in the ensemble. For the case of 9 and 20 networks the best method is conservative boosting. And finally for 40 networks the best method is Cels.

## 1 Introduction

The most important property of a neural network (NN) is the generalization capability. The ability to correctly respond to inputs which were not used in the training set.

One technique to increase the generalization capability with respect to a single NN consist on training an ensemble of NN, i.e., to train a set of NNs with different weight initialization or properties and combine the outputs of the different networks in a suitable manner to give a single output.

It is clear from the bibliography that this procedure in general increases the generalization capability [1,2].

The two key factors to design an ensemble are how to train the individual networks and how to combine the different outputs to give a single output.

Among the methods of combining the outputs, the two most popular are *voting* and *output averaging* [3]. In this paper we will normally use *output averaging* because it has no problems of ties and gives a reasonable performance.

In the other aspect, nowadays, there are several different methods in the bibliography to train the individual networks and construct the ensemble [1-3].

However, there is a lack of comparison among the different methods and it is not clear which one can provide better results.

---

<sup>1</sup> This research was supported by the project MAPACI TIC2002-02273 of CICYT in Spain.

One comparison can be found in [4], it is a previous work developed by our research group. In paper [4], eleven different methods are compared.

Now, we present more complete results by including nine new methods, so we increase the number of methods in the comparison to a total of twenty. The empirical results are quite interesting, one of the new methods analyzed in this paper seems to have the best performance in several situations.

## 2 Theory

In this section we briefly review the new nine ensemble methods introduced in this paper for comparison. The description of the rest of methods denoted by “Simple Ensemble”, “Ola”, “Evol”, “Decorrelated”, “Decorrelated2”, “CVC”, “Cels”, “Boosting”, “Bag\_Noise” and “Adaboost”, can be found in reference [4] and in the references cited there.

**CVC Version 2:** In the usual CVC the available data is divided in training, cross-validation and testing. After that, the data for training is divided by the number of networks giving several subsets. Then, one different subset is omitted for each network and the network is training with the rest of subsets.

The version 2 of CVC included in this paper is used in reference [5]. The data for training and cross-validation is jointed in one set and with this jointed set the usual division of CVC is performed. In this case, one subset is omitted for each network and the omitted subset is used for cross-validation.

**Aveboost:** Aveboost is the abbreviation of Average Boosting. This method was proposed in reference [6] as a variation of Adaboost. In Adaboost, it is calculated a probability for each pattern of being included in the training set for the following network. In this case a weighted adaptation of the probabilities is performed. The method is complex and a full description can be found in the reference.

**TCA, Total Corrective Adaboost:** It was also proposed in reference [6] and it is another variation of Adaboost. In this case the calculation of the probability distribution for each network is treated as an optimization problem and an iterative process is performed. The algorithm is complex and a full description is in [6].

**Aggressive Boosting:** Aggressive Boosting is a variation of Adaboost. It is reviewed in [7]. In this case it is used a common step to modify the probabilities of a pattern, this common step can increase or decrease the probability of the pattern for being included in the next training set.

**Conservative Boosting:** It is another variation of Adaboost reviewed in [7]. In this case the probability of the well classified patterns is decreased and the probability of wrong classified patterns is kept unchanged.

**ArcX4:** It is another variation of Boosting, it was proposed and studied in reference [8]. The method selects training patterns according to a distribution, and the probability of the pattern depend on the number of times the pattern was not correctly classified by the previous networks. The combination procedure proposed in the reference

is the mean average of the different networks. In our experiments we have used this procedure and also voting.

**EENCL Evolutionary Ensemble with Negative Correlation:** This method is proposed in reference [9]. The ensemble is build as a population of a genetic algorithm, the fitness function is selected to consider the precision in the classification of the individual networks and also to penalize the correlation among the different networks in the ensemble. The details can be found in the reference. In this paper two variations of the method proposed in the reference are used. They are denoted by EENCL UG (the ensemble used in this case is given by the last generation of the evolutionary algorithm) and EENCL MG (in this case the best generation is selected).

### 3 Experimental Results

We have applied the twenty ensemble methods to ten different classification problems. They are from the UCI repository of machine learning databases. Their names are Cardiac Arrhythmia Database (Aritm), Dermatology Database (Derma), Protein Location Sites (Ecoli), Solar Flares Database (Flare), Image Segmentation Database (Image), Johns Hopkins University Ionosphere Database (Ionos), Pima Indians Diabetes (Pima), Haberman's survival data (Surv), Vowel Recognition (Vowel) and Wisconsin Breast Cancer Database (Wdbc).

We have constructed ensembles of a wide number of networks, in particular 3, 9, 20 and 40 networks in the ensemble. In this case, we can test the results in a wide set of situations.

We trained the ensembles of 3, 9, 20 and 40 networks. We repeated this process of training an ensemble ten times for different partitions of data in training, cross-validation and test sets. With this procedure we can obtain a mean performance of the ensemble for each database (the mean of the ten ensembles) and an error in the performance calculated by standard error theory. The results of the performance are in table 1 for the case of ensembles of three networks, in table 2 for the case of nine and in table 3 for the case of 20. We omit the results of 40 networks by the lack of space and because the improvement of increasing the number of networks is in general low, a resume of the performance for 40 networks can be found below.

By comparing the results of table 1, and 2 with the results of a single network we can see that the improvement by the use of the ensemble methods depends clearly on the problem. For example in databases Aritm (except for the case of CVC version 2), Flare, Pima and Wdbc there is not a clear improvement.

In the rest of databases there is an improvement, perhaps the most important one is in database Vowel.

There is, however, one exception in the performance of the method Evol. This method did not work well in our experiments. In the original reference the method was tested in the database Heart. The results for a single network were 60%, for a simple ensemble 61.42% and for Evol 67.14%. We have performed some experiments with this database and our results for a simple network are  $82.0 \pm 0.9$ , clearly different.

**Table 1.** Results for the ensemble of three networks

|                | ARITM      | DERMA      | ECOLI      | FLARE      | IMAGEN       |
|----------------|------------|------------|------------|------------|--------------|
| Single Net.    | 75.6 ± 0.7 | 96.7 ± 0.4 | 84.4 ± 0.7 | 82.1 ± 0.3 | 96.3 ± 0.2   |
| Adaboost       | 71.8 ± 1.8 | 98.0 ± 0.5 | 85.9 ± 1.2 | 81.7 ± 0.6 | 96.8 ± 0.2   |
| Bagging        | 74.7 ± 1.6 | 97.5 ± 0.6 | 86.3 ± 1.1 | 81.9 ± 0.6 | 96.6 ± 0.3   |
| Bag_Noise      | 75.5 ± 1.1 | 97.6 ± 0.7 | 87.5 ± 1.0 | 82.2 ± 0.4 | 93.4 ± 0.4   |
| Boosting       | 74.4 ± 1.2 | 97.3 ± 0.6 | 86.8 ± 0.6 | 81.7 ± 0.4 | 95.0 ± 0.4   |
| Cels_m         | 73.4 ± 1.3 | 97.7 ± 0.6 | 86.2 ± 0.8 | 81.2 ± 0.5 | 96.82 ± 0.15 |
| CVC            | 74.0 ± 1.0 | 97.3 ± 0.7 | 86.8 ± 0.8 | 82.7 ± 0.5 | 96.4 ± 0.2   |
| Decorrelated   | 74.9 ± 1.3 | 97.2 ± 0.7 | 86.6 ± 0.6 | 81.7 ± 0.4 | 96.7 ± 0.3   |
| Decorrelated2  | 73.9 ± 1.0 | 97.6 ± 0.7 | 87.2 ± 0.9 | 81.6 ± 0.4 | 96.7 ± 0.3   |
| Evol           | 65.4 ± 1.4 | 57 ± 5     | 57 ± 5     | 80.7 ± 0.7 | 77 ± 5       |
| Ola            | 74.7 ± 1.4 | 91.4 ± 1.5 | 82.4 ± 1.4 | 81.1 ± 0.4 | 95.6 ± 0.3   |
| CVC version 2  | 76.1 ± 1.6 | 98.0 ± 0.3 | 86.8 ± 0.9 | 82.5 ± 0.6 | 96.9 ± 0.3   |
| AveBoost       | 73.4 ± 1.3 | 97.6 ± 0.7 | 85.3 ± 1.0 | 81.8 ± 0.8 | 96.8 ± 0.2   |
| TCA            | 70.7 ± 1.9 | 96.1 ± 0.6 | 85.4 ± 1.3 | 81.9 ± 0.7 | 94.8 ± 0.5   |
| ArcX4          | 75.4 ± 0.8 | 97.8 ± 0.5 | 85.3 ± 1.1 | 78.3 ± 0.9 | 96.6 ± 0.2   |
| ArcX4 Voting   | 73.0 ± 0.8 | 97.0 ± 0.5 | 85.7 ± 1.1 | 80.6 ± 0.9 | 96.5 ± 0.2   |
| Aggressive B   | 72.3 ± 1.9 | 97.0 ± 0.5 | 85.7 ± 1.4 | 81.9 ± 0.9 | 96.6 ± 0.3   |
| Conservative B | 74.8 ± 1.3 | 96.9 ± 0.8 | 85.4 ± 1.3 | 82.1 ± 1.0 | 96.5 ± 0.3   |
| EENCL UG       | 71 ± 2     | 96.8 ± 0.9 | 86.6 ± 1.2 | 81.4 ± 0.8 | 96.3 ± 0.2   |
| EENCL MG       | 74.5 ± 1.3 | 97.2 ± 0.8 | 86.6 ± 1.2 | 81.9 ± 0.5 | 96.0 ± 0.2   |
| Simple Ens.    | 73.4 ± 1.0 | 97.2 ± 0.7 | 86.6 ± 0.8 | 81.8 ± 0.5 | 96.5 ± 0.2   |

**Table 1 (continuation).** Results for the ensemble of three networks

|                | IONOS      | PIMA       | SURVI      | VOWEL       | WDBC       |
|----------------|------------|------------|------------|-------------|------------|
| Single Net.    | 87.9 ± 0.7 | 76.7 ± 0.6 | 74.2 ± 0.8 | 83.4 ± 0.6  | 97.4 ± 0.3 |
| Adaboost       | 88.3 ± 1.3 | 75.7 ± 1.0 | 75.4 ± 1.6 | 88.43 ± 0.9 | 95.7 ± 0.6 |
| Bagging        | 90.7 ± 0.9 | 76.9 ± 0.8 | 74.2 ± 1.1 | 87.4 ± 0.7  | 96.9 ± 0.4 |
| Bag_Noise      | 92.4 ± 0.9 | 76.2 ± 1.0 | 74.6 ± 0.7 | 84.4 ± 1.0  | 96.3 ± 0.6 |
| Boosting       | 88.9 ± 1.4 | 75.7 ± 0.7 | 74.1 ± 1.0 | 85.7 ± 0.7  | 97.0 ± 0.4 |
| Cels_m         | 91.9 ± 1.0 | 76.0 ± 1.4 | 73.4 ± 1.3 | 91.1 ± 0.7  | 97.0 ± 0.4 |
| CVC            | 87.7 ± 1.3 | 76.0 ± 1.1 | 74.1 ± 1.4 | 89.0 ± 1.0  | 97.4 ± 0.3 |
| Decorrelated   | 90.9 ± 0.9 | 76.4 ± 1.2 | 74.6 ± 1.5 | 91.5 ± 0.6  | 97.0 ± 0.5 |
| Decorrelated2  | 90.6 ± 1.0 | 75.7 ± 1.1 | 74.3 ± 1.4 | 90.3 ± 0.4  | 97.0 ± 0.5 |
| Evol           | 83.4 ± 1.9 | 66.3 ± 1.2 | 74.3 ± 0.6 | 77.5 ± 1.7  | 94.4 ± 0.9 |
| Ola            | 90.7 ± 1.4 | 69.2 ± 1.6 | 75.2 ± 0.9 | 83.2 ± 1.1  | 94.2 ± 0.7 |
| CVC version 2  | 89.7 ± 1.4 | 76.8 ± 1.0 | 74.1 ± 1.2 | 89.8 ± 0.9  | 96.7 ± 0.3 |
| AveBoost       | 89.4 ± 1.3 | 76.5 ± 1.1 | 75.1 ± 1.2 | 88.1 ± 1.0  | 95.6 ± 0.5 |
| TCA            | 87.9 ± 1.2 | 75.4 ± 0.8 | 73.0 ± 1.5 | 87.5 ± 1.1  | 91 ± 4     |
| ArcX4          | 89.4 ± 1.0 | 76.0 ± 0.8 | 68 ± 2     | 90.8 ± 0.9  | 96.3 ± 0.6 |
| ArcX4 Voting   | 89.0 ± 1.0 | 76.3 ± 0.8 | 74 ± 2     | 86.2 ± 0.9  | 96.1 ± 0.6 |
| Aggressive B   | 90.3 ± 0.9 | 74.3 ± 1.5 | 73.8 ± 1.5 | 86.9 ± 1.2  | 96.6 ± 0.6 |
| Conservative B | 89.4 ± 1.0 | 75.6 ± 1.2 | 75.6 ± 1.1 | 88.8 ± 1.1  | 97.0 ± 0.6 |
| EENCL UG       | 93.0 ± 1.0 | 74.7 ± 1.0 | 73.9 ± 1.2 | 87.2 ± 0.8  | 96.2 ± 0.4 |
| EENCL MG       | 93.7 ± 0.9 | 75.3 ± 1.0 | 73.9 ± 0.8 | 87.4 ± 0.7  | 96.4 ± 0.5 |
| Simple Ens.    | 91.1 ± 1.1 | 75.9 ± 1.2 | 74.3 ± 1.3 | 88.0 ± 0.9  | 96.9 ± 0.5 |

Now, we can compare the results of tables 1 and 2 for ensembles of different number of networks. We can see that the results are in general similar and the improvement of training an increasing number of networks, for example 20 and 40, is in general low. Taking into account the computational cost, we can say that the best alternative for an application is an ensemble of three or nine networks.

We have also calculated the percentage of error reduction of the ensemble with respect to a single network. We have used equation 1 for this calculation.

$$PorError_{reduction} = 100 \cdot \frac{PorError_{single\ network} - PorError_{ensemble}}{PorError_{single\ network}} \quad (1)$$

**Table 2.** Results for the Ensemble of nine networks

|                | ARITM      | DERMA      | ECOLI      | FLARE      | IMAGEN       |
|----------------|------------|------------|------------|------------|--------------|
| Adaboost       | 73.2 ± 1.6 | 97.3 ± 0.5 | 84.7 ± 1.4 | 81.1 ± 0.7 | 97.3 ± 0.3   |
| Bagging        | 75.9 ± 1.7 | 97.7 ± 0.6 | 87.2 ± 1.0 | 82.4 ± 0.6 | 96.7 ± 0.3   |
| Bag_Noise      | 75.4 ± 1.2 | 97.0 ± 0.7 | 87.2 ± 0.8 | 82.4 ± 0.5 | 93.4 ± 0.3   |
| Cels_m         | 74.8 ± 1.3 | 97.3 ± 0.6 | 86.2 ± 0.8 | 81.7 ± 0.4 | 96.6 ± 0.2   |
| CVC            | 74.8 ± 1.3 | 97.6 ± 0.6 | 87.1 ± 1.0 | 81.9 ± 0.6 | 96.6 ± 0.2   |
| Decorrelated   | 76.1 ± 1.0 | 97.6 ± 0.7 | 87.2 ± 0.7 | 81.6 ± 0.6 | 96.9 ± 0.2   |
| Decorrelated2  | 73.9 ± 1.1 | 97.6 ± 0.7 | 87.8 ± 0.7 | 81.7 ± 0.4 | 96.84 ± 0.18 |
| Evol           | 65.9 ± 1.9 | 54 ± 6     | 57 ± 5     | 80.6 ± 0.8 | 67 ± 4       |
| Ola            | 72.5 ± 1.0 | 86.7 ± 1.7 | 83.5 ± 1.3 | 80.8 ± 0.4 | 96.1 ± 0.2   |
| CVC version 2  | 76.1 ± 1.6 | 98.0 ± 0.3 | 86.8 ± 0.9 | 82.5 ± 0.6 | 96.9 ± 0.3   |
| AveBoost       | 73.4 ± 1.3 | 97.6 ± 0.7 | 85.3 ± 1.0 | 81.8 ± 0.8 | 96.8 ± 0.2   |
| TCA            | 70.7 ± 1.9 | 96.1 ± 0.5 | 85.4 ± 1.3 | 81.9 ± 0.7 | 94.8 ± 0.5   |
| ArcX4          | 75.4 ± 0.8 | 97.8 ± 0.5 | 85.3 ± 1.1 | 78.3 ± 0.9 | 96.6 ± 0.2   |
| ArcX4 Voting   | 73.3 ± 0.8 | 97.6 ± 0.5 | 84.9 ± 1.1 | 80.1 ± 0.9 | 97.2 ± 0.2   |
| Aggressive B   | 72.3 ± 1.9 | 97.0 ± 0.5 | 85.7 ± 1.4 | 81.9 ± 0.9 | 96.6 ± 0.3   |
| Conservative B | 74.8 ± 1.3 | 96.9 ± 0.8 | 85.4 ± 1.3 | 82.1 ± 1.0 | 96.5 ± 0.3   |
| EENCL UG       | 71 ± 2     | 96.8 ± 0.9 | 86.6 ± 1.2 | 81.4 ± 0.8 | 96.3 ± 0.2   |
| EENCL MG       | 74.5 ± 1.3 | 97.2 ± 0.8 | 86.6 ± 1.2 | 81.9 ± 0.5 | 96.0 ± 0.2   |
| Simple Ens     | 73.8 ± 1.1 | 97.5 ± 0.7 | 86.9 ± 0.8 | 81.6 ± 0.4 | 96.7 ± 0.3   |

**Table 2 (continuation).** Results for the ensemble of nine networks

|                | IONOS      | PIMA       | SURVI      | VOWEL      | WDBC       |
|----------------|------------|------------|------------|------------|------------|
| Adaboost       | 89.4 ± 0.8 | 75.5 ± 0.9 | 74.3 ± 1.4 | 94.8 ± 0.7 | 95.7 ± 0.7 |
| Bagging        | 90.1 ± 1.1 | 76.6 ± 0.9 | 74.4 ± 1.5 | 90.8 ± 0.7 | 97.3 ± 0.4 |
| Bag_Noise      | 93.3 ± 0.6 | 75.9 ± 0.9 | 74.8 ± 0.7 | 85.7 ± 0.9 | 95.9 ± 0.5 |
| Cels_m         | 91.9 ± 1.0 | 75.9 ± 1.4 | 73.4 ± 1.2 | 92.7 ± 0.7 | 96.8 ± 0.5 |
| CVC            | 89.6 ± 1.2 | 76.9 ± 1.1 | 75.2 ± 1.5 | 90.9 ± 0.7 | 96.5 ± 0.5 |
| Decorrelated   | 90.7 ± 1.0 | 76.0 ± 1.1 | 73.9 ± 1.3 | 92.8 ± 0.7 | 97.0 ± 0.5 |
| Decorrelated2  | 90.4 ± 1.0 | 76.0 ± 1.0 | 73.8 ± 1.3 | 92.6 ± 0.5 | 97.0 ± 0.5 |
| Evol           | 77 ± 3     | 66.1 ± 0.7 | 74.8 ± 0.7 | 61 ± 4     | 87.2 ± 1.6 |
| Ola            | 90.9 ± 1.7 | 73.8 ± 0.8 | 74.8 ± 0.8 | 88.1 ± 0.8 | 95.5 ± 0.6 |
| CVC version 2  | 89.7 ± 1.4 | 76.8 ± 1.0 | 74.1 ± 1.2 | 89.8 ± 0.9 | 96.7 ± 0.3 |
| AveBoost       | 89.4 ± 1.3 | 76.5 ± 1.1 | 75.1 ± 1.2 | 88.1 ± 1.0 | 95.6 ± 0.5 |
| TCA            | 87.9 ± 1.2 | 75.4 ± 0.8 | 73.0 ± 1.5 | 87.5 ± 1.1 | 91 ± 4     |
| ArcX4          | 89.4 ± 1.0 | 76.0 ± 0.8 | 68 ± 2     | 90.8 ± 0.9 | 96.3 ± 0.6 |
| ArcX4 Voting   | 91.3 ± 1.0 | 76.3 ± 0.8 | 73.9 ± 1.0 | 94.6 ± 0.9 | 96.6 ± 0.6 |
| Aggressive B   | 90.3 ± 0.9 | 74.3 ± 1.5 | 73.8 ± 1.5 | 86.9 ± 1.2 | 96.6 ± 0.6 |
| Conservative B | 89.4 ± 1.0 | 75.6 ± 1.2 | 75.6 ± 1.1 | 88.8 ± 1.1 | 97.0 ± 0.6 |
| EENCL UG       | 93.0 ± 1.0 | 74.7 ± 1.0 | 73.9 ± 1.2 | 87.2 ± 0.8 | 96.2 ± 0.4 |
| EENCL MG       | 93.7 ± 0.9 | 75.3 ± 1.0 | 73.9 ± 0.8 | 87.4 ± 0.7 | 96.4 ± 0.5 |
| Simple Ens     | 90.3 ± 1.1 | 75.9 ± 1.2 | 74.2 ± 1.3 | 91.0 ± 0.5 | 96.9 ± 0.5 |

The value of the percentage of error reduction ranges from 0%, where there is no improvement by the use of a particular ensemble method with respect to a single network, to 100%. There can also be negative values, which means that the performance of the ensemble is worse than the performance of the single network.

**Table 3.** Results for the Ensemble of twenty networks

|                | ARITM      | DERMA      | ECOLI      | FLARE      | IMAGEN       |
|----------------|------------|------------|------------|------------|--------------|
| Adaboost       | 71.4 ± 1.5 | 97.5 ± 0.6 | 86.0 ± 1.3 | 81.1 ± 0.8 | 97.29 ± 0.19 |
| Bagging        | 75.9 ± 1.7 | 97.6 ± 0.6 | 87.1 ± 1.0 | 82.2 ± 0.5 | 97.0 ± 0.3   |
| Bag_Noise      | 76.0 ± 1.1 | 97.3 ± 0.6 | 87.4 ± 0.8 | 82.1 ± 0.5 | 93.3 ± 0.3   |
| Cels_m         | 75.4 ± 1.2 | 93.9 ± 1.4 | 86.3 ± 1.3 | 81.5 ± 0.4 | 95.74 ± 0.19 |
| CVC            | 74.8 ± 1.3 | 97.3 ± 0.6 | 86.5 ± 1.0 | 81.7 ± 0.7 | 96.8 ± 0.2   |
| Decorrelated   | 76.1 ± 1.1 | 97.6 ± 0.7 | 87.1 ± 0.7 | 81.3 ± 0.5 | 96.9 ± 0.2   |
| Decorrelated2  | 73.9 ± 1.1 | 97.6 ± 0.7 | 88.1 ± 0.7 | 81.6 ± 0.5 | 96.8 ± 0.2   |
| Evol           | 65.9 ± 1.9 | 47 ± 5     | 55 ± 4     | 81.2 ± 0.5 | 63 ± 5       |
| Ola            | 72.5 ± 1.1 | 87.0 ± 1.4 | 84.3 ± 1.2 | 80.7 ± 0.4 | 96.4 ± 0.2   |
| CVC version 2  | 74.3 ± 1.2 | 97.5 ± 0.6 | 86.6 ± 1.1 | 81.8 ± 0.4 | 97.03 ± 0.17 |
| AveBoost       | 75.5 ± 1.1 | 97.9 ± 0.5 | 86.2 ± 1.2 | 82.4 ± 0.7 | 97.3 ± 0.3   |
| TCA            | 71.6 ± 1.8 | 92 ± 2     | 85.4 ± 1.5 | 79.7 ± 0.9 | 95.7 ± 0.3   |
| ArcX4          | 74.4 ± 1.4 | 97.8 ± 0.6 | 85.6 ± 0.8 | 78.4 ± 1.4 | 97.38 ± 0.19 |
| ArcX4 Voting   | 75.1 ± 1.4 | 97.3 ± 0.6 | 86.0 ± 0.8 | 78.6 ± 1.4 | 97.3 ± 0.2   |
| Aggressive B   | 74.8 ± 1.5 | 97.0 ± 0.6 | 87.1 ± 1.1 | 82.0 ± 0.5 | 97.2 ± 0.3   |
| Conservative B | 74.7 ± 0.9 | 97.9 ± 0.6 | 86.9 ± 1.2 | 82.8 ± 0.6 | 97.2 ± 0.3   |
| EENCL UG       | 72.9 ± 0.9 | 95.1 ± 1.1 | 87.2 ± 0.7 | 82.0 ± 0.8 | 96.9 ± 0.3   |
| EENCL MG       | 73.5 ± 1.6 | 96.2 ± 0.9 | 87.7 ± 1.0 | 81.4 ± 0.6 | 96.6 ± 0.3   |
| Simple Ens     | 73.8 ± 1.1 | 97.3 ± 0.7 | 86.9 ± 0.8 | 81.5 ± 0.5 | 96.7 ± 0.2   |

**Table 3 (continuation).** Results for the ensemble of twenty networks

|                | IONOS      | PIMA       | SURVI      | VOWEL      | WDBC       |
|----------------|------------|------------|------------|------------|------------|
| Adaboost       | 91.4 ± 0.8 | 74.8 ± 1.0 | 74.3 ± 1.5 | 96.1 ± 0.7 | 96.3 ± 0.5 |
| Bagging        | 89.6 ± 1.1 | 77.0 ± 1.0 | 74.6 ± 1.7 | 91.3 ± 0.6 | 97.5 ± 0.4 |
| Bag_Noise      | 92.7 ± 0.6 | 76.3 ± 0.8 | 74.6 ± 0.7 | 86.7 ± 0.7 | 96.1 ± 0.5 |
| Cels_m         | 93.3 ± 0.7 | 75.4 ± 1.0 | 64 ± 3     | 87.5 ± 0.8 | 96.5 ± 0.5 |
| CVC            | 89.6 ± 1.3 | 76.2 ± 1.3 | 73.8 ± 0.9 | 91.9 ± 0.5 | 97.4 ± 0.4 |
| Decorrelated   | 91.1 ± 0.9 | 76.1 ± 1.0 | 74.1 ± 1.4 | 93.3 ± 0.6 | 97.0 ± 0.5 |
| Decorrelated2  | 90.9 ± 0.9 | 76.1 ± 1.0 | 74.3 ± 1.3 | 93.3 ± 0.5 | 97.0 ± 0.5 |
| Evol           | 66.1 ± 1.2 | 65.2 ± 0.9 | 74.8 ± 0.7 | 60 ± 3     | 78 ± 3     |
| Ola            | 69.4 ± 1.2 | 74.2 ± 1.1 | 74.1 ± 0.7 | 88.7 ± 0.8 | 95.3 ± 0.6 |
| CVC version 2  | 91.0 ± 0.9 | 76.7 ± 0.8 | 73.6 ± 1.0 | 93.3 ± 0.6 | 95.9 ± 0.6 |
| AveBoost       | 91.4 ± 1.0 | 76.0 ± 1.1 | 74.8 ± 1.2 | 95.8 ± 0.6 | 95.8 ± 0.6 |
| TCA            | 86.1 ± 1.0 | 73.5 ± 0.9 | 71.3 ± 1.8 | 85 ± 3     | 94.4 ± 0.7 |
| ArcX4          | 92.0 ± 0.9 | 72.7 ± 1.1 | 69 ± 2     | 96.6 ± 0.5 | 96.4 ± 0.6 |
| ArcX4 Voting   | 92.6 ± 0.9 | 75.0 ± 1.1 | 73.8 ± 1.9 | 96.1 ± 0.5 | 96.6 ± 0.6 |
| Aggressive B   | 91.6 ± 0.9 | 75.5 ± 1.3 | 73.9 ± 1.7 | 96.9 ± 0.6 | 96.8 ± 0.6 |
| Conservative B | 92.4 ± 1.0 | 76.7 ± 1.2 | 72.8 ± 1.3 | 96.6 ± 0.6 | 96.4 ± 0.6 |
| EENCL UG       | 92.3 ± 1.1 | 75.2 ± 0.8 | 72.5 ± 1.5 | 88.2 ± 0.9 | 95.8 ± 0.4 |
| EENCL MG       | 92.3 ± 1.0 | 76.2 ± 1.3 | 74.1 ± 1.0 | 88.3 ± 0.9 | 96.5 ± 0.4 |
| Simple Ens     | 90.4 ± 1.0 | 75.9 ± 1.2 | 74.3 ± 1.3 | 91.4 ± 0.8 | 96.9 ± 0.5 |

This new measurement is relative and can be used to compare more clearly the different methods. Furthermore we can calculate the mean performance of error reduction across all databases this value is in table 4 for ensembles of 3, 9, 20 and 40 networks.

According to this global measurement *Ola*, *Evol* and *BagNoise* performs worse than the *Simple Ensemble*. The best methods are *Bagging*, *Cels*, *Decorrelated*, *Decorrelated2* and *Conservative Boosting*. In total, there are around ten methods which perform better than the *Simple Ensemble*.

The best method for 3 networks in the ensemble is *Decorrelated*, the best method for the case of 9 and 20 networks is *Conservative Boosting* and the best method for the case of 40 networks is *Cels* but the performance of *Conservative Boosting* is also good.



**Table 4.** Mean percentage of error reduction for the different ensembles

|                | Ensemble 3 Nets | Ensemble 9 Nets | Ensemble 20 Nets | Ensemble 40 Nets |
|----------------|-----------------|-----------------|------------------|------------------|
| Adaboost       | 1.33            | 4.26            | 9.38             | 12.21            |
| Bagging        | 6.86            | 12.12           | 13.36            | 12.63            |
| Bag_Noise      | -3.08           | -5.08           | -3.26            | -3.05            |
| Boosting       | -0.67           |                 |                  |                  |
| Cels_m         | 9.98            | 9.18            | 10.86            | 14.43            |
| CVC            | 6.18            | 7.76            | 10.12            | 6.48             |
| Decorrelated   | 9.34            | 12.09           | 12.61            | 12.35            |
| Decorrelated2  | 9.09            | 11.06           | 12.16            | 12.10            |
| Evol           | -218.23         | -297.01         | -375.36          | -404.81          |
| Ola            | -33.11          | -36.43          | -52.53           | -47.39           |
| CVC version 2  | 10.25           | 10.02           | 7.57             | 7.49             |
| AveBoost       | 1.13            | 10.46           | 9.38             | 10.79            |
| TCA            | -9.71           | -25.22          | -43.98           | -53.65           |
| ArcX4          | 1.21            | 2.85            | 7.85             | 10.05            |
| ArcX4 Voting   | -2.08           | 9.73            | 10.76            | 11.14            |
| Aggressive B   | 1.22            | 7.34            | 13.03            | 13.54            |
| Conservative B | 4.45            | 13.07           | 14.8             | 14.11            |
| EENCL UG       | 0.21            | -3.23           | -3.59            | 1.10             |
| EENCL MG       | 3.96            | 1.52            | 2.84             | 7.89             |
| Simple Ens     | 5.89            | 8.39            | 8.09             | 9.72             |

So, we can conclude that if the number of networks is low it seems that the best method is *Decorrelated* and if the number of network is high the best method is in general *Conservative Boosting*.

Also in table 4, we can see the effect of increasing the number of networks in the ensemble. There are several methods (*Adaboost*, *Cels*, *ArcX4*, *ArcX4 Voting*, *Aggressive Boosting* and *Conservative Boosting*) where the performance seems to increase slightly with the number of networks in the ensemble. But other methods like *Bagging*, *CVC*, *Decorrelated*, *Decorrelated2* and *Simple Ensemble* does not increase the performance beyond 9 or 20 networks in the ensemble. The reason can be that the new networks are correlated to the first ones or that the combination method (the average) does not exploit well the increase in the number of networks.

## 4 Conclusions

In this paper we have presented experimental results of twenty different methods to construct an ensemble of networks, using ten different databases. We trained ensembles of 3, 9, 20 and 40 networks in the ensemble to cover a wide spectrum of number of networks in the ensemble. The results showed that in general the improvement by the use of the ensemble methods depends clearly on the database, in some databases there is an improvement but in other there is not improvement at all. Also the improvement in performance from three or nine networks in the ensemble to a higher number of networks depends on the method. Taking into account the computational cost, an ensemble of nine networks may be the best alternative for most of the methods. Finally, we have obtained the mean percentage of error reduction over all databases. According to the results of this measurement the best methods are *Bagging*, *Cels*, *Decorrelated*, *Decorrelated2* and *Conservative Boosting*. In total, there are around ten methods which perform better than the *Simple Ensemble*. The best method

for 3 networks in the ensemble is *Decorrelated*, the best method for the case of 9 and 20 networks is *Conservative Boosting* and the best method for the case of 40 networks is *Cels* but the performance of *Conservative Boosting* is also good. So we can conclude that if the number of networks is low it seems that the best method is *Decorrelated* and if the number of network is high the best method is in general *Conservative Boosting*.

## References

1. Tumer, K., Ghosh, J., "Error correlation and error reduction in ensemble classifiers", *Connection Science*, vol. 8, nos. 3 & 4, pp. 385-404, 1996.
2. Raviv, Y., Intrator, N., "Bootstrapping with Noise: An Effective Regularization Technique", *Connection Science*, vol. 8, no. 3 & 4, pp. 355-372, 1996.
3. Drucker, H., Cortes, C., Jackel, D., et alt., "Boosting and Other Ensemble Methods", *Neural Computation*, vol. 6, pp. 1289-1301, 1994.
4. Fernandez-Redondo, Mercedes, Hernández-Espinosa, Carlos, Torres-Sospedra, Joaquín, "Classification by Multilayer Feedforward ensambles", *Internacional Symposium on Neural Networks, Lecture Notes in Computer Science*, Vol. 3173, pp. 852-857, 2004.
5. Verikas, A., Lipnickas, A., Malmqvist, K., Bacauskiene, M., Gelzinis, A., "Soft Combination of neural classifiers: A comparative study", *Pattern Recognition Letters*, Vol. 20, pp 429-444, 1999.
6. Oza, N.C., "Boosting with Averaged Weight Vectors", *Multiple Classifier Systems, Lecture Notes in Computer Science*, vol. 2709, pp. 15-24, 2003.
7. Kuncheva, L.I., "Error Bounds for Aggressive and Conservative Adaboost", *Multiple Classifier Systems, Lecture Notes in Computer Science*, vol. 2709, pp. 25-34, 2003.
8. Breiman, L., "Arcing Classifiers", *Annals of Statistic*, vol. 26, no. 3, pp. 801-849, 1998.
9. Liu, Y., Yao, X., Higuchi, T., "Evolutionary Ensembles with Negative Correlation Learning", *IEEE Trans. On Evolutionary Computation*, vol. 4, no. 4, pp. 380-387, 2000.

# Layered Network Computations by Parallel Nonlinear Processing

Naohiro Ishii<sup>1</sup>, Toshinori Deguchi<sup>2</sup>, and Hiroshi Sasaki<sup>3</sup>

<sup>1</sup> Aichi Institute of Technology, Yachigusa Yakusacho 1247, Toyota, 470-0392, Japan  
ishii@aitech.ac.jp

<sup>2</sup> Gifu National College of Technology, Motosu, Gifu, 501-0495, Japan  
deguchi@gifu-nct.ac.jp

<sup>3</sup> Fukui University of Technology, Fukui, 910-8505, Japan  
h-sasaki@ccmails.fukui-ut.ac.jp

**Abstract.** Among visual processings in the visual networks, movement detections are carried out in the visual cortex. The visual cortex for the movement detection, consist of two layered networks, called the primary visual cortex (V1), followed by the middle temporal area (MT). In the biological visual neural networks, a characteristic feature is nonlinear functions, which will play important roles in the visual systems. In this paper, V1 and MT model networks, are decomposed into sub-asymmetrical networks. By the optimization of the asymmetric networks, movement detection equations are derived. Then, it was clarified that asymmetric networks with the even-odd nonlinearity combined, are fundamental in the movement detection. These facts are applied to two layered V1 and MT networks, in which it was clarified that the second layer MT has an efficient ability to detect the movement.

## 1 Introduction

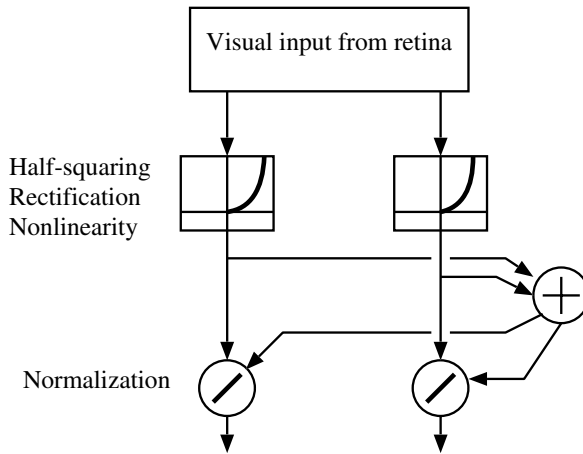
The visual information is processed firstly in the retina, next the thalamus on the way and the cortex, which are on the final visual pathway of the biological networks. It is questioned that what kinds of functions are realized in the structures of the networks. What are the relations between the structures and the functions in the neural networks? Among characteristics of the visual networks, movement detections are carried out in the visual cortex. The visual cortex for the movement detection, consist of two layered networks, called the primary visual cortex area (V1), followed by the middle temporal area (MT). The computational model of networks in visual cortex V1 and MT, was developed by Simoncelli and Heeger[15, 16]. The model networks in V1 and MT, are identical in their structure. In this paper, first we start from the network model developed by Simoncelli and Heeger[16], which shows a linear receptive field, followed by half-aquaring rectification and normalization in V1 and next V1 afferents, followed by half-squaring rectification and normalization in MT. Then, the half-squaring nonlinearity and normalization (saturation) is analyzed by the

approximation of Taylor series. Thus, the model is transformed in the parallel network structures, decomposed into asymmetric sub-networks.

Next, we will discuss the relations between the functions and the structure in the asymmetric networks with nonlinear functions, which are fundamental in the parallel networks expanded in V1 and MT model. The asymmetric network with nonlinear functions, are clarified in retinal networks by Naka, et al[9]. By the optimization conditions on the asymmetric networks, movement detection equations are derived. Then, it was clarified that the even-odd nonlinearity combined asymmetric networks, are fundamental in the movement detection.

## 2 Parallel Processing by Neural Networks

Simoncelli and Heeger presented a model of the cortical area V1 and the middle temporal area MT, which detect the movement stimulus in the vertebrate brain[16]. Lu and Sperling showed the human visual motion perception model, in which the half-wave rectification plays an significant role in their mechanisms[17]. The computational model for the visual motion, consists of two primary stages corresponding to cortical areas V1 and MT, which is developed by Simoncelli and Heeger. The network model is shown in Fig.1.



**Fig. 1.** Processing Model in Cortical Area V1

In the cortical area V1 and also the middle temporal area MT, the movement detection is carried out. Model in these areas, is shown in Fig.1 (Simoncelli et al). In Fig.1, the half-wave squaring rectification followed by the normalization which includes saturation, is approximated by the nonlinear sigmoid function in the following,

$$f(x) = \frac{1}{1 + e^{-\eta(x-\theta)}} \quad (1)$$

By Taylor expansion of the equation (1) at  $x = \theta$ , the equation (2) is derived as follows,

$$\begin{aligned}
 f(x)_{x=\theta} &= f(\theta) + f'(\theta)(x - \theta) + \frac{1}{2!}f''(\theta)(x - \theta)^2 + \dots \\
 &= \frac{1}{2} + \frac{\eta}{4}(x - \theta) + \frac{1}{2!} \left( -\frac{\eta^2}{4} + \frac{\eta^2 e^{\eta\theta}}{2} \right) (x - \theta)^2 \dots
 \end{aligned}
 \tag{2}$$

In the equation (2), a sigmoid function is adopted as the approximation of the half-wave squaring nonlinearity, which is followed by the normalization operation as shown in Fig.1. In the Taylor series of the sigmoid function, the parameter  $\eta$  becomes to be large as  $\eta \geq 8$  and  $\theta \simeq 0.5$  by max of  $f(x) = 1$  in the equation (2). Thus, the first order  $(x - \theta)$  term  $\eta/4$  exists and the second order term  $(x - \theta)^2$  also exists by the relation  $\eta\theta \gg \log_e 2$ .

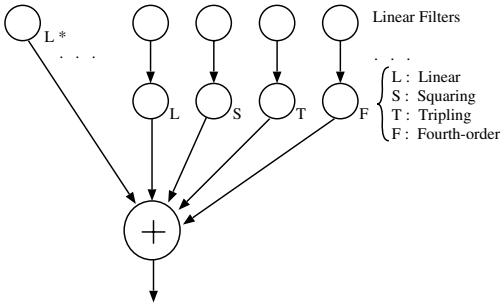


Fig. 2. Network Decomposed Model

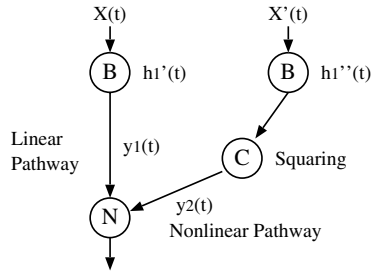


Fig. 3. Asymmetric Neural Network

Fig.2 shows parallel-pathways model with half-wave squaring rectification, which is approximated by the nonlinear higher order terms. The linear term  $L^*$  on the left pathway is represented, while the linear term  $L$  on the right pathway, the squaring nonlinearity  $S$ , which is the second order nonlinearity on it, the tripling nonlinearity  $T$ , which is the third order nonlinearity on it, and the fourth order nonlinearity on it, are represented.

To analyze functions of the network model in Fig.2, the biological network in the catfish retina was clarified by Naka[7, 9, 12]. Naka[4, 9] presented a simplified, but essential networks of catfish inner retina, which has an asymmetric structure of the neural networks. A biological network of catfish retina is shown in Fig.3, which might process the spatial interactive information between bipolar cells  $B_1$  and  $B_2$ . The bipolar  $B$  cell response is linearly related to the input modulation of light. The  $C$  cell shows an amacrine cell, which plays an important roll in the nonlinear function as squaring of the output of the bipolar cell  $B_2$ .

The  $N$  amacrine cell was clarified to be time-varying and differential with band-pass characteristics in the function. It is shown that  $N$  cell response is realized by a linear filter, which is composed of a differentiation filter followed

by a low-pass filter. Thus the asymmetric network in Fig.3 is composed of a linear pathway and a nonlinear pathway.

### 3 Asymmetric Neural Network with Quadratic Nonlinearity

Movement perception is carried out firstly in the retinal neural network. Fig.4 shows a schematic diagram of a motion problem in front of the asymmetric network in Fig.3.

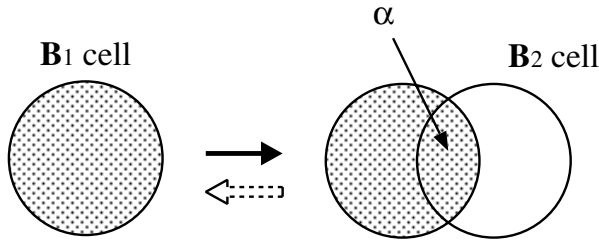


Fig. 4. Schematic diagram of motion problem for spatial interaction

We assume here the input functions  $x(t)$  and  $x''(t)$  to be Gaussian white noise, whose mean values are zero, but their deviations are different in their values. In Fig.4, moving stimulus shows that  $x(t)$  merges into  $x''(t)$ , thus  $x''(t)$  is mixed with  $x(t)$ . Then, we indicate the right stimulus by  $x'(t)$ . By introducing a mixed ratio,  $\alpha$ , the input function of the right stimulus, is described in the following equation, where  $0 \leq \alpha \leq 1$  and  $\beta = 1 - \alpha$  hold. Then, Fig.4 shows that the moving stimulus is described in the following equation,

$$x'(t) = \alpha x(t) + \beta x''(t) \tag{3}$$

Let the power spectrums of  $x(t)$  and  $x''(t)$ , be  $p$  and  $p''$ , respectively and an equation  $p = kp''$  holds for the coefficient  $k$ , because we assumed here that the deviations of the input functions are different in their values.

First, on the linear pathway of the asymmetrical network in Fig.3, the input function is  $x(t)$  and the output function is  $y(t)$ , which is an output after the linear filter of cell N.

$$y(t) = \int h_1'''(\tau)(y_1(t - \tau) + y_2(t - \tau))d\tau + \varepsilon \tag{4}$$

where  $y_1(t)$  shows the linear information on the linear pathway,  $y_2(t)$  shows the nonlinear information on the nonlinear pathway and  $\varepsilon$  shows error value. The  $y_1(t)$  and  $y_2(t)$  are given, respectively as follows,

$$y_1(t) = \int_0^\infty h_1'(\tau)x(t - \tau)d\tau \tag{5}$$

$$y_2(t) = \int_0^\infty \int_0^\infty h_1''(\tau_1)h_1''(\tau_2)x'(t - \tau_1)x'(t - \tau_2)d\tau_1d\tau_2 \tag{6}$$

We assume here the linear filter **N** to have only summation operation without in the analysis. Thus the impulse response function  $h_1'''(t)$  is assumed to be value 1 without loss of generality.

### 3.1 Optimization of Asymmetric Neural Networks

Under the assumption that the impulse response functions,  $h_1'(t)$  of the cell **B**<sub>1</sub>,  $h_1''(t)$  of the cell **B**<sub>2</sub> and moving stimulus ratio  $\alpha$  in the right to be unknown, the optimization of the network is carried out. By the minimization of the mean squared value  $\xi$  of  $\varepsilon$  in equation (4), the following necessary equations for the optimization of equations (5) and (6), are derived,

$$\frac{\partial \xi}{\partial h_1'(t)} = 0, \quad \frac{\partial \xi}{\partial h_2''(t)} = 0 \quad \text{and} \quad \frac{\partial \xi}{\partial \alpha} = 0 \tag{7}$$

Then, the following three equations are derived as the conditions for the optimization in the equation (7).

$$\begin{aligned} E[y(t)x(t - \lambda)] &= h_1'(\lambda)p \\ E[(y(t) - C_0)x(t - \lambda_1)x(t - \lambda_2)] &= 2p^2\alpha^2h_1''(\lambda_1)h_1''(\lambda_2) \\ E[(y(t) - C_0)x'(t - \lambda_1)x'(t - \lambda_2)] &= 2p'^2h_1''(\lambda_1)h_1''(\lambda_2) \end{aligned} \tag{8}$$

where  $C_0$  is the mean value of  $y(t)$ , which is shown in the following. Here, the equations (9) can be rewritten by applying Wiener kernels, which are related with input and out put correlations by Lee and Schetzen's method[8].

First, we can compute the 0-th order Wiener kernel  $C_0$ , the 1-st order one  $C_{11}(\lambda)$ , and the 2-nd order one  $C_{21}(\lambda_1, \lambda_2)$  on the linear pathway by the cross-correlations between  $x(t)$  and  $y(t)$ . The suffix  $i, j$  of the kernel  $C_{ij}(\cdot)$ , shows that  $i$  is the order of the kernel and  $j = 1$  means the linear pathway, while  $j = 2$  means the nonlinear pathway. Then, the 0-th order kernel under the condition of the spatial interaction of cell's impulse response functions  $h_1'(t)$  and  $h_1''(t)$ , becomes

$$\begin{aligned} C_0 &= E[y(t)] \\ &= p(\alpha^2 + k\beta^2) \int_0^\infty (h_1''(\tau_1))^2 d\tau_1 \end{aligned} \tag{9}$$

The 1-st order kernel is defined by the correlation between  $y(t)$  and  $x(t)$  as on the linear pathway, and the correlation value is derived from the optimization condition of equation (9). Then, the following 1-st order kernel is shown.

$$C_{11}(\lambda) = \frac{1}{p}E[y(t)x(t - \tau)] = h_1'(\lambda) \tag{10}$$

The 2-nd order kernel is defined and it becomes from the optimization condition of equation (9), in the following,

$$\begin{aligned}
 C_{21}(\lambda_1, \lambda_2) &= \frac{1}{2p^2} E[(y(t) - C_0)x(t - \lambda_1)x(t - \lambda_2)] \\
 &= \alpha^2 h_1''(\lambda_1)h_1''(\lambda_2)
 \end{aligned}
 \tag{11}$$

From equations (3), (10) and (11), the ratio  $\alpha$  is a mixed coefficient of  $x(t)$  to  $x'(t)$ , is shown by  $\alpha^2$  as the amplitude of the second order Wiener kernel. On the nonlinear pathway, the 1-st order kernel  $C_{12}(\lambda)$  and the 2-nd order kernel  $C_{22}(\lambda_1, \lambda_2)$  is defined by the cross-correlations between  $x(t)$  and  $y(t)$  as shown in the following. Here,  $C_{22}(\lambda_1, \lambda_2)$  is computed from the optimization condition of equation (9), while  $C_{12}(\lambda)$  is derived as an additional condition from  $C_{11}(\lambda)$ .

$$\begin{aligned}
 C_{12}(\lambda) &= \frac{1}{p(\alpha^2 + k\beta^2)} E[y(t)x'(t - \lambda)] \\
 &= \frac{\alpha}{\alpha^2 + k(1 - \alpha)^2} h_1'(\lambda)
 \end{aligned}
 \tag{12}$$

and

$$C_{22}(\lambda_1, \lambda_2) = h_1''(\lambda_1)h_1''(\lambda_2)
 \tag{13}$$

The motion problem is how to detect the movement in the increase of the ratio  $\alpha$  in Fig.4. This implies that for the motion of the light from the left side circle to the right one, the ratio  $\alpha$  can be derived from the kernels described in the above, in which the second order kernels  $C_{21}$  and  $C_{22}$  are abbreviated in the representation of equations (11) and (13). Then, the following

$$(C_{21}/C_{22}) = \alpha^2
 \tag{14}$$

holds. From the equation (14) the ratio  $\alpha$  is shown as follows.

$$\alpha = \sqrt{\frac{C_{21}}{C_{22}}}
 \tag{15}$$

The equation (15) is called here  $\alpha$ -equation, which implies the change of the movement stimulus on the network and shows the detection of the movement change by the  $\alpha$ . The equation (15) does not show the direction of the movement. Some measure to show the direction of the movement, is needed. We will discuss how to detect the direction of the movement in the following.

From the first order kernels  $C_{11}$  and  $C_{12}$ , and the second order kernels in the above discussion, the following movement equation is derived,

$$\frac{C_{12}}{C_{11}} = \frac{\sqrt{\frac{C_{21}}{C_{22}}}}{\frac{C_{21}}{C_{22}} + k \left(1 - \sqrt{\frac{C_{21}}{C_{22}}}\right)^2}
 \tag{16}$$



where  $k$  in the equation shows the difference of the power of the stimulus between the receptive fields of the left and the right cells  $\mathbf{B}_1$  and  $\mathbf{B}_2$ .

In Fig.4, the left side stimulus moves from the receptive field of the cell  $\mathbf{B}_1$  to the receptive field of the right cell  $\mathbf{B}_2$ , gradually, in bold line, while the movement of the stimulus is shifted from the right to the left as shown in dotted line. Then, the different equations from the equations (15) and (16), are similarly derived, which shows that the different equations exist for the stimulus movement directions[18].

### 3.2 Asymmetric Networks with Odd-Even Orders Nonlinearities

Asymmetric network with 1-st and 4-th orders nonlinearities, are considered. The asymmetric network consists of the 1-st order nonlinearity (odd order nonlinearity) on the left pathway and the 4-th order (even order nonlinearity) on the right pathway. On the left (1-st order nonlinearity),

$$\begin{aligned}
 C_{11}(\lambda) &= h'_1(\lambda) \\
 C_{21}(\lambda_1, \lambda_2) &= 6\alpha^2(\alpha^2 + k\beta^2)ph''_1(\lambda_1)h''_1(\lambda_2) \int_0^\infty h''_1(\tau)d\tau \tag{17}
 \end{aligned}$$

On the right (4-th order nonlinearity),

$$\begin{aligned}
 C_{12}(\lambda) &= \frac{1}{(\alpha^2 + k\beta^2)}h'_1(\lambda) \\
 C_{22}(\lambda_1, \lambda_2) &= 6(\alpha^2 + k\beta^2)ph''_1(\lambda_1)h''_1(\lambda_2) \int_0^\infty h''_1(\tau)d\tau \tag{18}
 \end{aligned}$$

From equations (17) and (18), the  $\alpha$ -equation and the movement equation, are derived as follows. The the  $\alpha$ -equation becomes

$$C_{21}(\lambda_1, \lambda_2) = \alpha^2 C_{22}(\lambda_1, \lambda_2) \quad \therefore \alpha = \sqrt{\frac{C_{21}}{C_{22}}} \tag{19}$$

The movement equation becomes

$$C_{12}(\lambda) = \frac{\alpha}{(\alpha^2 + k\beta^2)}C_{11}(\lambda) \quad \therefore \frac{C_{12}}{C_{11}} = \frac{\sqrt{\frac{C_{21}}{C_{22}}}}{\frac{C_{21}}{C_{22}} + k \left(1 - \sqrt{\frac{C_{21}}{C_{22}}}\right)^2} \tag{20}$$

The  $\alpha$ -equation (19) and the movement equation (20), are same to those of (15) and (16), respectively, in the asymmetric network with 1-st and 2-nd orders nonlinearities.

The 3-rd order kernels, become

$$C_{31}(\lambda_1, \lambda_2, \lambda_3) = 0, \quad C_{32}(\lambda_1, \lambda_2, \lambda_3) = 0$$

The 4-th order kernels, become

$$\begin{aligned} C_{41}(\lambda_1, \lambda_2, \lambda_3, \lambda_4) &= \alpha^4 h_1''(\lambda_1) h_1''(\lambda_2) h_1''(\lambda_3) h_1''(\lambda_4) \\ C_{42}(\lambda_1, \lambda_2, \lambda_3, \lambda_4) &= h_1''(\lambda_1) h_1''(\lambda_2) h_1''(\lambda_3) h_1''(\lambda_4) \end{aligned} \tag{21}$$

From the equation (21), the  $\alpha$ -equation is derived as follows, which is equivalent to the equation (19).

$$\therefore \alpha = \left\{ \frac{C_{41}}{C_{42}} \right\} = \left( \sqrt{\frac{C_{21}}{C_{22}}} \right)$$

Thus, the asymmetric network with 1-st and 4-th orders nonlinearities, are equivalent to that with 1-st and 2-nd orders nonlinearities. Similarly, it is shown the asymmetric networks with the even order on the one side pathway and with another even order on the other side pathway, cannot detect the direction of the movement. Those network characteristics are generalized as follows: the asymmetric network with the odd order nonlinearity on the one pathway and the even order nonlinearity on the other pathway, has both  $\alpha$ -equation and the directional movement equation.

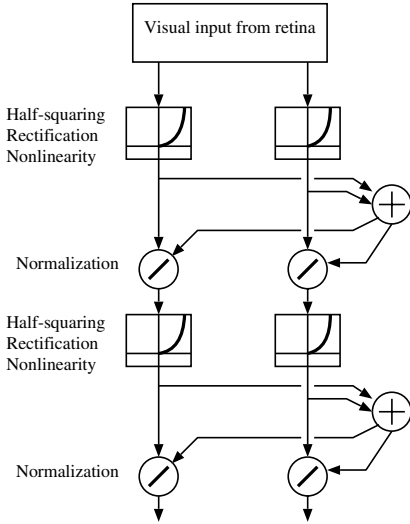
## 4 Layered Model by Decomposed Asymmetric Networks

Heeger and Simoncelli presented a network model with half-wave squaring rectification nonlinearity in V1 and MT cortex areas in the brain[14, 15, 16]. In this paper, this parallel processing model is interpreted analytically to have the tracking function of the moving stimulus, which is based on the asymmetrical nonlinear networks described in the above.

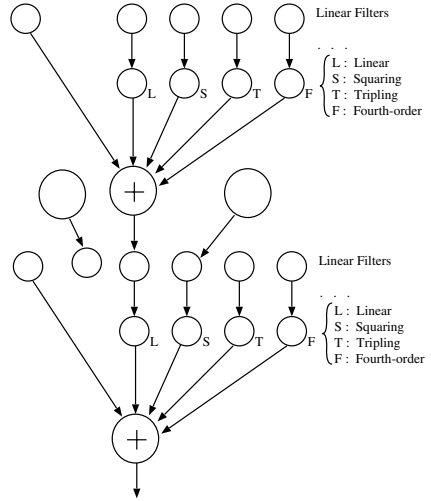
Fig.5 shows a layered network model of cortex areas V1 followed by MT, which implies the same network model in Fig.1. The sigmoid function is represented by Taylor series as shown in the section 2, which includes the 1-st, the 2-nd, the 3-rd, the 4-th ... and higher orders nonlinearity terms. Thus, the network model in Fig.5, is transformed to that of layered decomposed asymmetric model as shown in Fig.6. The function of the layered decomposed networks in Fig.6, is based on the asymmetric network with nonlinearities in the section 3. In V1 decomposed network in Fig.6, the left side linear pathway (1-st order nonlinearity) and the right side nonlinear pathway, computes the  $\alpha$ -equation (detection of stimulus) and the movement equation (detection of the moving direction).

Fig.6 shows that only the first layer V1, is weak in the detection of the stimulus compared to the second layer MT, since these odd order nonlinearities in V1, has transformed to the even order nonlinearities (2-nd, 6-th, 10-th ... orders) in

the second layer MT. This shows that the second layer MT, has both efficient abilities of the detection of the stimulus and the direction of the stimulus. It is proved that the  $\alpha$ -equation and the movement equation are not derived in the



**Fig. 5.** Layered model of V1 followed by MT



**Fig. 6.** Layered decomposed networks of Fig.5

symmetric network with the even order nonlinearity on the pathway and the other even nonlinearity on the another pathway. Both the even (or odd) nonlinearities on the parallel pathways, do not have efficient works on the movement detection.

## 5 Conclusion

In this paper, the neural networks in the cortical areas V1 and MT, are discussed to make clear the function of their movement detection ability. Then, the networks are decomposed to the asymmetric networks with nonlinear higher order terms. The ability is described by the moving detection equation ( $\alpha$ -equation) and the movement direction equation. It was clarified that the asymmetric network with the even and the odd nonlinearities, has efficient ability in the movement detection. Based on the asymmetric network principle, it is shown that the parallel-symmetric network with half-squaring rectification of V1 and MT, has efficient ability in the movement detection.

## References

1. Hassenstein, B. and Reichard, W., "Systemtheoretische analyse der zeit-, reihenfolgen- and vorzeichenauswertung bei der bewegungsperzeption des rasselkafers," *Chlorophanus. Z. Naturf*, 11b, pp.513–524, 1956.
2. Barlow, H.B. and Levick, R. W., "The mechanism of directional selectivity in the rabbit's retina," *J. Physiol.* 173: pp.377–407, 1965
3. Victor J. D. and Shapley K. M., "The nonlinear pathway of Y ganglion cells in the cat retina," *J. Gen. Physiol.*, vol.74, pp.671–689, 1979.
4. Ishii, N. and Naka, K.-I., "Movement and Memory Function in Biological Neural Networks," *Int. J. of Artificial Intelligence Tools*, vol.4, No.4, pp.489–500, 1995.
5. Ishii, N., Sugiura, S., Nakamura, M., Yamauchi, "Sensory Perception, Learning and Integration in Neural Networks," *Proc. IEEE Int. Conf. on Information Intelligence & Systems*, pp.72–79, 1999.
6. Ishii, N. and Naka, K.-I., "Function of Biological Asymmetrical Neural Networks," *Biological and Artificial Computation: From Neuroscience to Technology*, LNCS vol.1240 Springer, pp.1115–1125 , 1997.
7. Korenberg, M. J., Sakai, H. M. and Naka, K.-I., "Dissection of the neuron network in the catfish inner retina," *J. Neurophysiol.* 61: pp.1110–1120, 1989.
8. Marmarelis, P. Z. and Marmarelis, V. Z., *Analysis of Physiological System: The White Noise Approach*, New York: Plenum Press, 1978.
9. Naka, K.-I., Sakai, H. M. and Ishii, N., "Generation of transformation of second order nonlinearity in catfish retina," *Annals of Biomedical Engineering*, 16: pp.53–64, 1988.
10. Shapley, R., "Visual cortex: pushing the envelope," *Nature: neuroscience*, vol.1, pp.95–96,1998.
11. Reichardt, W., *Autocorrelation, a principle for the evaluation of sensory information by the central nervous system*, Rosenblith Edition., Wiley, 1961.
12. Sakuranaga, M. and Naka, K.-I., "Signal transmission in the catfish retina. III. Transmission to type-C cell," *J. Neurophysiol.* 58: pp.411–428,1987.
13. Taub, E., Victor, J. D., and Conte, M. M., "Nonlinear preprocessing in short-range motion," *Vision Research*, 37: pp.1459–1477, 1997.
14. Heeger, D. J., "Modeling simple-cell direction selectivity with normalized, half-squared, linear operators," *J. Neurophysiol.* 70: pp.1885–1898, 1993.
15. Heeger, D. J., Simoncelli, E. P., and Movshon, J. A., "Computational models of cortical visual processing," *Proc. Natl. Acad. Sci, USA*, vol.93, pp.623–627, 1996.
16. Simoncelli E. P., and Heeger, D. J., "A Model of Neuronal Responses in Visual Area MT," *Vision Research*, vol.38, pp.743–761, 1998.
17. Lu, Z. L., and Sperling, G., "Three-systems theory of human visual motion perception: review and update," *J. Opt. Soc. Am. A*, Vol.18, pp.2331–2370, 2001.
18. Ishii, N., Deguchi, T., and Sasaki, H., "Parallel Processing for Movement Detection in Neural Networks with Nonlinear Functions," *Intelligent Data Engineering and Automated Learning*, LNCS vol.3177, pp.626–633, 2004.

# Fast Classification with Neural Networks via Confidence Rating\*

J. Arenas-García, V. Gómez-Verdejo, S. Muñoz-Romero,  
M. Ortega-Moral, and A.R. Figueiras-Vidal

Department of Signal Theory and Communications,  
Universidad Carlos III de Madrid, 28911 Leganés-Madrid, Spain  
{jarenas, vanessa, smunoz, ortegam, arfv}@tsc.uc3m.es  
<http://www.tsc.uc3m.es>

**Abstract.** We present a novel technique to reduce the computational burden associated to the operational phase of neural networks. To get this, we develop a very simple procedure for fast classification that can be applied to any network whose output is calculated as a weighted sum of terms, which comprises a wide variety of neural schemes, such as multi-net networks and Radial Basis Function (RBF) networks, among many others. Basically, the idea consists on sequentially evaluating the sum terms, using a series of thresholds which are associated to the confidence that a partial output will coincide with the overall network classification criterion. The possibilities of this strategy are well-illustrated by some experiments on a benchmark of binary classification problems, using RealAdaboost and RBF networks as the underlying technologies.

## 1 Introduction

Artificial Neural Networks (ANNs) are nowadays widely used in classification and regression problems because of their extraordinary abilities for input-output knowledge discovery [1, 2]. During the training phase, ANNs automatically adjust their parameters by means of a set of labeled patterns. Later, in the operational phase, the network is used to predict the output corresponding to new unseen patterns, or the class they belong to. Some of the most classical techniques, such as Multi-Layer Perceptrons (MLPs) and Radial Basis Functions Networks (RBFNs) [1] are known to have universal approximation capabilities. Besides, their performance can be improved ever further by means of Multi-Net (M-N) systems [3] that combine the output of several base learners. Recently, Support Vector Machines (SVMs) have been proposed [4] achieving excellent results, mainly in binary classification problems. Although the recognition capabilities of M-N systems and SVMs is out of discussion, the computational complexity of the resulting machine can be very high if the number of base learners or support vectors, respectively, is very large.

---

\* This work has been partly supported by CICYT grant TIC2002-03713.

In this paper we consider the binary classification problem, and we propose a very simple procedure to speed up the operational phase of most ANNs. In particular, this method can be applied to any network whose output consists of a weighted sum of different terms:

$$y(\mathbf{x}) = \text{sign}(o(\mathbf{x})) = \text{sign}\left(\sum_{m=1}^M \alpha_m o_m(\mathbf{x})\right) \quad (1)$$

where  $\mathbf{x} \in \mathfrak{R}^d$  is the input pattern and  $y(\mathbf{x}) = \pm 1$  the prediction about the class it belongs to,  $o_m(\mathbf{x})$  are the outputs of the  $M$  units that make up the network, and  $\alpha_m \in \mathfrak{R}$  their associated weights. Note that, to keep our formulation as general as possible, we are not assuming any predetermined implementation for the network units. In this way,  $o_m(\mathbf{x})$  could be a *kernel* function [5] as well as the output of a base learner of a M-N system.

For certain networks, such as M-N systems, RBFNs or SVMs, the number of units which are combined is usually very large, leading to a very intensive computational demand during the classification phase. To speed up pattern classification, we propose to sequentially evaluate the different terms in (1), achieving the following partial sums:

$$y_k(\mathbf{x}) = \sum_{m=1}^k \alpha_m o_m(\mathbf{x}) \quad (2)$$

Hopefully, if the absolute value of  $y_k(\mathbf{x})$  is high enough, the current sign of  $y_k(\mathbf{x})$  will coincide with  $y(\mathbf{x})$ , making it possible to avoid the calculation of the restating  $M - k$  network units. To be more precise, if

$$y_k(\mathbf{x}) > \eta_k^+ \quad \text{or} \quad y_k(\mathbf{x}) < \eta_k^-$$

we will classify  $\mathbf{x}$  according to the present sign of  $y_k(\mathbf{x})$ . On the contrary, if

$$\eta_k^- < y_k(\mathbf{x}) < \eta_k^+$$

we assume that the confidence about  $y(\mathbf{x})$  being equal to the sign of  $y_k(\mathbf{x})$  is not enough, making it necessary to proceed with the evaluation of the  $(k + 1)$ -th unit.

Obviously, to apply the above method in a particular scenario, it is necessary to select the thresholds  $\{\eta_k^+\}$  and  $\{\eta_k^-\}$  during the training phase. In the next section we will explain how this can be done with little extra computational effort. Furthermore, to get the most from this fast classification strategy, it would be desirable to sort out the units appropriately, taking into account the most relevant ones during an early stage of the classification process.

## 2 Setting the Classification Thresholds

In first term, we propose to set up the positive and negative thresholds,  $\{\eta_k^+\}$  and  $\{\eta_k^-\}$  respectively, to guarantee that the original network criterion is kept,

and no additional errors are provoked by the fast classification method. It is useful to distinguish two different cases depending on the range of  $o_m(\mathbf{x})$ :

i)  $o_m(\mathbf{x}) : \mathbb{R}^d \rightarrow [-1, 1]$ : In this case we have that the difference between the overall network and its  $k$ -th partial output is given by

$$\bar{y}_k(\mathbf{x}) = y_M(\mathbf{x}) - y_k(\mathbf{x}) = \sum_{m=k+1}^M \alpha_m o_m(\mathbf{x}) \tag{3}$$

that can be bounded by

$$- \sum_{m=k+1}^M |\alpha_m| < \bar{y}_k(\mathbf{x}) < \sum_{m=k+1}^M |\alpha_m|$$

Thus, choosing

$$\eta_k^+ = \sum_{m=k+1}^M |\alpha_m|; \quad \eta_k^- = -\eta_k^+ \tag{4}$$

satisfies our requirement that the overall network classification is preserved.

ii)  $o_m(\mathbf{x}) : \mathbb{R}^d \rightarrow [0, 1]$ : Using similar arguments to those in the previous case we get

$$\sum_{\substack{m=k+1 \\ \alpha_m < 0}}^M \alpha_m < \bar{y}_k(\mathbf{x}) < \sum_{\substack{m=k+1 \\ \alpha_m > 0}}^M \alpha_m$$

that allows us to set

$$\eta_k^+ = - \sum_{\substack{m=k+1 \\ \alpha_m < 0}}^M \alpha_m; \quad \eta_k^- = - \sum_{\substack{m=k+1 \\ \alpha_m > 0}}^M \alpha_m \tag{5}$$

Obviously, (4) and (5) are worst-case choices, consequently providing very reduced computational savings. It is immediate to relax these selections by introducing an additional parameter  $\beta \in [0, 1]$  that allows to redefine the thresholds as

$$\eta_k^{+'} = \beta \eta_k^+; \quad \eta_k^{-'} = \beta \eta_k^-$$

Parameter  $\beta$  measures our confidence that a fast decision on  $\mathbf{x}$  will respect the criterion of the overall classifier. As we have explained, if  $\beta = 1$  we are 100% sure that no additional errors are introduced by the fast classification process. On the contrary, when  $\beta = 0$  only the first unit is considered. Finally, values between 0 and 1 correspond to situations where an intermediate degree of certainty is required.

### 3 Sorting Out the Network Units

As explained in Section 1, not only the selection of the thresholds, but also the order of the network units plays an essential role in the performance of the fast classification method. As a general rule, to achieve a maximum computational saving, it is convenient to place the most relevant units in the first positions.

Although the construction of certain networks, such as RealAdaboost (RA) ensembles [6], directly gives the units in decreasing order of importance, this is not true in general. In this second case, different criteria can be established to measure unit relevance, and thus to sort out the units that compose the network. For instance, we could use the absolute values of  $\alpha_m$ . Another possibility is to use the average activation of each unit that we define as

$$\gamma_m = \sum_{i=1}^l |\alpha_m o_m(\mathbf{x}_i)| \quad (6)$$

where  $\mathbf{x}_i, i = 1, \dots, l$ , are the training patterns.

For illustrating both strategies we have considered an example in which the fast classification method was applied to a RBFN trained on the *contraceptive* dataset that will be presented in the next section. Analyzing the curves in Fig. 1 we can see that using an appropriate order is very important if one wants to achieve significant computational savings without affecting the recognition performance of the original classifier ( $\beta = 1$ ).

Although more sophisticated schemes could be designed, in the sequel we will use the decreasing order of  $|\alpha_m|$ , that provided a maximum computational saving in the previous example.

## 4 Experiments

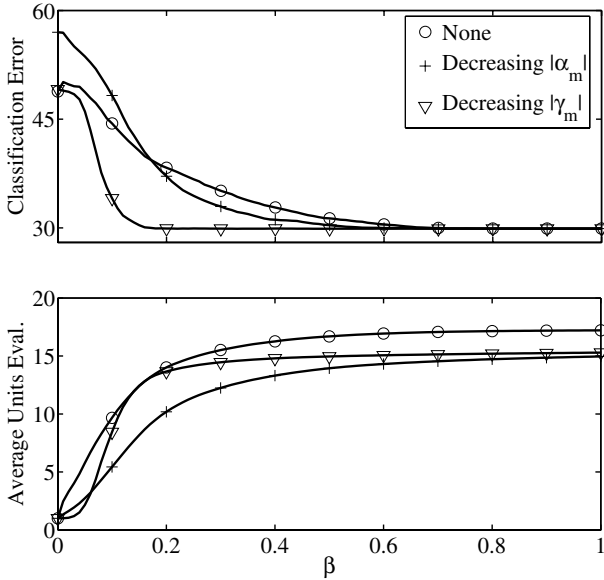
To illustrate the validity of the fast classification strategy, we have carried out a number of experiments on a benchmark of seven binary classification tasks. A summary of the main characteristics of each problem, including the number of training and test patterns in each class and the dimension of the input space, is given in Table 1. *Kwok* is a synthetic problem that was introduced in [7]; the rest are datasets of real data taken from the Machine Learning Repository of the University of California Irvine [8]. When no predefined train/test partition was provided, we used a 10 fold cross-validation strategy.

In the next two subsections we present results when using RealAdaboost ensembles of MLP classifiers, and for the excellent RBFN implementation of Rättsch (see, for instance, [9]). In both cases, all results were averaged over 50 independent designs.

### 4.1 Fast Classification with RealAdaboost Schemes

RealAdaboost (RA) [6] is a boosting algorithm that works by sequentially adding learners to an ensemble, progressively paying attention to the hardest to classify





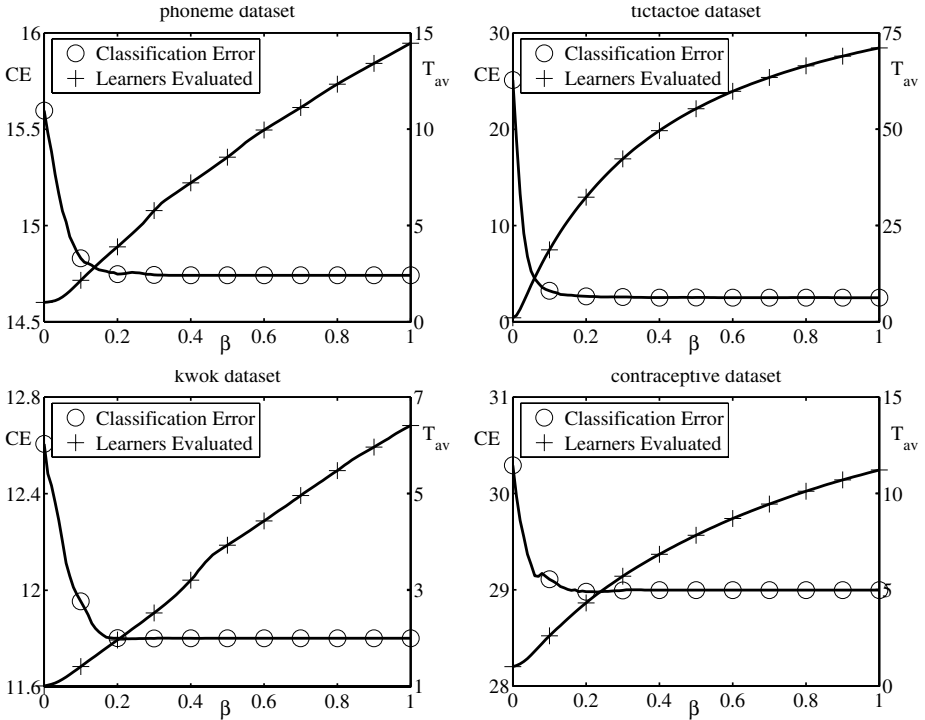
**Fig. 1.** Classification Error and average number of units evaluated during the test phase for a RBFN classifier trained on the *contraceptive* dataset when using different strategies for sorting out the network units

**Table 1.** Benchmark dataset description

|                      | # train   | # test    | dim |
|----------------------|-----------|-----------|-----|
| <i>kwok</i>          | 300/200   | 6120/4080 | 2   |
| <i>tictactoe</i>     | 199/376   | 133/250   | 9   |
| <i>contraceptive</i> | 506/377   | 338/252   | 9   |
| <i>image</i>         | 821/1027  | 169/293   | 18  |
| <i>spam</i>          | 1088/1673 | 725/1115  | 57  |
| <i>phoneme</i>       | 952/2291  | 634/1527  | 5   |
| <i>waveform</i>      | 2694/1306 | 659/341   | 21  |

patterns. For this set of experiments we have used MLP networks with one single hidden layer as the elements of RA, using the basic backpropagation algorithm for their training. Both the number of hidden neurons ( $N$ ) and the number of rounds for RA ( $T$ ) were fixed on a trial and error basis for each problem.

It is easy to identify the RA classification function with Eq. (1) by simply setting  $M = T$ , and letting  $o_m(\mathbf{x})$  and  $\alpha_m$  be the outputs of each base learner and its associated weight. For this particular case, and given the special characteristics of the RA scheme, it is not necessary to modify the original order of the units.



**Fig. 2.** Fast Classification with RealAdaboost ensembles. Each figure shows the classification error (CE) and average number of learners ( $T_{av}$ ) that need to be evaluated for different values of the confidence rate parameter  $\beta$

In Fig. 2 we have depicted the influence of  $\beta$  on the test classification error (CE) and on the average number of learners evaluated ( $T_{av}$ ) for four of the problems considered. As expected,  $\beta$  imposes a compromise between these two quantities, but there exists a wide range for which no performance degradation is observed and significant complexity reductions can be achieved.

Results for the complete benchmark are reported in Table 2. The original CE of RA is given under the ‘ $\beta = 1$ ’ column. We can see that, even for this very conservative case, important reductions in the computational load can be achieved. It is important to remark that this gain is obtained at a negligible cost during the training phase. Finally, computational savings of about 90% are not rare for lower values of  $\beta$ , keeping in all cases a very similar CE to that of the overall RA scheme.

## 4.2 Fast Classification with RBF Networks

RBFNs are well-known classifiers that obtain their output as a weighted linear combination of *radial basis functions*. We will only consider here the case of

**Table 2.** RealAdaboost test classification error (CE) and average number of learners evaluation ( $T_{av}$ ) for the seven problems in the benchmark and three values of  $\beta$ . Computational savings with respect to the complete evaluation of the RA machine are given inside parentheses

|                      | $N$ | $T$ | CE (%)<br>$\beta = 1$ | $T_{av}$<br>$\beta = 1$ | CE (%)<br>$\beta = .5$ | $T_{av}$<br>$\beta = .5$ | CE (%)<br>$\beta = .3$ | $T_{av}$<br>$\beta = .3$ |
|----------------------|-----|-----|-----------------------|-------------------------|------------------------|--------------------------|------------------------|--------------------------|
| <i>kwok</i>          | 4   | 20  | 11.8                  | 6.41<br>(-67.9%)        | 11.8                   | 3.92<br>(-80.4%)         | 11.8                   | 2.52<br>(-87.4%)         |
| <i>tictactoe</i>     | 8   | 100 | 2.51                  | 71.1<br>(-28.9%)        | 2.53                   | 55.3<br>(-44.7%)         | 2.58                   | 42.27<br>(-57.7%)        |
| <i>contraceptive</i> | 2   | 20  | 29                    | 11.21<br>(-43.9%)       | 29                     | 7.82<br>(-60.7%)         | 29                     | 5.7<br>(-71.5%)          |
| <i>image</i>         | 4   | 50  | 2.84                  | 7.5<br>(-85%)           | 2.85                   | 4.49<br>(-91%)           | 2.97                   | 2.99<br>(-94%)           |
| <i>spam</i>          | 5   | 50  | 5.68                  | 9.48<br>(-81%)          | 5.69                   | 5.13<br>(-89.7%)         | 5.73                   | 3.09<br>(-93.8%)         |
| <i>phoneme</i>       | 9   | 50  | 14.74                 | 14.46<br>(-71.1%)       | 14.74                  | 8.54<br>(-82.9%)         | 14.74                  | 5.77<br>(-88.4%)         |
| <i>waveform</i>      | 2   | 20  | 7.89                  | 6.73<br>(-66.3%)        | 7.9                    | 3.69<br>(-81.6%)         | 7.89                   | 2.23<br>(-88.8%)         |

**Table 3.** Test CE and Kernel Evaluations per Pattern (KEPP) when using RBFNs. The relative reduction on the KEPP with respect to the number of units in the network (second column) is also shown inside parentheses

|                      | KEPP | CE (%)<br>$\beta = 1$ | KEPP<br>$\beta = 1$ | CE (%)<br>$\beta = .5$ | KEPP<br>$\beta = .5$ | CE (%)<br>$\beta = .3$ | KEPP<br>$\beta = .3$ |
|----------------------|------|-----------------------|---------------------|------------------------|----------------------|------------------------|----------------------|
| <i>kwok</i>          | 10   | 11.68                 | 7.45<br>(-25.5%)    | 11.75                  | 6.26<br>(-37.4%)     | 12.85                  | 4.7<br>(-53%)        |
| <i>tictactoe</i>     | 29   | 0.39                  | 24.45<br>(-15.7%)   | 0.39                   | 22.55<br>(-22.2%)    | 0.39                   | 20.57<br>(-29.1%)    |
| <i>contraceptive</i> | 18   | 29.9                  | 14.96<br>(-16.9%)   | 30.37                  | 13.94<br>(-22.5%)    | 32.96                  | 12.24<br>(-32%)      |
| <i>image</i>         | 185  | 3.16                  | 151.7<br>(-18%)     | 3.16                   | 139.98<br>(-24.3%)   | 3.16                   | 129.83<br>(-29.8%)   |
| <i>spam</i>          | 277  | 6.06                  | 246.67<br>(-10.9%)  | 6.06                   | 235.24<br>(-15.1%)   | 6.06                   | 224.17<br>(-19.1%)   |
| <i>phoneme</i>       | 325  | 12.06                 | 284.92<br>(-12.3%)  | 12.06                  | 268.67<br>(-17.3%)   | 12.06                  | 253.17<br>(-22.1%)   |
| <i>waveform</i>      | 80   | 8.75                  | 61.76<br>(-22.8%)   | 8.75                   | 56.96<br>(-28.8%)    | 8.87                   | 52.06<br>(-34.9%)    |

Gaussian kernels:

$$o_m(\mathbf{x}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{c}_m\|^2}{2\sigma_m^2}\right)$$

We used 2%, 5%, and 10% of the training data as the number of kernels in the RBFN, keeping the setting which offered a best result. The position and width of the kernels were automatically adjusted by means of a gradient descent scheme.

Results are reported in Table 3 in terms of test CE and average number of Kernel Evaluations per Pattern (KEPP) during the test phase. Again, the CE is kept under control even for  $\beta = 0.3$ , but now the computational saving is not as impressive as in the RA case. We are currently studying other ways to sort out the RBFN kernels to further reduce the KEPP.

## 5 Conclusions

In this paper we have presented a very simple procedure to alleviate the computational burden required by powerful neural network classifiers. The scheme works by sequentially comparing the partial output of the net to a positive and a negative thresholds, taking decisions as soon as a desired degree of certainty is reached.

The proposed scheme has shown to be specially suited for Adaboost type classifiers, in which the constituent units are by construction given in order of relevance. Other ANN schemes obtain more limited improvements and they could benefit from the design of algorithms that obtain a convenient order of the network units. Another interesting line of research would consist on extending these ideas to speed up the training phase.

## References

1. C. M. Bishop: *Neural Networks for Pattern Recognition*. Oxford Univ. Press, New York, NY (1995).
2. O. D. Duda, P. E. Hart and D. G. Stork: *Pattern Classification*. Wiley, New York, NY (2001).
3. A. J. C. Sharkey (ed.): *Combining Artificial Neural Nets. Ensemble and Modular Multi-Net Systems. Perspectives in Neural Computing*. Springer-Verlag, London, UK (1999).
4. C. J. C. Burges: A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, **2** (1998) 121-167.
5. B. Schölkopf and A. J. Smola: *Learning with Kernels*. MIT Press, Cambridge, MA (1998).
6. R. E. Schapire and Y. Singer: Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, **37**(3) (1999) 297-336.
7. J. T. Kwok: Moderating the Output of Support Vector Machine Classifiers. *IEEE Trans. on Neural Networks*, **10**(5) (1999) 1018-1031.
8. C. L. Blake and C. J. Merz: *UCI Repository of machine learning databases*. <http://www.ics.uci.edu/~mllearn/MLRepository.html>. University of California, Irvine, Dept. of Information and Computer Sciences (1998).
9. K.-R Müller et al.: Predicting time series with Support Vector Machines. In *Advances in Kernel Methods—Support Vector Learning*, Cambridge, MA (1999).

# Characterization and Synthesis of Objects Using Growing Neural Gas

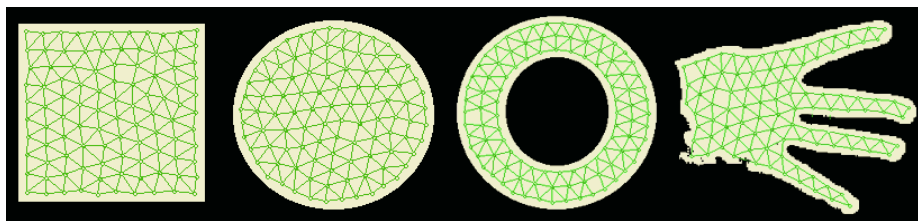
José García, Francisco Flórez, Juan Manuel García, and Antonio Hernández

Departamento de Tecnología Informática y Computación,  
Universidad de Alicante, Apdo 99 . 03080 Alicante, Spain  
{florez, juanma, jgarcia, ahernandez}@dtic.ua.es  
<http://www.ua.es/i2rc>

**Abstract.** In this article it is made a study of the characterization capacity and synthesis of objects of the self-organizing neural models. These networks, by means of their competitive learning, try to preserve the topology of an input space. This capacity is being used for the representation of objects and their movement with topology preserving networks. We characterized the object to represent by means of the obtained maps and kept information solely on the coordinates and the colour from the neurons. From this information it is made the synthesis of the original images, applying mathematical morphology and simple filters on the information which it is had.

## 1 Introduction

The objective of the synthesis or compression of objects is to minimize the amount of data to store for the correct representation of the images, with the smaller loss of information. The compression algorithms usually try to eliminate the redundancy in the data, so that it is possible the reconstruction of the image. The usual sequence of characterization/synthesis process will consist of the redundancy data reduction obtaining the most relevant data, the codification of these data, transmission, decoding,



**Fig. 1.** Representation of bi-dimensional objects with self-organizing networks

and reconstruction or synthesis of the images. It is known that the self-organizing neural networks, by means of a competitive learning, make an adaptation of the vec-

tors of reference of the neurons as well as of the network of interconnection among them; obtaining a mapping that tries to preserve the topology of an input space. Also, they are able of a continuous readjustment with new input patterns, with no need to begin the learning again. These capacities have been used for the representation of objects [1] and their movement [2] by means of a self-organizing model, Growing Neural Gas (GNG) [3], that has a process of learning less strict than other self-organizing models, like the Kohonen maps [4].

In figure 1 the capacity of representation of bi-dimensional objects with these neural networks can be observed, extracting the more relevant topological information as much to make the analysis of the object shape, to make the synthesis of this. In this work we use GNG for the characterization and syntheses of gestures of the hand.

## 2 Growing Neural Gas

Growing Neural Gas is a neural model in which a topology of union between the neurons does not settle down and in which, from an initial number of neurons, these they are inserted or eliminated until a condition of conclusion is fulfilled (maximum number of neurons, time available, error of quantization).

The algorithm of learning of the GNG he is the following one:

1. Start with two neurons  $a$  and  $b$  at random positions  $w_a$  and  $w_b$  in  $\mathcal{R}^d$ .
2. Generate an input signal  $\xi$  according to a density function  $\mathcal{P}(\xi)$ .
3. Find the nearest neuron (winner neuron)  $s_1$  and the second nearest  $s_2$ .
4. Increase the age of all the edges emanating from  $s_1$ .
5. Add the squared distance between the input signal and the winner neuron to a counter error of  $s_1$ :

$$\Delta error(s_1) = \|w_{s_1} - \xi\|^2 \tag{1}$$

6. Move the winner neuron  $s_1$  and its topological neighbours (neurons connected to  $s_1$ ) towards  $\xi$  by a learning step  $\mathcal{E}_w$  and  $\mathcal{E}_n$ , respectively, of the total distance:

$$\Delta w_{s_1} = \mathcal{E}_w (\xi - w_{s_1}) \tag{2}$$

$$\Delta w_{s_n} = \mathcal{E}_n (\xi - w_{s_n}) \tag{3}$$

7. If  $s_1$  and  $s_2$  are connected by an edge, set the age of this edge to 0. If it does not exist, create it.
8. Remove the edges larger than  $a_{max}$ . If this results in isolated neurons (without emanating edges), remove them as well.
9. Every certain number  $\lambda$  of input signals generated, insert a new neuron as follows:
  - Determine the neuron  $q$  with the maximum accumulated error.
  - Insert a new neuron  $r$  between  $q$  and its further neighbour  $f$ :

$$w_r = 0.5(w_q + w_f) \quad (4)$$

- Insert new edges connecting the neuron  $r$  with neurons  $q$  and  $f$ , removing the old edge between  $q$  and  $f$ .
- Decrease the error variables of neurons  $q$  and  $f$  multiplying them with a constant  $\alpha$ . Initialize the error variable of  $r$  with the new value of the error variable of  $q$  and  $f$ .

10. Decrease all error variables by multiplying them with a constant  $\beta$ .
11. If the stopping criterion is not yet achieved, go to step 2.

In summary, the adaptation of the network to the input space takes place in step 6. The insertion of connections (step 7) between the two neurons nearer to each one of the input patterns establishes, finally, an induced Delaunay triangulation by the input space. The elimination of connections (step 8) eliminates the edges that no longer must comprise of this triangulation. This is made eliminating the connections between neurons that no longer are next or that they have nearer neurons. Finally, the accumulation of the error (step 5) allows to identify those zones of the input space where it is necessary to increase the number of neurons to improve the mapping.

### 3 System Description

The characterization/synthesis system is made up of modules for the segmentation, characterization and synthesis of images or sequences of images (figure 2).

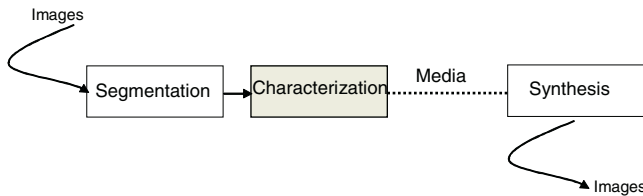


Fig. 2. System for characterization/synthesis of images

#### 3.1 Segmentation

The segmentation of images in colour has been a field on which has studied extensively in the last years [5]. As well as the use of this for the tracking of objects in images [6][7][8]. The colour offers several advantages respect to the geometric information that it can be insufficient under adverse conditions, like partial occlusions or changes of scale or resolution. In this work we have tried to make an initial module for colour segmentation that allowed the segmentation of gestures of the hand in sequences of images with complex background, whose results served as input to a complex neural system for gesture recognition [2]. For it, a study of different colour models for skin segmentation has been made and the use of model HSI with thresholds empirically has been decided. This model is invariant to small changes in the illumination of the scene and provides a robust images segmentation.

### 3.2 Characterization

Given an image  $I(\chi, y) \in \mathcal{R}$  we perform the transformation  $\psi_{\mathcal{T}}(\chi, y) = \mathcal{T}(I(\chi, y))$  that associates to each one of the pixels its probability of belonging to the object, according to a property  $\mathcal{T}$ . For instance, in figure 1, this transformation is a threshold function.

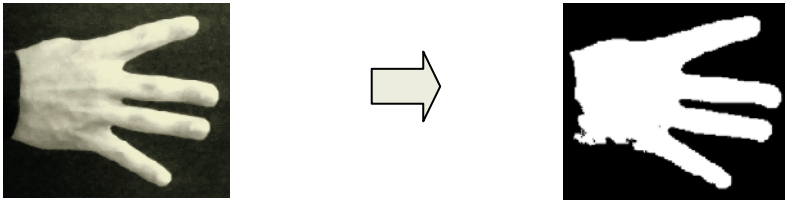


Fig. 3. Silhouette extraction

If we consider  $\xi = (\chi, y)$  and  $\mathcal{P}(\xi) = \psi_{\mathcal{T}}(\xi)$ , we can apply the learning algorithm of the GNG to the image  $I$ , so that the network adapts its topology to the object. This adaptive process is iterative, so the GNG represents the object during all the learning, giving the opportunity to stop the process if necessary, obtaining a good answer (figure 3).

As a result of the GNG learning we obtain a graph, the Topology Preserving Graph  $\mathcal{TPG} = \langle \mathcal{N}, \mathcal{C} \rangle$ , with a vertex (neurons) set  $\mathcal{N}$  and an edge set  $\mathcal{C}$  that connect them (figure 1). This  $\mathcal{TPG}$  establishes a Delaunay triangulation induced by the object.

### 3.3 Synthesis

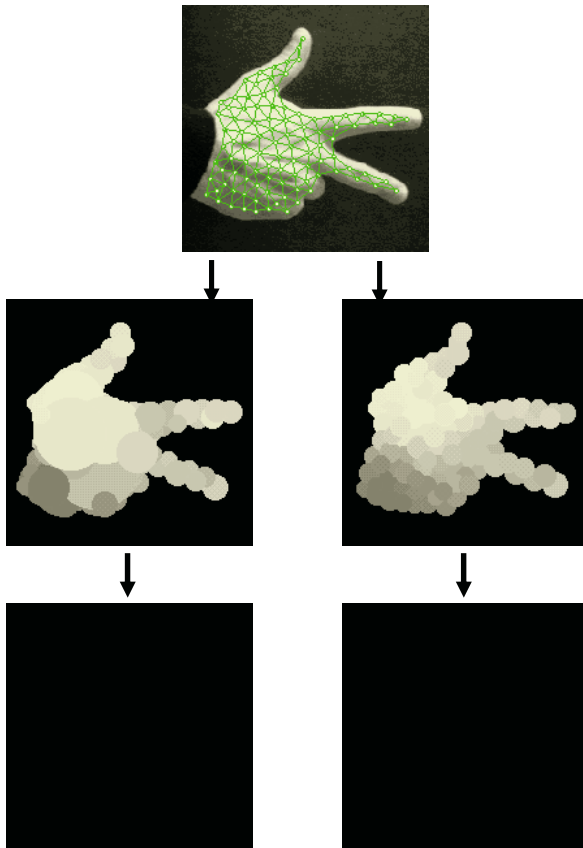
The capacity for information compression of self-organizing neural networks has been demonstrated in this work. The representation of a bi-dimensional object by means of  $\mathcal{GPT}$  extracts more important information about the object topology to make the analysis of the object shape. In this point, the inverse problem considers: how to generate the shape of the object from his  $\mathcal{GPT}$ ? That is to say, how to apply this work to the synthesis of objects. For it, it is come following two alternatives, depending on the information available at the moment for making the synthesis: If at the moment of the adaptation of  $\mathcal{GPT}$  we have information associated to each neuron, relative to the distance  $r$  of any of them to the closest point of the contour of the object  $t$ ; it is drawn for each one of the neurons a circle with radius  $r$ .

If we only had the  $\mathcal{GPT}$ , without additional information, it is drawn circles with centre in each one of the neurons and with a radius  $r$  that is based on the set of edges that leave each one of them:



$$r_i = \frac{\sum_{\forall j \in \mathcal{N}_i} \|w_i - w_j\|^2}{\text{card}(\mathcal{N}_i)} \tag{5}$$

In figure can be seen the result of both steps of the synthesis process. In a first step ellipses in the coordinates of the neurons are drawn and later in a second step a medium filter is applied to smooth the contour of the object. Also, if information from the intensity or colour of any of the points on the original image that corresponds to each one of the neurons coordinates is available, we can create a colour image, so that the synthesis has a similar visual appearance than the real object. In order to improve the quality of the synthesis, eliminating the effect of the drawing of multiple circles, is made a filtrate (medium, erosion/expansion...).



**Fig. 4.** Gray level images synthesis from a self-organizing map

## 4 Experiments and Results

Experiments with different gestures from the hand have been made to prove the effectiveness of the method, obtaining results, that although not represent exactly the original aspect of the gestures, have enough quality to visually recognize the gestures made. The images used took with a window from 200x200 pixels, using a GNG neural network with 49 neurons. Therefore we are reconstructing the object with only 49 coordinates instead of the over 4000 points of the image.

In the following figures the results of the GNG learning process and later reconstruction of the gestures of the hand can be observed.

The colour of any of the neurons coordinates in the original image is assigned to the area of influence in the object, calculated based on the Voronoi diagram obtained from the Delaunay triangulation that the neurons represent.

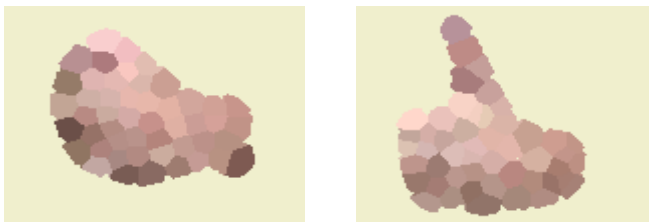
Finally an improvement is made on the previous reconstruction, applying an average filter on the result of the morphologic operations of reconstruction.



**Fig. 5.** Original colour images for characterization/reconstruction



**Fig. 6.** Object characterization with GNG



**Fig. 7.** Reconstruction based on the Voronoi diagram



**Fig. 8.** Reconstruction with average filter

## 5 Conclusions and Further Work

In this work we have demonstrated the compression capacity of self-organizing maps, taking as example the characterization and later reconstruction from gestures of the hand. For it, a previous study of the capacity of segmentation of different colour models has become. Although the reconstruction has not enough quality for some applications, the reduction of information when making the synthesis turns to the method in a tool of compression with numerous applications like the video-conference for deaf people. At the present time we are working in the optimization of the reconstruction models, as well as in the optimization of the self-organizing maps used and the hardware implementation of these to increase the global system speed.

## References

1. Flórez, F., García, J.M., García, J., Hernández, A.: Representation of 2D Objects with a Topology Preserving Network. In Proceedings of the 2<sup>nd</sup> International Workshop on Pattern Recognition in Information Systems (PRIS'02), Alicante. ICEIS Press (2001) 267-276
2. Flórez, F., García, J.M., García, J., Hernández, A.: Hand Gesture Recognition Following the Dynamics of a Topology-Preserving Network. In Proc. of the 5<sup>th</sup> IEEE Intern. Conference on Automatic Face and Gesture Recognition, Washington, D.C. IEEE, Inc. (2001) 318-323
3. Fritzke, B.: A Growing Neural Gas Network Learns Topologies. In Advances in Neural Information Processing Systems 7, G. Tesauro, D.S. Touretzky y T.K. Leen (eds.), MIT Press (1995) 625-632
4. Kohonen, T.: Self-Organizing Maps. Springer-Verlag, Berlin Heidelberg (1995)
5. W. Skarbek y A. Koschan, Colour Image Segmentation - A Survey, Technical Report 94-32, Technical University Berlin. Octubre 1994.
6. M. Sonka, V. Hlavac y R. Boyle. Image Processing, Analysis, and Machine Vision (2nd edition). Brooks/Cole Publishing Company, 1998.
7. C. Cédras y M. Shah. Motion-based recognition: a survey. Image and Vision Computing, 13(2):129-155, 1995.
8. S. McKenna , Y. Raja, y S. Gong. Color Model Selection and Adaptation in Dynamic Scenes. European Conference on Computer Vision, 1998
9. H. Stern y B. Efron. Adaptative Color Space Switching for Face Tracking in Multi-Colored Lighting Environments. In Proceedings of the 5th International Conference on Automatic Face and Gesture Recognition, Washington ,2002.

# ARGEN + AREPO: Improving the Search Process with Artificial Genetic Engineering

Agustín León-Barranco and Carlos A. Reyes-García

Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE),  
Luis Enrique Erro No. 1, Sta. Ma. Tonanzintla, Puebla, 72840, México  
{agustinleonb, kargaxxi}@inaoep.mx

**Abstract.** In this paper we analyze the performance of several evolutionary algorithms in the feature and instance selection problem. It is also introduced the ARGEN + AREPO search algorithm which has been tested in the same problem. There is no need to adapt parameters in this genetic algorithm, except the population size. The reported preliminary results show that using this technique in a wrapper model to search data subsets, we can obtain similar accuracy like with the hill-climbers and genetic algorithms models also here presented, but keeping a less amount of data.

## 1 Introduction

To search feature and instance subsets, looking for a reduction of the original sets, can help to improve performance in prediction tasks using the smallest possible computational resources. Two of the most used search strategies in this kind of problems are Hill-Climbers (HCs) and Genetic Algorithms (GAs). The kind of GA presented in this paper keeps the advantages of both searching techniques and, besides, it is reinforced by means of the ARGEN concept here described. The result is an “*artificial algorithm with a more natural behavior*”.

## 2 Data Selection

The models to select feature and instance subsets presented in this paper are based on wrapper architecture [7]. These models use the *Distance-Weighted K-Nearest Neighbors* (DW-KNN) classifier [11] to measure the accuracy of training feature and instance subsets over the evaluation data (unseen data). The neighbors in the DW-KNN algorithm for the reported experiments were three. Two kinds of search algorithms were used in the models: hill-climbers and genetic algorithms. They allow us to search and select a feature and instance subset from the original training data. The search process is run until a maximum number of iterations have been reached. The wrapper model returns the feature and instance subset with the highest fitness value found, i.e., the data subset which best classifies the evaluation data. The fitness function is formed by the accuracy over the “*evaluation data*” with a certain “*training*”

subset” (taken from the “original training data”) and, the proportion of the “training subset” with regard to the “original training data” [8]. The fitness function is formed in the following way:

$$F_F (DataSet) = P (DataSet) - \alpha \frac{n + m}{N + M}$$

Where the value of “ $\alpha$ ” is 1, to differentiate two data subsets which have the same accuracy, but, different amount of data, giving a higher fitness value to the one which has less data. To represent feature and instance subsets in this paper, bit-strings of length  $N+M$  are used, where  $N$  is the number of instances and  $M$  the number of features of the original training data set. One bit “1” in the  $i^{th}$  position indicates that one feature or instance should be part of the “training subset” and one bit “0” indicates that it should be eliminated, see Figure 1.

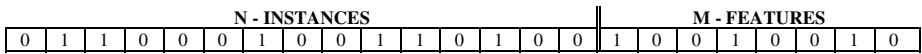


Fig. 1. Representation of one chromosome (training subset) which has 6 ( $n$ ) instances with 3 ( $m$ ) features each, these data are selected from an “original training data“ set of 15 ( $N$ ) instances with 8 ( $M$ ) features each

### 3 Hill-Climbers and Genetic Algorithms

The hill-climbing algorithms are searching techniques that use binary notation. Unlike the genetic algorithms, regularly they manipulate a single chromosome [4]. Generally these start with a randomly generated solution. Then, every bit is flipped in a certain order, one at-a-time. If a better solution is found, it is kept, and the procedure continues until no improvements can be found or, until a given number of iterations is reached. There are many different HC algorithms, but, in this paper only four were selected for evaluation in the feature and instance selection problem, they are the following: SAHC (*Steepest-Ascent Hill Climbing*), NAHC (*Next-Ascent Hill Climbing*), RMHC (*Random-Mutation Hill Climbing*) [9] and RBC (*Random Bit Climbing*) [4]. The GAs are powerful heuristic search strategies based upon evolution mechanisms, some researchers use these as a basis of optimum-seeking techniques in case of vast search spaces [2], in this work, the feature and instance subsets space. There are many problems which are easily solved by means of GAs, but, can be difficult for a basic hill-climbing technique. Five GAs were selected for evaluation and to compare their performance with the HCs and the novel algorithm presented in this paper. The chosen algorithms are the following: SGA (*Simple Genetic Algorithm*) [10], sometimes known as *canonical genetic algorithm* [14], TDA (*Traditional Genetic Algorithm*) which differs from the first because one or more individuals are passed to the next generation by elitism (steady-state) [13], CHC (*Cross-generational elitist selection, Heterogeneous recombination and Cataclysmic mutation*) [5], GENITOR (*Genetic ImplemenTOR*) [15] and CFRSC (*Common Feature/Random Sample Climbing*) [6]. Section 4 presents a general description of the genetic algorithms suggested in this paper.

## 4 ARGEN + AREPO

ARGEN (*Artificial Genetic Engineering*) + AREPO (*Artificial Evolution of POPulations*) is an improved version of AREPO, an adaptive evolutionary algorithm. AREPO is a population level based evolutionary algorithm that moves from Classical Genetic Algorithms towards *Artificial Evolution of Populations*. It is composed by individuals able to meet and interact. AREPO is a technique for setting the genetic parameters during a run by adapting the population size and the operator rates on the basis of the environmental constrain of a maximum population size (resource of the ecosystem). In addition, genetic operators are seen as alternative reproduction strategies and also fighting among individuals is used [1].

**Table 1.** Results of HCs on the Feature and Instance Selection for UCI Repository datasets

| Database           | Method | Real Accuracy (%) |       | Method Accuracy (%) |       | Storage |
|--------------------|--------|-------------------|-------|---------------------|-------|---------|
|                    |        | Average           | Best  | Average             | Best  |         |
| Breast Cancer (WI) | SAHC   | 94.57             | 97.86 | 98.56               | 100   | 27.95   |
|                    | NAHC   | 94.14             | 96.43 | 99.14               | 100   | 1.27    |
|                    | RMHC   | 95.43             | 97.14 | 98.78               | 100   | 27.21   |
|                    | RBC    | 94.50             | 97.86 | 97.12               | 100   | 31.37   |
| Glass              | SAHC   | 60.93             | 69.77 | 77.86               | 88.09 | 29.05   |
|                    | NAHC   | 60.93             | 69.77 | 86.19               | 95.24 | 9.42    |
|                    | RMHC   | 62.56             | 72.09 | 78.33               | 85.71 | 28.16   |
|                    | RBC    | 60.23             | 69.77 | 69.29               | 83.33 | 26.57   |
| Ionosphere         | SAHC   | 83.94             | 88.73 | 92.29               | 98.57 | 20.96   |
|                    | NAHC   | 83.66             | 87.32 | 96.57               | 100   | 2.55    |
|                    | RMHC   | 84.08             | 87.32 | 92.00               | 97.14 | 20.89   |
|                    | RBC    | 84.65             | 90.14 | 90.43               | 94.29 | 22.68   |
| Iris               | SAHC   | 93.67             | 96.67 | 99.67               | 100   | 27.09   |
|                    | NAHC   | 93.00             | 100   | 99.67               | 100   | 3.80    |
|                    | RMHC   | 94.33             | 100   | 99.00               | 100   | 26.25   |
|                    | RBC    | 92.33             | 100   | 98.67               | 100   | 29.81   |
| Liver Disorders    | SAHC   | 61.01             | 73.91 | 75.94               | 79.71 | 30.46   |
|                    | NAHC   | 59.13             | 72.46 | 90.87               | 94.20 | 11.21   |
|                    | RMHC   | 57.97             | 73.91 | 75.94               | 78.26 | 28.45   |
|                    | RBC    | 60.00             | 73.91 | 71.45               | 78.26 | 26.25   |
| Monks1             | SAHC   | 96.39             | 98.15 | 96.39               | 97.68 | 27.15   |
|                    | NAHC   | 99.12             | 100   | 99.77               | 100   | 13.23   |
|                    | RMHC   | 95.65             | 100   | 95.74               | 100   | 27.14   |
|                    | RBC    | 85.32             | 96.76 | 86.76               | 97.68 | 38.16   |
| Monks2             | SAHC   | 64.58             | 71.76 | 71.48               | 73.61 | 42.52   |
|                    | NAHC   | 62.82             | 68.06 | 81.39               | 84.72 | 29.58   |
|                    | RMHC   | 65.37             | 69.44 | 71.30               | 73.15 | 43.67   |
|                    | RBC    | 63.89             | 69.44 | 67.68               | 70.83 | 26.73   |
| Monks3             | SAHC   | 98.89             | 100   | 99.72               | 100   | 26.76   |
|                    | NAHC   | 100               | 100   | 100                 | 100   | 7.14    |
|                    | RMHC   | 98.01             | 100   | 98.93               | 100   | 23.39   |
|                    | RBC    | 91.34             | 97.22 | 92.45               | 99.07 | 29.88   |

ARGEN is the reinforced part of AREPO. GEN (*Genetic Engineering*) [12] biologically is a set of techniques that allow us to modify the organism’s characteristics in a predetermined sense, by means of the alteration of its genetic material. ARGEN (*Artificial GEN*) is a term that is introduced in this paper to include all kinds of alterations made to an individual (in GAs) to improve his fitness. Three levels of alteration have been distinguished: “*bit-level*”, where the alterations are directly made to the chromosome’s bits (mutation), “*individual-level*”, where the alteration is given by the way in which new information is introduced to one chromosome (crossover) and finally, the “*population-level*” which is the way the chromosomes are selected to mate and reproduce (*artificial selection*). The difference with the traditional way is that this alterations poof-based let us adapt the individual to a certain environment, i.e., adapt better the individuals to a certain problem.

The general ARGEN + AREPO algorithm is the following:

1. Initialize population
2. Evaluate population
3. Alter best individual (bit-level)
4. While not termination condition
  - 4.1. Get two individuals (natural & artificial selection)
  - 4.2. If "meeting"
    - 4.2.1. If "reproduction"
      - 4.2.1.1. Crossover (individual-level)
      - 4.2.1.2. Evaluate
    - 4.2.2. Else
      - 4.2.2.1. Simple Mutation
      - 4.2.2.2. Evaluate
  - 4.3. Else
    - 4.3.1. Fight
5. Alter best individual (bit-level)

#### 4.1 Adaptation Rules

One of the main features of ARGEN+AREPO algorithm is the adaptability of the parameters like in [1]. Population size is constrained by the environmental limit and its dynamics are determined by the meeting probability, reproduction and competition rules among individuals. These three adaptive rates are defined as:

$$P_m = \frac{C_p}{M_p} \quad P_r = 1 - P_m \quad P_c = 1 - P_r$$

Where  $P_m$  is the meeting probability (the population density),  $P_r$  is the reproduction rate,  $P_c$  the competition rate,  $C_p$  the current population size and  $M_p$  the maximum population size. Thus, if the population density is low the reproduction rate is high and the competition rate is low. Vice versa, because of the lack of environmental resources, the competition rate gets higher and the reproduction rate decreases.

#### 4.2 Meeting and Competition

When two individuals meet they can interact in two ways: by reproducing (crossover) or by fighting for natural resources (the stronger kills the weaker), otherwise the current individual can generate a new individual by a simple mutation. Competition starts according to the adaptive rate  $P_c$ . It means that when two individuals meet and they do not mate then they fight for survival, the stronger kills the weaker and this one is removed from the population.

#### 4.3 Initialization and Finalization

In ARGEN + AREPO the initial size of the population is a random number, limited by the maximum size of individuals. All individuals are unique, ranked in the population by their fitness. As can be observed in Table 1, the results of hill-climbing algorithms show that these algorithms are good searching techniques, so, in the initial population we alter the best chromosome by means of a hill-climber algorithm with the intention to improve his fitness. The finalization of the algorithm is given in the same way; the best chromosome is altered with a hill-climbing algorithm to improve its fitness. Like in Genitor, ARGEN + AREPO algorithm ends when a maximum number of iterations are reached.

**Table 2.** Performance of GAs in the data selection problem. The results of the original AREPO algorithm and the one introduced are included

| Database           | Method        | Real Accuracy (%) |       | Method Accuracy (%) |       | Storage |
|--------------------|---------------|-------------------|-------|---------------------|-------|---------|
|                    |               | Average           | Best  | Average             | Best  |         |
| Breast Cancer (WI) | SGA           | 95.14             | 99.29 | 97.91               | 99.28 | 29.45   |
|                    | TGA           | 95.21             | 97.14 | 99.21               | 100   | 16.80   |
|                    | CHC           | 95.86             | 97.14 | 99.21               | 100   | 25.56   |
|                    | CFRSC         | 94.93             | 97.86 | 99.35               | 100   | 4.87    |
|                    | GENITOR       | 95.86             | 98.57 | 99.35               | 100   | 23.97   |
|                    | AREPO         | 95.21             | 97.14 | 99.35               | 100   | 26.02   |
|                    | ARGEN + AREPO | 95.11             | 97.84 | 98.93               | 100   | 1.42    |
| Glass              | SGA           | 63.02             | 67.44 | 75.48               | 88.09 | 29.67   |
|                    | TGA           | 60.93             | 72.09 | 86.67               | 92.86 | 15.31   |
|                    | CHC           | 61.63             | 69.77 | 84.76               | 90.48 | 28.93   |
|                    | CFRSC         | 60.46             | 76.74 | 87.86               | 95.24 | 18.79   |
|                    | GENITOR       | 61.63             | 76.74 | 85.48               | 92.86 | 24.69   |
|                    | AREPO         | 59.30             | 65.17 | 83.33               | 95.24 | 23.45   |
|                    | ARGEN + AREPO | 57.14             | 66.67 | 85.12               | 93.02 | 8.95    |
| Ionosphere         | SGA           | 84.65             | 88.73 | 90.29               | 97.14 | 22.25   |
|                    | TGA           | 84.51             | 90.14 | 96.86               | 100   | 11.25   |
|                    | CHC           | 84.65             | 87.32 | 96.14               | 100   | 16.10   |
|                    | CFRSC         | 85.21             | 90.14 | 97.57               | 100   | 3.12    |
|                    | GENITOR       | 85.35             | 88.73 | 96.29               | 98.57 | 17.99   |
|                    | AREPO         | 85.35             | 90.14 | 95.29               | 97.14 | 18.66   |
|                    | ARGEN + AREPO | 84.86             | 92.86 | 95.77               | 98.59 | 2.49    |
| Iris               | SGA           | 94.67             | 100   | 98.67               | 100   | 28.33   |
|                    | TGA           | 93.00             | 100   | 100                 | 100   | 6.05    |
|                    | CHC           | 93.67             | 100   | 100                 | 100   | 10.39   |
|                    | CFRSC         | 94.33             | 100   | 99.33               | 100   | 6.03    |
|                    | GENITOR       | 94.00             | 100   | 100                 | 100   | 15.83   |
|                    | AREPO         | 93.00             | 100   | 100                 | 100   | 20.22   |
|                    | ARGEN + AREPO | 96.67             | 100   | 100                 | 100   | 2.67    |
| Liver Disorders    | SGA           | 62.32             | 71.01 | 76.23               | 79.71 | 32.17   |
|                    | TGA           | 60.43             | 68.12 | 90.72               | 95.65 | 28.83   |
|                    | CHC           | 61.30             | 68.12 | 85.36               | 88.41 | 32.15   |
|                    | CFRSC         | 59.71             | 68.12 | 92.03               | 97.10 | 24.82   |
|                    | GENITOR       | 61.74             | 73.91 | 85.65               | 88.41 | 36.39   |
|                    | AREPO         | 61.45             | 68.12 | 83.04               | 86.96 | 35.14   |
|                    | ARGEN + AREPO | 60.43             | 71.01 | 90.29               | 97.10 | 9.15    |
| Monks1             | SGA           | 97.87             | 100   | 98.80               | 100   | 25.28   |
|                    | TGA           | 100               | 100   | 100                 | 100   | 11.01   |
|                    | CHC           | 100               | 100   | 100                 | 100   | 17.30   |
|                    | CFRSC         | 100               | 100   | 100                 | 100   | 10.97   |
|                    | GENITOR       | 100               | 100   | 100                 | 100   | 18.51   |
|                    | AREPO         | 100               | 100   | 100                 | 100   | 20.97   |
|                    | ARGEN + AREPO | 100               | 100   | 100                 | 100   | 8.79    |
| Monks2             | SGA           | 68.80             | 73.61 | 70.93               | 74.54 | 46.51   |
|                    | TGA           | 69.26             | 74.07 | 86.48               | 88.43 | 59.41   |
|                    | CHC           | 68.10             | 73.61 | 81.76               | 84.26 | 59.88   |
|                    | CFRSC         | 66.90             | 72.22 | 84.26               | 86.57 | 52.43   |
|                    | GENITOR       | 68.80             | 75.00 | 80.32               | 82.41 | 54.75   |
|                    | AREPO         | 66.39             | 71.76 | 77.78               | 80.56 | 53.45   |
|                    | ARGEN + AREPO | 65.51             | 71.76 | 85.23               | 89.81 | 37.55   |
| Monks3             | SGA           | 96.39             | 100   | 98.06               | 100   | 22.35   |
|                    | TGA           | 99.58             | 100   | 99.86               | 100   | 8.24    |
|                    | CHC           | 99.31             | 100   | 99.58               | 100   | 14.43   |
|                    | CFRSC         | 99.54             | 100   | 99.91               | 100   | 8.62    |
|                    | GENITOR       | 100               | 100   | 100                 | 100   | 17.54   |
|                    | AREPO         | 100               | 100   | 100                 | 100   | 19.67   |
|                    | ARGEN + AREPO | 100               | 100   | 100                 | 100   | 6.19    |



#### 4.4 Natural and Artificial Selection

*Artificial selection is the modification of the hereditary constitution of the offspring by controlling the crossover between the parents.* The idea presented in ARGEN + AREPO combines the *Natural and Artificial Selection* because the first is based upon the notion that all individuals in nature have a chance of mating. So, at each iteration, we pick the  $i^{\text{th}}$  individual of the population, for  $i$  from 1 to current population size, then, applying artificial selection, the second individual is selected by ranking, like in the Genitor algorithm. In this way everyone has a chance of mating and bio-diversity is enhanced.

#### 4.5 Simple Mutation and Alterations at the Individual-Level

In ARGEN + AREPO mutation is performed according to the adaptive reproduction rate  $Pr$  and when it occurs, in the bit-string  $k$  bits are randomly flipped. The mutated individual does not replace the original one; simply it is ranked in the population, in case to be different from all individuals in the population, so the population size increases in one unit. When the population size reaches its maximum value, the offspring is ranked in the population, but only if it is different from all individuals in the population and, in that case the least fit individual is removed from the population. Crossover is performed according to the adaptive rate  $Pr$ , the alterations made between individuals is directed by the 2-point “reduced surrogate” [14] crossover operator, and if it occurred the resulting offspring would not replace their parents, they would be simply ranked in the population. In this way, population increases by two new elements. When the population reaches its maximum limit then the alterations at individual-level is as in the simple mutation.

## 5 Preliminary Results

For the data selection problem, the performance of the four HCs and all genetic algorithms here presented (including AREPO and ARGEN + AREPO) was evaluated in the wrapper model. The results for HCs are presented in Table 1 and the results for GAs in Table 2; the column “method” shows the searching algorithm used in the wrapper model. The stop criterion was to reach a maximum of 8000 evaluations of the fitness function, for all the searching techniques. In SGA and TGA a population of 50 individuals was used with a crossover rate of 0.8 and a mutation rate of 0.01. CHC uses the same number of individuals and a rate of 0.35 for the cataclysmic mutation. The population size in CF/RSC is 10, and, the individuals generated by each of them are 10. In GENITOR, AREPO and ARGEN + AREPO the population used was 1000 individuals. All datasets used in the experiments were taken from the Machine Learning Database Repository at the University of California, Irvine [7]. For each database 10 experiments were performed. At each experiment, 60% of data for training and 20% for evaluation were randomly selected from the corresponding database; both directed to the wrapper model. The remainder 20% was left to evaluate the accuracy of the feature and instance subset given by the wrapper model. There is an exception in Monks1, Monks2 and Monks3 because each of these datasets are divided in training and test subsets, thus, from the test subset 50% of data were randomly selected for

evaluation and the remainder for testing. The “Average” and “Best” columns from “Method Accuracy” show the average accuracy of the 10 data subsets given by the wrapper model over their respective evaluation subsets, and, the best accuracy obtained from the 10 experiments, respectively. The “Average” and “Best” columns from “Real Accuracy” show the average and best accuracy, respectively, of the 10 data subsets returned by the wrapper model, but, evaluated over their respective test subsets. Finally the “Storage” column shows the percentage of the original training set that is necessary to store to classify new instances, these results are also an average of the 10 experiments.

### 5.1 Automatic Infant Cry Recognition

Additionally, it was chosen an Infant Cry database to show the performance of ARGEN + AREPO in a different problem from those of the UCI Repository. The Automatic Infant Cry Recognition Process detects pathologies of recently born babies by means of their cry, it is very similar to Automatic Speech Recognition. It is divided in two main parts, the first one corresponds to signal processing, or acoustic feature extraction, and the second part is the pattern classification. The patterns are represented by vectors of Mel Frequency Cepstral Coefficients (MFCC). The used dataset has 1376 instances of 305 features each. There are three kinds of infant cry: normal, hypo-acoustics and asphyxia. The experiments with the infant cry dataset followed the same methodology applied to HCs and GAs. The results are presented in Table 3.

**Table 3.** Results at the classification of normal and pathological baby cry, this last divided in two classes: asphyxia, and hypoacoustics

| Database   | Method        | Real Accuracy (%) |       | Method Accuracy (%) |      | Storage |
|------------|---------------|-------------------|-------|---------------------|------|---------|
|            |               | Average           | Best  | Average             | Best |         |
| Infant Cry | SGA           | 98.88             | 99.64 | 99.64               | 100  | 24.68   |
|            | TGA           | 98.84             | 100   | 100                 | 100  | 12.94   |
|            | CHC           | 99.31             | 100   | 100                 | 100  | 18.43   |
|            | CFRSC         | 97.97             | 98.91 | 99.93               | 100  | 1.92    |
|            | GENITOR       | 99.24             | 100   | 100                 | 100  | 18.07   |
|            | AREPO         | 99.16             | 100   | 99.96               | 100  | 20.57   |
|            | ARGEN + AREPO | 98.19             | 99.64 | 100                 | 100  | 2.94    |

### 5.2 ARGEN + AREPO and Support Vector Machines

In all experiments to select feature and instance subsets, it was used the DW-KNN classifier, which shows a good performance. In order to test our algorithms with a different classifier we selected Support Vector Machines [3] (SVMs) which have gained much attention since their introduction. Experimental results show that SVMs can reach certain performance that is equal or higher than other classifiers, requiring less training data. Results of the ARGEN + AREPO algorithm combined with this classifier in a wrapper model to the feature and instance selection problem are presented in Table 4. In all experiments, the kernel function used was “*rbf*” (radial basis function), and the regularization constant “*C*” received a value of 10. We could see that in the “average” of the “real accuracy”, in five of eight databases, the classifier DW-KNN (Table 2) is better than SVMs (Table 4), although this last shows much better accuracy in those databases where DW-KNN is deficient.

## 6 Conclusions

In the wrapper model used to the feature and instance selection problem, the HCs algorithms have shown good performance, even over some GAs. In the experiments we could notice that performance of the training data subsets returned when using some GAs, obtain high accuracy while the wrapper model is applied, but, this accuracy falls down when new unseen instances are being classified. The reasons of this effect could be due to the mixing of overfitting [11] and the lack of exploration in the search process. The ARGEN + AREPO algorithm introduced in this work has shown to be a better option, to the feature and instance selection problem, using the DW-KNN classifier and the SVM classifier because in both, is reached a high accuracy keeping the least amount of training data.

**Table 4.** Feature and instance selection with the ARGEN + AREPO algorithm, but, a different classifier in the wrapper model and the test phase is used, SVMs

| Database           | Real Accuracy (%) |       | Real Accuracy (%) |       | Storage |
|--------------------|-------------------|-------|-------------------|-------|---------|
|                    | Average           | Best  | Average           | Best  |         |
| Breast Cancer (WI) | 92.23             | 94.24 | 98.14             | 99.29 | 5.19    |
| Glass              | 60.95             | 69.05 | 77.67             | 90.70 | 19.07   |
| Ionosphere         | 91.14             | 94.29 | 99.58             | 100   | 6.66    |
| Iris               | 95.67             | 100   | 99.67             | 100   | 3.17    |
| Liver Disorders    | 55.07             | 62.32 | 81.74             | 85.51 | 9.40    |
| Monks1             | 99.44             | 100   | 99.21             | 100   | 9.84    |
| Monks2             | 75.28             | 79.17 | 82.68             | 87.5  | 76.10   |
| Monks3             | 99.49             | 100   | 99.68             | 100   | 8.46    |

## References

- [1] Annunziato, Mauro., and Pizzuti, Stefano. "Adaptive Parameterization of Evolutionary Algorithms Driven by Reproduction and Competition". ESIT 2000, Aachen, Germany.
- [2] Bäck, Thomas. "Optimization by Means of Genetic Algorithms". E. Köhler, editor, 36th International Scientific Colloquium. p. 163-169. Technical University of Ilmenau. 1991.
- [3] Cortes, C., and Vapnik, V. "Support Vector Networks. Machine Learning", 20:1-25. 1995.
- [4] Davis, Lawrence. "Bit-Climbing, Representational Bias, and Test Suite Design". In L. Booker and R. Belew, editors, Proceedings of the Fourth International Conference on Genetic Algorithms. p. 18-23. Morgan Kaufmann, 1991.
- [5] Eshelman, Larry J. "The CHC Adaptive Search Algorithm: How to Have Safe Search When Engaging in Nontraditional Genetic Recombination". In G. Rawlins, editor, Foundations of Genetic Algorithms. p. 265-283. Morgan Kaufmann, 1991.
- [6] Guerra, Cesar, et al. "Fast and Accurate Feature Selection Using Hybrid Genetic Strategies". CEC99: Proceedings of the Congress on Evolutionary Computation. 1999.
- [7] Kohavi, Ron., and John, George H. "Wrappers for Feature Subset Selection". 1996. AIJ, issue on relevance, v. 97 (1-2). p. 273-324. 1997.
- [8] Kuncheva, Ludmila I., Jain, Lakhmi C. "Nearest Neighbor Classifier: Simultaneous Editing and Feature Selection". Pattern Recognition Letters, 20, 1999, 1149-1156.
- [9] Kuri, Á., and Galaviz, J.. Algoritmos Genéticos. IPN, UNAM & FCE., Mexico, 2002.
- [10] Michalewicz, Z. Genetic Algorithms + Data Structures = Evolution Programs. Springer-Verlag, Berlin, 1999.

- [11] Mitchell, Tom M. *Machine Learning*. McGraw Hill. New York. 1997.
- [12] Steinbrecher, Ricarda. “*What is Genetic Engineering?*”. *Synthesis/Regeneration: A Magazine of Green Social Thought*, Vol. 18. pp. 9-12. 1999.
- [13] Syswerda, G. “*A Study of Reproduction in Generational and Steady-State Genetic Algorithms*”. In G. Rawlins, editor, *Foundations of Genetic Algorithms*. p. 94-101. Morgan Kaufmann, 1991.
- [14] Whitley, Darrel. “*A Genetic Algorithm Tutorial*”. *Statistics and Computing (UK)*. 4(2). p. 65-85. Chapman & Hall. 1994.
- [15] Whitley, Darrell. “*The GENITOR Algorithm and Selective Pressure: Why Rank-Based Allocation of Reproductive Trials is Best*”. Technical Report CS-89-105. 1989.

# Modelling Perceptual Discrimination

Janet Aisbett<sup>1</sup>, James T. Townsend<sup>2</sup>, and Greg Gibbon<sup>1</sup>

<sup>1</sup>The University of Newcastle

<sup>2</sup>Indiana University

**Abstract.** Accounts of perceptual decision making, such as evidence accrual models, represent mental states as noisy numerical vectors describing the stimuli. As such, these are not biological models. An alternative scheme is presented in which mental states are represented as functions. This generalises an analogue coding scheme for numbers, and might be biologically implemented as functions of cortical activity. Some properties of this representation are illustrated in modelling accuracy and response time patterns observed in a classic experiment into perceptual processes.

## 1 Introduction

This paper concerns representation in the context of perceptual discrimination tasks, such as selection of the brighter of two lights. In these tasks, the mental representation is assumed to change with time since the stimuli were presented. A response is made when the representation supports a particular selection, or when a guess is forced. Modelling of discrimination tasks must therefore attempt to explain time to respond, as well as response accuracy.

This paper presents a model of perceptual decision making in which mental representations are a vector space of real-valued functions defined on an  $n$ -dimensional manifold -- such functions we call *spatial maps*. A property of a stimulus is modelled as the restriction of the function representing the perceived stimulus to a subset of the manifold. As biological motivation, one could think of the functions as some physical function of neural activity defined on a cortical layer, for example, an electromagnetic field (McFadden 2002).

Conventionally, stimuli are represented as numerical vectors, each component of which summarises a perceived quality or property. Component values may be treated as observations of a random variable that provides noisy estimates of the property or quality, where the noise might be external or internal to the decision-making system being modelled. Decision boundaries divide the vector space into disjoint sets, each associated with a different response. For example, the simplest task involving two stimulus classes, each perceived via Gaussian-distributed observations on a single dimension, has been described by signal detection theory (Townsend & Ashby 1983). The decision maker cannot know for certain which class any particular observation came from, but can set a threshold on perceived stimulus values and respond according to whether the perceived value is above or below the threshold.

A variety of evidence accrual models are used to explain response time in perceptual decision making, that is, the time at which the decision is made relative to presentation of the stimuli. (See Ratcliff and Smith (2004) for a comparison of major models.) In modelling a two-choice decision involving noisy one-dimensional stimuli, a single accumulator may be assumed to gather evidence in the form of a sum of differences between repeated noisy observations of the stimuli, or alternatively a pair of accumulators each gathers evidence for one stimuli. In such discrete accumulator models, the number of observations  $N$  is assumed to increase linearly with time. A response is made when the difference between the sums lies outside a no-decision region around zero (the decision boundary); or when one of a pair of accumulators reaches its threshold; or when the time available expires.

Conventional models of perceptual decision making are not biological models. Perceptual response time effects have been emulated in neural net models (eg. Brown & Heathcote, in press). These nevertheless employ numerical vectors to represent perceived stimuli.

The purpose of our paper is threefold. Firstly, it shows how the biologically motivated spatial maps might represent stimuli with measurable physical properties. Secondly, it demonstrates how a psychometric function could have a formulation not available if numerical vectors were used to represent perceived stimuli. Finally, it applies the model to explain data from a classic discrimination experiment (Vickers & Packer 1982).

## 2 Spatial Map Representations

Aisbett & Gibbon (2001, 2003) have previously proposed real valued functions as a flexible representation scheme. Townsend, Solomon and Spencer-Smith (2001) and Townsend and Spencer-Smith (2003) have investigated geometric and topological properties of certain families of functions, arguing that such representations are needed to simulate more complex psychological processing. They suggest that for some psychological modelling, internal states should be represented as a function on a low dimensioned manifold, say  $n$ -dimensional for  $n \leq 3$ .

The relationship between measurable values of a physical stimulus and representation based on functions has not previously been investigated. This section does this.

It is instructive to reflect first on ANN modelling of perceptual decision making. Input to networks designed to model early perceptual processing are sampled physical signals, usually 2-D picture arrays, or audio traces. Processing is aimed at reducing the dimensionality of the input, to produce activation on nodes that correspond to dimensions such as redness, angle or size. There may be one output node from the early processing system for each perceptual dimension, or there may be a set of nodes in which value is coded by the ordering of the nodes in the set, as well as by nodal activation.

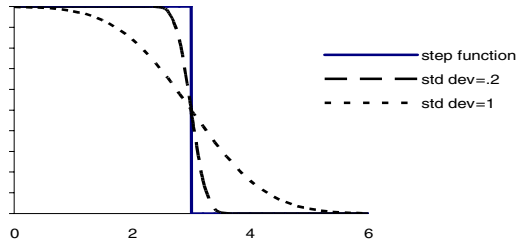
If order is used to code value, coding may be either place-based or thermometer. Place based coding schemes are familiar from Arabic and other number systems, as well as from binary coding of integers. Thermometer coding provides alternative “analog” representations of real numbers or intervals (eg. Smith 1993). It is robust in

the presence of noise, at the cost of redundancy. Thermometer coding of integers on input nodes has been shown to improve the ability of feed forward neural networks to model categorical perception (Tijsseling & Harnad 1997).

Thermometer coding provides an insight into how qualities that have magnitude, such as length or brightness, might be modelled using spatial maps.

Thermometer coding of a set of numbers  $N$  can be based on an arbitrary set  $X$  provided that there are a set of sets  $\{X_b \subset X\}_{b \in N}$  which satisfy the relationship

$$X_a \subset X_b \text{ for any pair } a, b \text{ in } N \text{ with } a < b. \tag{1}$$



**Fig. 1.** Spatial maps on  $R^+$  representing a noiseless reading (step function) and two noisy readings (shown as erfc or complementary cumulative distribution functions of a Gaussian)

An equivalent coding of the numbers in  $N$  can be defined using a set of characteristic or membership functions  $\chi_b$  on  $X$ , where  $\chi_b(x) = 1$  if  $x \in X_b$ ;  $\chi_b(x) = 0$  if  $x \notin X_b$ . The coding can be generalised by allowing the functions  $\chi_b$  to take values in the interval  $[0, 1]$ . If  $\chi_b(x)$  is between 0 and 1 then  $x$  can be interpreted as being possibly in the set representing  $b$ . This allows imprecision in perceived values to be modelled directly in the internal representation, rather than as an external probability distribution. Figure 1 illustrates modelling two noisy readings of a value  $b$  using the complement to a Gaussian cumulative distribution function, where  $X$  is the positive reals. This contrasts with the noise free step function representing the analog thermometer coding of  $b$ , that is, a function  $f$  on the non-negative reals such that  $f(x) = 1$  for  $x < b$  and  $f(x) = 0$  for  $x \geq b$ .

A further generalisation of thermometer coding extends the set of functions that are used to represent values on a perceptual dimension, from sets of functions indexed by numbers in  $N$  to other sets of functions from  $X$  into the unit interval. It also extends the range of the functions to the non-negative reals,  $R^+$ , to allow addition.

This leads us to the general definition of internal representations as members of a real vector space  $\Phi$  of functions from an  $n$ -manifold  $X$  into  $R$ . If  $X$  is an  $n$ -dimensional manifold, such as a subset of  $R^n$ , and the functions are continuous, the members of  $\Phi$  can be thought of as *spatial maps*. Continuity and other conditions that may be imposed on  $\Phi$  mean the set of internal representations is richer than an arbitrary set of vectors having one component for each member of  $X$ .

Dimensions are defined as (connected) subsets  $X_i$  of  $X$ . A value on a dimension specified by  $X_i$  is the restriction  $f|_{X_i}$  of a function  $f$  in  $\Phi$ . If subsets  $X_i$  and  $X_j$  are disjoint, the dimensions are separable. If the sets intersect, interaction effects occur and the dimensions are so-called integral. The ability to represent these dimensional interactions are an advantage of the spatial map formulation (Aisbett & Gibbon 2001).

A family  $\Phi_n$  defined on a subset  $Y$  of  $R^n$  can be associated with any family  $\Gamma$  of probability density functions on a real (possibly infinite) interval  $[0, a)$ . Define the embedded complementary cumulative distribution function (CCDF)  $f$  of a distribution  $p$  in  $\Gamma$  by  $f(x) = \int_{|y|>x} p(y)dy$ . This is a radially symmetric function, taking the CCDF of  $p$  along each radial line. Form  $\Phi_n$  from the basis set of embedded CCDFs. Suppose that the  $n$ -dimensional sphere of radius  $a$  is contained in  $Y$ . Then if  $f$  is the embedded CCDF corresponding to  $p \in \Gamma$ , substituting for  $f$  and integrating by parts shows that  $\int_Y f(x)dx$  is the expected value  $\int_Y x p(x)dx$  multiplied by the volume of the unit sphere in  $R^n$ . The family  $\Phi_n$  therefore naturally models noisy mental representations of the physical magnitudes of a stimulus quality. This is illustrated in Figure 1 for  $n=1$ . (Note that if noise were modelled pointwise as observations of a random variable (so that the value at any point  $x$  in  $Y$  is  $f(x) + \epsilon$ ) then the noisy function that results will not generally be a member of  $\Phi_n$  even when  $f$  is. It will not be continuous even when  $f$  is continuous.)

Mental representations are time varying. The notation  $f_t(x)$  will be used to denote the value of a spatial map  $f$  at time  $t$  on the point  $x$  of  $X$ . In this paper, time is treated as discrete, although this assumption is not essential. Section 4 introduces additional assumptions needed to model the time course of representations in a discrimination task.

### 3 Decisions and Response Time

Any choice decision involves some notion of comparative similarity. We therefore require that  $\Phi$  have an  $L^1$  norm, denoted  $\| \cdot \|_X$ . Then the distance between a pair of perceived stimuli  $f$  and  $g$  is defined to be

$$d(f, g; X) = \|f - g\|_X \equiv \int_X |f(x) - g(x)|dx. \tag{2}$$

For example, the distance between the step function and one of the erfc functions depicted in figure 1 is the sum of the roughly triangular regions which meet at the intersection of the step and erfc functions.

The integrability required to define the  $L^1$  norm is a natural biological condition, as physical activity of neural populations can be aggregated over subvolumes.

The norm in (2) restricts to subsets  $Y$  of  $X$  where it will be denoted  $\| \cdot \|_Y$ . On dimensions it can be interpreted as the magnitude of the stimulus on that dimension. (The integral may not be defined for some pathological subsets that are not of interest here.) The distance between perceived stimulus values  $f$  and  $g$  on a dimension is the norm  $\|f - g\|_{X_i}$  on the subset  $X_i$  defining that dimension.

For  $X$ , or for any subset  $Y$  of  $X$ , a partial order can be defined, namely

$$f <_Y g \text{ if and only if } f(x) < g(x) \text{ for all } x \in Y \tag{3}$$



This is obviously not a total order. An  $n$ -dimensional real space does not have a total order either, but this goes further in allowing that a single dimension does not have a total order. Thus, on a single dimension,  $\|f\|_{X_i} <_{X_i} \|g\|_{X_i}$  would not necessarily mean that stimulus  $f|_{X_i}$  was perceived to be smaller than  $g|_{X_i}$ .

Turning now to the modelling of a discrimination task: suppose two stimuli originally presented at time  $t=0$  have mental representations  $f_t$  and  $g_t$  at time  $t$ . We consider only the simple task of deciding which stimulus is “biggest”. There might nevertheless be multiple dimensions of the stimuli which are relevant to the task instructions, as for example when selecting the largest size based on both height and width. Suppose the relevant qualities of the stimuli are defined on a subset  $Y$ .

If the mental representations of  $f_t$  and  $g_t$  cannot be ordered, then there may be no natural response. However, it may be impractical to wait till one function exceeds the other everywhere on  $Y$ : indeed, this might never occur. We therefore propose that the response is made at the earliest period  $t$  for which the following so-called *stopping rule* is satisfied:

$$\| \|f_t\|_{X_i} - \|g_t\|_{X_i} \| / \|f_t - g_t\|_{X_i} = | \int_{X_i} f_t(x) - g_t(x) dx | / \int_{X_i} |f_t(x) - g_t(x)| dx > K, \quad (4)$$

for  $K \leq 1$  is a task-dependent parameter.  $K$  will be larger when the task requires accuracy rather than speed, for example. Note that if numbers were used rather than functions on a measurable set, equation (4) is always satisfied so it cannot be employed in determining response time.

Having waited until one function appears likely to exceed the other, the response at that time  $t$  will be made according to whether or not

$$\| \|f_t\|_Y <_Y \|g_t\|_Y. \quad (5)$$

## 4 Modelling a Two-Choice Discrimination Task

This section reinterprets data reported by Vickers and Packer (1982) on experiments investigating models of confidence in perceptual discrimination tasks. The experimental stimuli were pairs of different length bars, displayed on a screen. The length of the bars was constant throughout a trial, but the position of the longer bar varied randomly between right and left. The longer bar was always the same length, but the shorter bar took one of 5 lengths, resulting in 5 different levels of discriminability in the conditions of the experiment. Participants chose which side had the longer bar, and assigned a confidence rating to their selection. In trial blocks, participants were either instructed to be as accurate as possible, or to be fast while trying to be accurate (speed condition).

As expected, the percentage of correct responses fell and the time to respond increased as the level of discriminability decreased (see the top two plots in Figure 2, reproduced from figure 2 of Vickers & Packer). The results also exhibit a well known phenomenon called the speed/accuracy tradeoff – if participants are hurried making a decision, they are less accurate.

We therefore have to model the tendency to develop more accurate mental representations of stimuli magnitudes with time. We do this by changing the relationship between the representation and the magnitude, which in section 2 we proposed to be based on CCDFs. So suppose the representation of a stimulus of

magnitude  $L$  is, at time  $t$ , the embedded CCDF of a distribution defined on  $R^+$  with mean  $m(t)$  and variance  $v(t)$ . Suppose initially that  $m(0)$  is an observation of a normal distribution with mean the true stimulus value, and variance  $s$ . Then to model convergence in the estimate of  $L$  as well as reducing noise  $v$ , set

$$m(t) = |w.m(t-1) + (1-w)(L - m(t-1)) + n(t)|; \text{ and } v(t) = |m(t)-L|^2. \tag{6}$$

Here,  $n$  is zero-mean Gaussian noise, and  $w$  is from a uniform distribution on  $[0, 1]$ .

Six parameters are needed to model the physical stimuli lengths over all levels, 2 stopping rule criteria  $K$  are used for speed or accuracy conditions, and 1 parameter  $s$  is needed for the variance in starting point  $m(0)$ . There is also an implicit parameter used to scale time to fit the experimental data.

The simulated results when the CCDFs are those of truncated normal distributions. are shown in the lower plots in Figure 2. They fit the actual data closely.

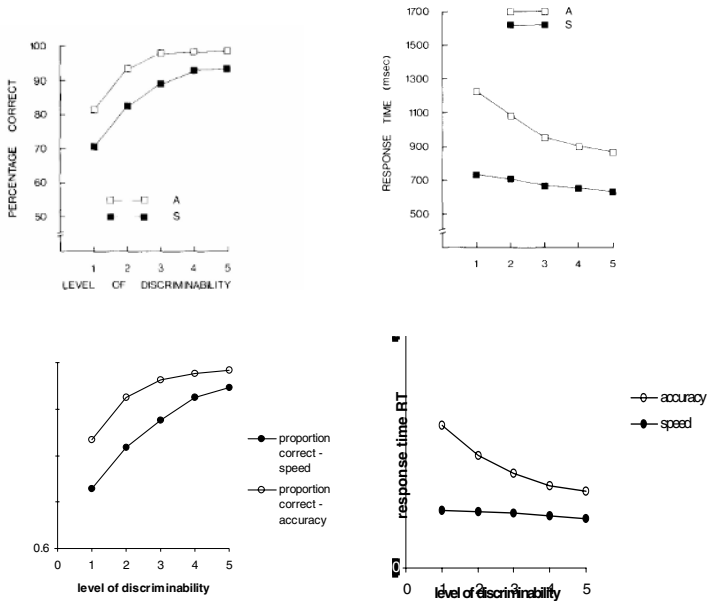


Fig. 2. Top plots – experimental results from Vickers & Packer (1982) for average accuracy and response time against difficulty of task. Bottom plots – simulated results. See text

## 5 Discussion

This paper presented a representational scheme based on families of functions we called spatial maps, and illustrated its application to perceptual discrimination. Functional representations had been suggested earlier by Townsend and colleagues, and explored in the context of shape representation. The present scheme was motivated by analogue coding of numbers and was not specialised for any property or quality types.

The advantage of spatial map representation over direct modelling of physical magnitudes as real numbers is the ability to capture noise in the representation of the stimulus, rather than with ancillary modelling structures. We suggested embedding 1-dimensional CCDFs in 2 (or higher) dimensions as radially symmetric functions. These carry information about the uncertainty in the estimated value which is coded as the mean. While embedded CCDFs are inefficient, they are biologically realisable in 2 or, more correctly, 3 dimensions. We modelled accumulation over time as convergence of the CCDFs to a step function at the true value, allowing for noise that is proportional to the error in the estimate.

The reanalysis of experiments by Vickers and Packer (1982) showed how a psychometric function, the response time, depended on a stopping rule that could not be formulated if numerical vector representations were assumed. Further reanalysis of published experimental results will be fruitful in developing insights into the extent to which the conventional use of numerical vectors to model mental representations limits potential interpretations of mental activity. Targeted experiments to distinguish between use of spatial representation and numerical representations in perception are needed.

A more detailed mathematical specification of the families used in the representation is required to ensure all properties specified in modelling processes are available, while constraining the families as far as possible. The link between psychometric modelling and biological modelling of psychological phenomena also bears further work. Biological models are usually equated with connectionist models. We have argued against this because of the simplicity of the connectionist modelling, and suggested that functional representations may be as valid. To sustain this argument the nature of the functions and the set on which they are defined need to be explored and identified neurophysiologically.

## References

- J. Aisbett and G. Gibbon (2001) Conceptual spaces as voltage maps. In J. Mira and A. Prieto, *Connectionist Models of Neurons, Learning Processes, and Artificial Intelligence*. Lecture Notes in Computer Science 2084, Springer Verlag, 783-790.
- Aisbett J. & Gibbon, G (2003) Preserving similarity in representation: a scheme based on images *Proceedings Joint International Conference on Cognitive Science Sydney*
- Brown, S. and Heathcote, A. (in press). A ballistic model of choice response time. *Psychological Review*.
- McFadden, J (2002) Synchronous firing and its influence on the brain's electromagnetic field: evidence for an electromagnetic theory of consciousness. *Journal of Consciousness Studies* 9, 4, 23-50.
- Ratcliff, R and Smith, P. (2004) A comparison of sequential sampling models for two-choice reaction time. *Psychological Review* 111,2,333-367
- Smith, M (1993) *Neural networks for statistical modeling*. Van Nostrand Reinhold.
- Tijsseling, A. and Harnad, S. (1997) Warping Similarity Space in Category Learning by Backprop Nets. In *Proceedings of SimCat 1997: Interdisciplinary Workshop on Similarity and Categorization*. (Eds Ramscar, Hahn, Cambouropoulos & Pain)
- Townsend, J.T. and Ashby, F. (1983) *Stochastic modeling of elementary psychological processes*. Cambridge University Press.

- Townsend, J. T, Solomon B & Spencer-Smith, J. (2001) The perfect Gestalt: Infinite dimensional Riemannian face spaces and other aspects of face perception, *Computational geometric and process perspectives on facial cognition* (Eds. M. Wenger M and J. Townsend) 39-82.
- Townsend, J. and Spencer-Smith, J. (2003) Two kinds of global perceptual separability and curvature. *Psychophysics beyond sensation: Laws and invariant of human cognition.* (Ed. C. Kaernbach, E. Shroger, & H. Muter) Erlbaum.
- Vickers, D (1970) Evidence for an accumulator of psychophysical discrimination. *Ergonomics*, 13, 37–58.
- Vickers, D and Packer, J. (1982) Effects of alternating set for speed and accuracy on response time, accuracy and confidence in a unidimensional discrimination task *Acta Psychologica* 50, 179-197.

# Memory Retrieval in a Neural Network with Chaotic Neurons and Dynamic Synapses

Zhijie Wang<sup>1</sup> and Hong Fan<sup>2</sup>

<sup>1</sup> College of Information Science and Technology,  
Donghua University, 200051, Shanghai, China  
wangzj@dhu.edu.cn

<sup>2</sup> Glorious sun School of Business and Management,  
Donghua University, 200051, Shanghai, China  
hongfan@dhu.edu.cn

**Abstract.** An associative neural network with chaotic neuron model and synaptic depression (CSDNN) is constructed. Memory switching phenomenon in the network is demonstrated. Simulation results show that with various parameter value settings and with various initial conditions, the memory retrieval frequency of CSDNN distributes uniformly among the stored patterns, and the rate of memory retrieval of CSDNN is much higher than that of a chaotic neural network. The possible utilization of memory retrieval in CSDNN is also discussed.

## 1 Introduction

It is well known that an autoassociative neural network can store memories through its synaptic connections [3][4]. Unlike a conventional autoassociative neural network which evolves to settle at a stable steady state, a chaotic associative neural network generates peculiar behavior that the network retrieves some stored patterns and escapes from them in the transient phase [1][2][5]. Such a chaotic associative memory is constructed with chaotic neuron models interconnected through a conventional auto-associative matrix of synaptic weights. On the other hand, once the weights of the synapses in a conventional autoassociative neural network are learned by learning rules, they are fixed during the memory retrieval process. This implies that the synapses of the neurons are assumed to be 'static', i.e., that they change their weights only on the slow time scale. However, it has been discovered that synaptic plasticity occurs across many time scales—from the order of days to the order of milliseconds [10]. The dynamic association in a neural network when a synaptic depression is included in the network is investigated [7]. Similar to a chaotic associative memory, the output of such a network transits among the stored patterns during the retrieval process.

In this paper, we explore the dynamics of an associative neural network with chaotic neuron models and synaptic depression (CSDNN). We also find the memory switching phenomena in this kind of network. More importantly, we find that the retrieval frequency of the stored pattern in CSDNN is much higher than that in a cha-

otic neural network [1]. Stated in other way, after CSDNN escapes from a stored pattern, it spends much less time to evolve to another stored pattern than a chaotic neural network. Therefore, if the associative dynamics is interpreted as a memory searching process, CSDNN is a quick memory searching network. We also make a discussion in the last of this paper that CSDNN might be a potential good mean for solving combinatorial optimization problems.

## 2 Models of the Neural Network with Chaotic Neurons and Dynamic Synapses

The chaotic neuron model in this paper is described by the following equation [1] [2]:

$$x(t + 1) = f[A(t) - \alpha \sum_{d=0}^t k^d g\{x(t - d)\} - \Theta], \tag{1}$$

where  $t$  is a discrete time step ( $t = 0, 1, 2, \dots$ ),  $x(t)$  is the neuronal output with an analog value between 0 and 1 at the discrete time  $t$ ,  $f$  is the activation function,  $A(t)$  is the external stimulations at the time  $t$ ,  $g$  is the refractory function,  $\alpha$  is the refractory scaling parameter,  $k$  is the refractory decay parameter, and  $\Theta$  is the threshold.

The term  $k^d g\{x(t - d)\}$  represents the influence of the refractoriness due to the neuronal output of  $d$  time steps ago.

An internal state of the neuron can be defined as follows:

$$y(t + 1) = A(t) - \alpha \sum_{d=0}^t k^d g\{x(t - d)\} - \Theta. \tag{2}$$

A reduced difference equation describing dynamics of the internal state can be represented as follows:

$$y(t + 1) = ky(t) - \alpha g[f\{y(t)\}] + \theta(t), \tag{3}$$

where  $\theta(t) \equiv A(t) - kA(t - 1) - \Theta(1 - k)$ .

When we set  $g(x) = x, \theta(t) = a$  which is temporally constant, and  $f(y) = 1 / \{1 + \exp(-y / \mathcal{E})\}$ , which is the logistic function with the steepness parameter  $\mathcal{E}$ , the model shows not only periodic response but also chaotic response according to the parameter values [1] [2].

The chaotic neural network is composed of the chaotic neurons shown above. The dynamics of the  $i$  th chaotic neuron integrated in a chaotic neural network is described as follows [2]:

$$x_i(t + 1) = f \left[ \sum_{j=1}^M v_{ij} \sum_{d=0}^t k_e^d A_j(t - d) + \sum_{j=1}^N w_{ij}(t) \sum_{d=0}^t k_f^d x_j(t - d) - \alpha \sum_{d=0}^t k_r^d g\{x_i(t - d)\} - \Theta_i \right] \tag{4}$$

where  $v_{ij}$  is the synaptic weight to the  $i$  th neuron from the  $j$  th external input,

$w_{ij}(t)$  is the synaptic weight to the  $i$  th neuron from the  $j$  th neuron,  $k_e$  is the decay parameter for the external inputs,  $k_f$  is the decay parameter for the feedback inputs, and  $k_r$  is the decay parameter for the refractoriness.

Equation 4 can be transformed into the following reduced and simultaneous forms [2]:

$$\xi_i(t+1) = \sum_{j=1}^M v_{ij} A_j(t) + k_e \xi_i(t), \tag{5}$$

$$\eta_i(t+1) = \sum_{j=1}^N w_{ij}(t) x_j(t) + k_f \eta_i(t), \tag{6}$$

$$\zeta_i(t+1) = -\alpha g\{x_i(t)\} + k_r \zeta_i(t) - \theta_i, \tag{7}$$

$$\theta_i \equiv \Theta_i(1 - k_r),$$

$$x_i(t+1) = f\{\xi_i(t+1) + \eta_i(t+1) + \zeta_i(t+1)\}. \tag{8}$$

In this paper, the chaotic neural network is expressed as a simplified form of Eqs.9-11:

$$x_i(t+1) = f\{\eta_i(t+1) + \zeta_i(t+1)\}, \tag{9}$$

$$\eta_i(t+1) = \sum_{j=1}^N w_{ij}(t) x_j(t) + k_f \eta_i(t), \tag{10}$$

$$\zeta_i(t+1) = k_r \zeta_i(t) - \alpha x_i(t) + a_i, \tag{11}$$

where  $a_i$  denotes the sum of the threshold and the temporally constant external input to the  $i$  th neuron.

The weight of the dynamic synapse is determined by two parts as follows [7]:

$$w_{ij}(t) = w_{ij}^c \cdot r_j(t), \tag{12}$$

where  $w_{ij}^c$  is a static term that is determined by the stored patterns;  $r_j(t)$  is the term that is caused by the synaptic depression.

$w_{ij}^c$  is defined as follows:

$$w_{ij}^c = (1/N_s) \sum_{p=1}^{N_s} (2\mu_i^p - 1)(2\mu_j^p - 1), \tag{13}$$

where  $N_s$  is the number of the stored patterns;  $\mu_i^p$  is the  $i$  th component of the  $p$  th stored pattern. The four stored patterns is shown in Fig. 1 (see [1] for details).



**Fig. 1.** The four stored patterns. The output pattern of the 100 neurons is displayed in the form of a 10 by 10 matrix

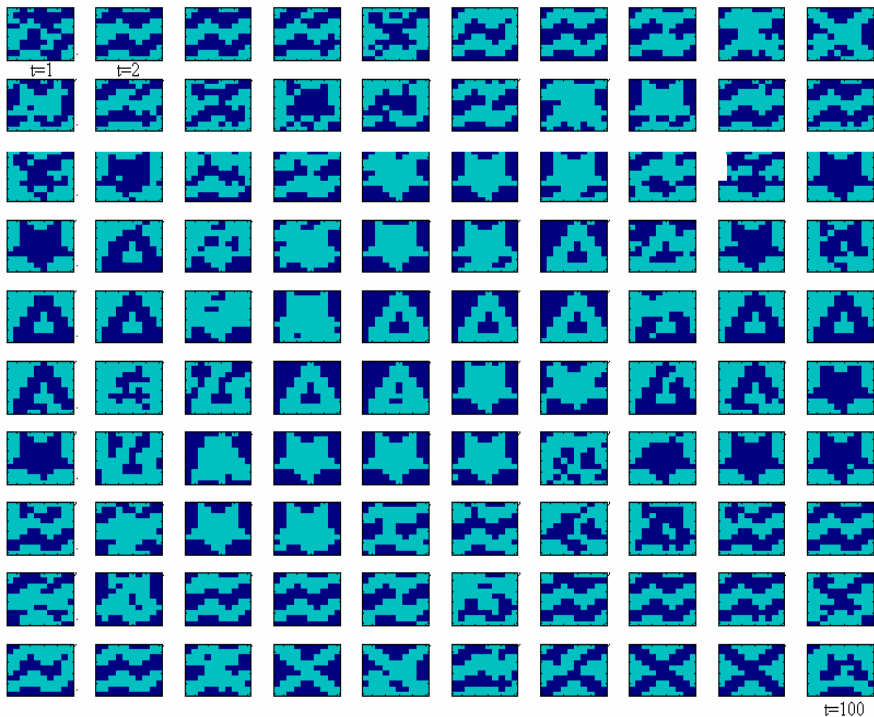
The model of the synaptic depression can be described as follows (see [7] for details):

$$r_i(t + 1) = r_i(t) + \Delta t \cdot (1 - r_i(t) / \tau - U \cdot r_i(t) \cdot x_i(t)), \tag{14}$$

where  $r_i(t)$ , which affects the weight of the dynamic synapse (see Eqn. 12), is the recovered synaptic resources of the neuron  $i$ ;  $\tau$  is the recovery constant; the term  $U \cdot r_i(t)$  represents a fraction of the recovered synaptic resources of the neuron  $i$ ;  $\Delta t$  is the time step of discretization.

### 3 Simulation Results



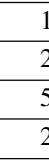
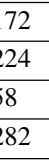
CSDNN is composed of chaotic neurons and dynamic synapses. There are two groups of parameters. The group of  $\varepsilon, \alpha, k_f, k_r$  and  $a_i$  is for chaotic neuron models, the other group of  $\tau$  and  $U$  is for synaptic depression. As the dynamics of the chaotic-neural network [1] can be studied by changing the parameter of the refractoriness  $k_r$ , and the effect of the depression of the dynamic synapses can be adjusted by the pa





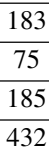

**Fig. 2.** Memory retrieval of CSDNN under the parameters of  $\varepsilon = 0.015, \alpha = 6, k_f = 0.1, k_r = 0.6, a_i = 2, \tau = 10$  and  $U = 0.08$ . The time step  $t$  is from 0 to 100





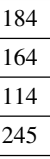

**Table 1.** Memory (stored patterns and reverse stored patterns) retrieval frequency with  $U$  fixed at 0.01 and  $k_r$  increased from 0.7 to 0.8. The total time steps of the simulation is 5000

| $k_r$                                                                             | 0.7    |         | 0.75   |         | 0.8    |         |
|-----------------------------------------------------------------------------------|--------|---------|--------|---------|--------|---------|
| Pattern                                                                           | Stored | Reverse | Stored | Reverse | Stored | Reverse |
|  | 172    | 200     | 141    | 185     | 188    | 195     |
|  | 224    | 422     | 270    | 270     | 156    | 169     |
|  | 58     | 229     | 183    | 365     | 118    | 167     |
|  | 282    | 310     | 184    | 185     | 241    | 205     |
| $R$                                                                               | 0.38   |         | 0.36   |         | 0.29   |         |

**Table 2.** Memory (stored patterns and reverse stored patterns) retrieval frequency with  $k_r$  fixed at 0.65 and  $U$  increased from 0.07 to 0.1. The total time steps of the simulation is 5000

| $U$                                                                               | 0.07   |         | 0.09   |         | 0.08   |         | 0.1    |         |
|-----------------------------------------------------------------------------------|--------|---------|--------|---------|--------|---------|--------|---------|
| Pattern                                                                           | Stored | Reverse | Stored | Reverse | Stored | Reverse | Stored | Reverse |
|  | 183    | 221     | 91     | 92      | 91     | 90      | 167    | 166     |
|  | 75     | 223     | 90     | 180     | 197    | 144     | 125    | 166     |
|  | 185    | 219     | 46     | 91      | 54     | 92      | 171    | 170     |
|  | 432    | 255     | 182    | 45      | 288    | 183     | 208    | 86      |
| $R$                                                                               | 0.36   |         | 0.16   |         | 0.23   |         | 0.26   |         |

**Table 3.** Memory (stored patterns and reverse stored patterns) retrieval frequency in four runs. Each run starts with randomly generated initial conditions.  $k_r$  is fixed at 0.8 and  $U$  is fixed at 0.01. The total time steps of the simulation is 5000

|                                                                                     | Run 1  |         | Run 2  |         | Run 3  |         | Run 4  |         |
|-------------------------------------------------------------------------------------|--------|---------|--------|---------|--------|---------|--------|---------|
| Pattern                                                                             | Stored | Reverse | Stored | Reverse | Stored | Reverse | Stored | Reverse |
|  | 184    | 199     | 174    | 187     | 174    | 191     | 183    | 193     |
|  | 164    | 175     | 162    | 173     | 154    | 169     | 165    | 176     |
|  | 114    | 157     | 120    | 163     | 128    | 172     | 113    | 161     |
|  | 245    | 208     | 238    | 206     | 243    | 215     | 245    | 211     |
| $R$                                                                                 | 0.29   |         | 0.28   |         | 0.28   |         | 0.29   |         |

parameter of  $U$  in Eqn. 14, we explore the characteristic of the memory retrieval behavior of the CSDNN by changing these two parameters. If the refractoriness and the depression of the dynamic synapses are both strong, the behavior of CSDNN seems somewhat like random. Therefore, we fix one of the two parameters at a small value, increasing the other parameter from a small value to a large value. Other parameters

are fixed following related literatures [1][7]. We fix  $\varepsilon = 0.015$ ,  $\alpha = 6$ ,  $k_f = 0.1$ ,  $a_i = 2$  and  $\tau = 10$  in this paper.

Fig .2 shows an example of the memory retrieval process of CSDNN. CSDNN visits a stored pattern (reversed stored pattern), escapes from the stored pattern, searches for next stored pattern, and once again visits another stored pattern. The main subject in this paper is to explore the memory (stored pattern and reverse stored pattern) retrieval frequency, and Tab .1, Tab .2 and Tab .3 show the simulation result. Tab .1 is the retrieval frequency of the stored patterns and reverse stored patterns with  $U$  fixed at 0.01 and  $k_r$  increased from 0.7 to 0.8. Tab .2 is the memory retrieval frequency with  $k_r$  fixed at 0.65 and  $U$  increased from 0.07 to 0.1. Tab .3 is the memory retrieval frequency with four randomly generated initial conditions. To characterize the memory retrieval frequency of CSDNN, we define the rate of memory retrieval as follows:

$$R = \frac{\text{the time steps of the network visiting the stored patterns}}{\text{the total time steps of the simulation}} \quad (15)$$

We decide that the neural network is visiting a stored pattern (reverse stored pattern) if the Hamming distance of the output pattern of the neural network equals that of the stored pattern exactly.

Tab .1, Tab .2 and Tab .3 show that with various parameter value settings and with various initial conditions, the retrieval frequency of CSDNN distributed uniformly among the four stored patterns, and the rate of memory retrieval in CSDNN is around 0.29, which is much higher than that in a chaotic neural network [1]( around 0.03).

## 4 Discussion

A neural network with chaotic neurons and dynamic synapses (CSDNN) is constructed. The associative dynamics in such a network is similar to that in chaotic neural networks [1], but CSDNN transits more quickly among stored patterns than chaotic neural networks. Since the associative properties of chaotic neural network are similar to dynamical behavior of the olfactory system [11] and may play a functional role of rapid and unbiased access to the learned patterns in the olfactory system, the quick memory search property of CSDNN might be very important for living beings, especially as the external environment for them change rapidly.

The quick memory search property of CSDNN might also be utilized in technical field. Neural network approaches have been shown to be powerful tools for combinatorial optimization, but the conventional neural network approaches suffer from the local minimum problems. Neural networks combined with simulated annealing techniques have a globally searching ability for combinatorial optimization problems [8][9], but they suffer from other drawbacks such as efficiency. The quick memory search property of CSDNN might be helpful for addressing these problems. The stored patterns can be viewed in some sense the local or global minima of the energy function of the network. The quick memory transition means that the network can escape from local minima quickly once it traps in the local minima. If global mini-

imum of the energy function of the network is viewed as a memory, the quick memory transition also means that the network can find the global minimum quickly. Thus CSDNN may be a potential good mean for solving optimization problems in technical field.

## References

1. Adachi, M., Aihara, K.: Associative dynamics in chaotic neural network, *Neural Networks*, Vol.10, No.1. (1997) 83-98
2. Aihara, K., Takabe, T., and Toyoda, M.: Chaotic neural networks. *Physics Letters A*, Vol.144, No.6,7. (1990) 333-340
3. Hopfield, J.J. : Neural networks and physical systems with emergent collective computation abilities. *Proc. Nat. Acad. Sci, USA*, 79(1982) 2445-2558
4. Nakano, K.: Associatron-a model of associative memory. *IEEE Transactions, SMC-2*. (1972) 381-388
5. Wang, Z.J.and Aihara, K.: A Fuzzy-Like phenomenon in chaotic autoassociative memory, *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, Vol. E85-A, No.3. (2002) 714-722
6. Wang, Z.J.and Aihara, K.: A Fuzzy-Like phenomenon in in a Dynamic Neural Network, *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, Vol. E86-A, No.8. (2003) 2125-2135
7. Pantic, L., Torres, J.J. and Kappen H.J.: Associative memory with dynamic synapses, *Neural Computation*, Vol. 14, No. 12. (2002) 2903-2923
8. Chen, L. N., Aihara K.: Chaotic simulated annealing by a neural network model with transient chaos, *Neural Networks*, Vol.8, No.6 (1995) 915-930
9. Chen, L. N., Aihara K.: Chaotic dynamics of neural networks and its application to combinatorial optimization, *Differential Equations and Dynamical Systems*, Vol. 9, No. 3-4. (2001) 139-168
10. Maass, W., and Bishop, C.M., (eds.): *Pulsed Neural Networks*, The MIT Press (1998)
11. Freeman, W. J.: Tutorial on neurobiology: from single neurons to brain chaos, *International journal of bifurcation and chaos*, Vol.2. (1992) 451-482

# Neural Network Based 3D Model Reconstruction with Highly Distorted Stereoscopic Sensors

Wan-liang Wang<sup>1</sup>, Bing-bing Xia<sup>1</sup>, Qiu Guan<sup>1</sup>,  
and Shengyong Chen<sup>1,2,\*</sup>

<sup>1</sup> College of Information Engineering, Zhejiang University of Technology,  
Hangzhou 310014, China

{wwl, gq, csy }@zjut.edu.cn, xiabing214@sohu.com

<sup>2</sup> National Laboratory of Pattern Recognition of Automation Institute,  
Chinese Academy of Sciences, China

sy@ieee.org

**Abstract.** In stereoscopic vision, there are two artificial eyes implemented so that it can obtain two separate views of the scene and simulate the binocular depth perception of human beings. Traditionally, camera calibration and 3D reconstruction of such a vision sensor are performed by geometrical solutions. However, the traditional camera model is very complicated since nonlinear factors in it and needs to approximate the light projection scheme by a number of parameters. It is even very difficult to model some highly distorted vision sensors, such as fish-eye lens. In order to simplify both the camera calibration and 3D reconstruction procedures, this work presents a method based on neural networks which is brought forward according to the characteristics of neural network and stereoscopic vision. The relation between spatial points and image points is established by training the network without the parameters of the cameras, such as focus, distortions besides the geometry of the system. The training set for our neural network consists of a variety of stereo-pair images and corresponding 3D world coordinates. Then the 3D reconstruction of a new scene is simply using the trained network. Such a method is more similar to how human's eyes work. Simulations and real data are used to demonstrate and evaluate the procedure. We observe that the errors obtained from our experimentation are accurate enough for most machine-vision applications.

## 1 Introduction

In a stereoscopic vision system, the inputs to the computer are 2D-projections of 3D world objective things. The task of machine vision is to reconstruct 3D world according to such 2D pictures using the triangulation metrology and interpret the context of the scene. In this course, with a geometrical approach, it is unavoidable to determine the

---

\* This work is supported by the National Natural Science Foundation of China (NSFC No. 60405009 and 60374056) and Zhejiang Natural Science Foundation of China (ZJNSF No.Y104185).

relative position, the direction and internal and external parameters of the cameras. This procedure is termed as camera calibration.

Vision sensor calibration is an important issue of stereoscopic vision study, its purpose is to solve the position, parameters of the camera and build the model of imaging. Traditional camera calibration can be divided into two methods: one is linear model; another is non-linear model. Though linear model is simple, it is a standard basic model of the camera calibration, as it has not considered some linear or non-linear distortion in the course of imaging. So it cannot describe the geometry relation of imaging accurately, especially while using wide-angle lens, there is greater distortion in the place far away from the center of picture, making the precision of calibration greatly influenced. Though the non-linear model [1] takes care of the distortion effects, but it also leads to a more complicated mathematic equation at the same time. Furthermore, after the sensor is calibrated, the system still needs other mathematical approaches to formulate the 3D computation for obtaining the coordinates of a point in the scene. Sometime another nonlinear is necessary to perform such a computation.

In order to simplify the calibration and reconstruction procedures, this paper proposes a new method for three-dimensional vision calibration and reconstruction, which is based on feed-forward neural network, utilizing the artificial neural network through directly learning the information of the 2D pictures and information of the corresponding 3D coordinates, getting the relation between them, thus avoiding to build the complicated distortion model of cameras. Therefore, such a method works very similar to how human's eyes work and the stereoscopic sensor can work well even with a highly distorted vision sensor, such as camera pair with fish-eye lenses.

## 2 The General Imaging Model of a Camera

The common foundation of the optics triangulation is the theory of the geometrical optics imaging. The camera is used as the eye of a machine, through which realize the conversing from scenery to image. Fig.1 shows the imaging procedure from world coordinates to image coordinates (on pixels):

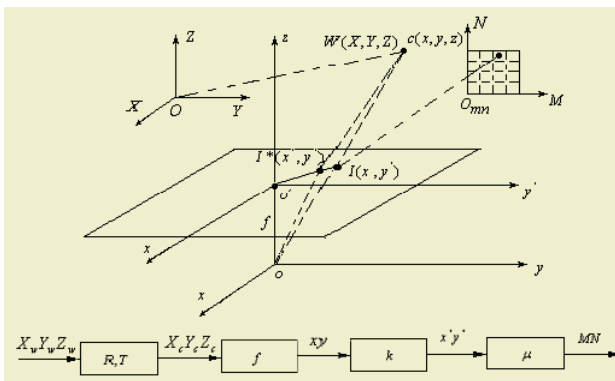


Fig. 1. The imaging model of a general camera

1) Transformation from world coordinates system to camera coordinates system:

$$\begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} = \begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} = M_1 \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \quad (1)$$

2) Transformation form camera coordinate system to no-distorted 2D-image coordinates:

$$Z_c \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} \quad (2)$$

3) The relation between the 2D-sensor coordinate system and image coordinate system:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{dx} & 0 & u_0 \\ 0 & \frac{1}{dy} & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (3)$$

The general projection form of an ideal camera:

$$\begin{aligned} Z_c \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} &= \begin{bmatrix} \frac{1}{dx} & 0 & u_0 \\ 0 & \frac{1}{dy} & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} \alpha_x & 0 & u_0 & 0 \\ 0 & \alpha_y & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} = M_1 M_2 \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} = M \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \end{aligned} \quad (4)$$

In the formula M is a 3-by-4 matrix, as we know, using more than 6 given points in space and their image pair-point coordinates, we can get projection matrix through least square method, accordingly get all internal and external parameters.

The non-linear model adds a few distortion parameters  $\delta_x(x, y), \delta_y(x, y)$  on the basis of linear model:

$$\begin{cases} x^* = x + \delta_x(x, y) \\ y^* = y + \delta_y(x, y) \end{cases} \quad (5)$$

It needs to use non-linear optimization to standardizing the camera while considering non-linear distortion, but it will also introduce too much non-linear parameter always and not only affect the precision, but also cause the instability of solution [2]. The method adopted in this paper need not to consider the non-linear distortion since the neural network algorithm has good nonlinear mapping ability. Through the learning of the network, in fact it has already taken the non-linear factor into account automatically. So it easily solved this non-linear optimization problem.

### 3 Neural Network Model

Back-Propagation (BP) Network is a kind of neural algorithm used widely for solving non-linear problems. The basic thought of the error back-propagation algorithm is to revise the connection coefficient from outputting floor according to the mean square errors between the Expected output and the network output of a sample, utilizing the method of gradient descent. The topological structure of neural network model that based on BP algorithm includes input layer, latent layer and output layer. BP network can learn a large number of mapping modes without any mathematic function describes the relation between the output mode and input mode.

According to transmission projection principle, the coordinates of corresponding points in two images are corresponded with the world coordinates of their spatial point. And there exits a nonlinear relation, which can be used to train the network. So we can establish such a mapping relation between the coordinate system of image and the coordinate system. From Fig.2,  $X, Y, Z$  presents the coordinate of a world point, corresponding the coordinate of two image  $(u_1, v_1)$  and  $(u_2, v_2)$ . In this neural network model,  $u_1, v_1, u_2, v_2$  is the input of the neural network, and  $X, Y, Z$  is the output of the neural network.

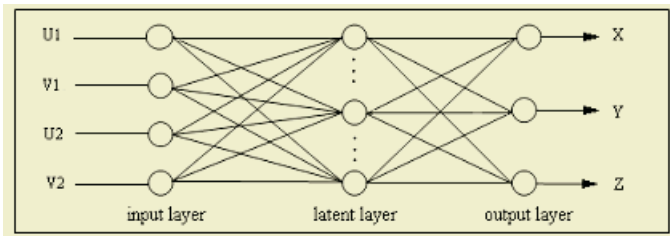


Fig. 2. ANN model for camera calibration

According to the above-mentioned neural network models, we adopted this feed-forward network with three layers. The common method of setting the number of node of latent layer is to try several numbers. Here we set up less likely latent nodes to train the network first, then increase the latent node gradually. Practically, it is determined as

$$m = \text{sqrt}(n + l) + a \tag{6}$$

This is used as an initial value of this trying method, among them  $m$  is counted for the nodes of latent floor,  $n$  is counted for the nodes of input floor,  $l$  is counted in order to export the node and  $a$  is a constant between  $1 \sim 10$ .

## 4 Experiments

### 4.1 The Calibration Data

In this paper, the calibration data was provided by Zhenyou Zhang from website [3]. The calibration object for reference is a level board with white background, on which distributes a number of  $8 \times 8$  black lattices. Each lattice has four coordinates. As a result there are 256 points known for test altogether. Fig.3 illustrated the pictures obtained from two cameras that located at different space location.

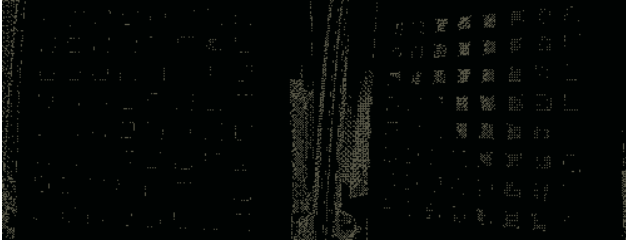


Fig. 3. Two images from two cameras (after [3])

### 4.2 Network Structure

In the experiments we finally selected the network structure of a single latent floor with 7 nodes after several times of trying. A sigmoid activation function is adopted for the latent layer and a linear activation function for both input and output layer. Then we improved the BP algorithm with the momentum law.

The method of momentum law is an improvement of ordinary BP algorithm, which is to improve the calculation of connection values. After putting into the momentum item, the calculation formula of a connection value is determined as:

$$W_i^{(l)}(k+1) = \alpha \delta_i^{(l)} y^{(l-1)} + \eta \Delta W_i^{(l)}(k-1) + W_i^{(l)}(k) \quad (7)$$

Here  $\alpha$  stands for study step,  $\eta$  for momentum factor. The adoption of the additional momentum item reduces the sensitiveness of network to details of curved error surface, suppressing the network to fall into some minimum effectively and contributing to reduce the training time at the same time. As a result, it improves the speed of learning and increases the dependability of the algorithm. The 'trainbpm.m' function offered in Matlab toolbox can be used to train the BP networks, which has less than four layers with additional momentum factors.



### 4.3 Network Training

The network is trained with 64 groups of sample data. The convergence tempo of the network mainly depends on the complexity of nonlinear mapping among the inputs and outputs. The more complicated the mapping relationship is, the longer time it is to train.

In order to make the BP network based on gradient drop rule to converge at the minimum point at a large, the learning rate should be small enough, but this will lengthen the time of converging greatly. And high learning rate will lead to fall into some local minimum point but difficult to converge. Through the method of turning step in the course of experiment, we decided the learning rate to be 0.02.

Fig.4 and Fig.5 show the learning errors in the course of training. From Fig.4 we can find the learning error in the first 1000 training decreased rapidly as the increase of the times of training.

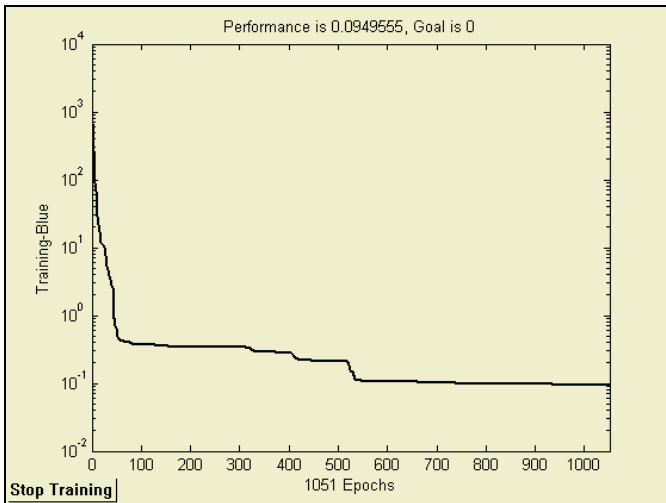
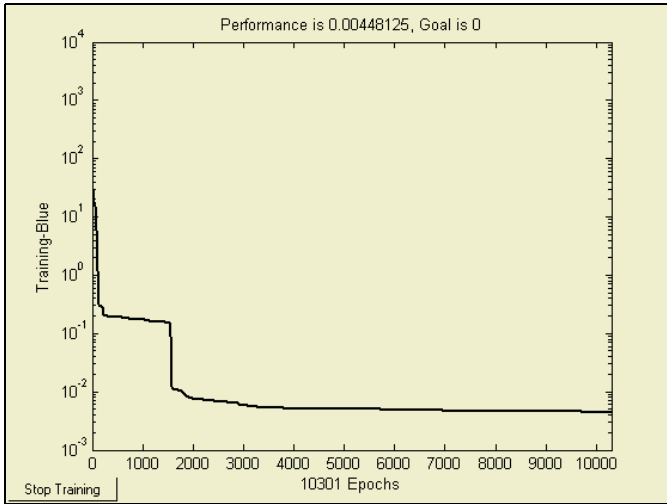


Fig. 4. Training after 1000 times

From Fig. 5 we find out that the magnitude of learning error decreased from  $10^{-1}$  to  $10^{-3}$  after 10000 times of training. So the simulation in which the neural network was trained has produced very satisfactory results. Increasing the training times to make the training error decreased to the magnitude of  $10^{-4}$  or more will not produce significant improvement of accuracy.

In this test, because the sample data are not very large, though we had not carried on the normalization to the sample data, the training result is still satisfactory. Of course, as increasing the size of the sample data, we need to carry on the normalization to the sample data since the limitation in the sensitive district of the *sigmoid* function. At that time, all sample data including inputs and outputs were normalized between 0 and 1 before training [5]. It is necessary to acquire a quick learning. The normalization can be performed by

$$x' = (x - x_{\min}) / (x_{\max} - x_{\min}) \quad x' \in [0,1].$$



**Fig. 5.** Training after 10000 times

**4.4 Network Testing**

In order to check the accuracy of the trained network, we test the network with eight stereo-pair points, which their corresponding spatial coordinates are known and were not included in the training set. Then we use the network to perform 3D-reconstruction. The mean square error between the Expected output and the network output is the way to evaluate this method [5]. Mathematically, it is written as:

$$RMS = \sqrt{(\Delta x^2 + \Delta y^2 + \Delta z^2)}/3$$

Here,  $\Delta x, \Delta y, \Delta z$  represent the differences between the corresponding coordinates of Expected output and network output.

**Table 1.** Simulation Result

(Mm)

| Expected output |         |       | Network output |         |          | <i>RMS</i> |
|-----------------|---------|-------|----------------|---------|----------|------------|
| $x_w$           | $y_w$   | $z_w$ | $x_w$          | $y_w$   | $z_w$    |            |
| 67.222          | -67.222 | 0     | 67.134         | -67.193 | 0.000178 | 0.053495   |
| 67.222          | -26.667 | 0     | 67.222         | -26.711 | -0.00039 | 0.025404   |
| 58.333          | -13.889 | 0     | 58.344         | -13.872 | -0.00078 | 0.011699   |
| 53.333          | -53.333 | 0     | 53.361         | -53.395 | 0.000415 | 0.039278   |
| 49.444          | -26.667 | 0     | 49.451         | -26.707 | 0.000229 | 0.023445   |
| 31.667          | -35.556 | 0     | 31.678         | -35.634 | 0.000676 | 0.045481   |
| 22.778          | -44.444 | 0     | 22.795         | -44.52  | -9.3E-05 | 0.044963   |
| 17.778          | -62.222 | 0     | 17.806         | -62.296 | -0.00175 | 0.045691   |

From Table.1, it is obvious that through the neural network of BP study algorithms, satisfactory results of three-dimension coordinates can be obtained directly as well as avoiding non-linear computations. The precision can be up to 0.1 mm, while the error of traditional calibration method is above 1 mm. Therefore, the proposed method based on improved BP neural network can carry on the three-dimensional vision camera calibration very well. The accuracy is higher than that of traditional methods and can meet the requirements of common industrial applications.

However, this 3D vision measuring method using neural network still exists drawbacks, the main one is that it needs many training sets compared with geometrical methods and the training process is relatively slow. In addition, the number of latent nodes needed to be fixed through times of trying.

## 5 Conclusion

This paper proposed a neural network approach perform both the camera calibration and 3D reconstruction with a stereoscopic vision sensor, which works very similarly to how human perceives the three-dimensional scene. The distinct advantage of the approach using neural network is that the method comprises the complicated nonlinear relation between 2D information on the images and three-dimensional information in the world, which can include various kinds of distortion and other nonlinear factors during the imaging period. Therefore, the method is very suitable for calibration and 3D reconstruction of a highly distorted stereoscopic vision sensor in which a traditional geometrical approach usually fails or is very expensive to implement. Furthermore, the inherent parallel running ability of the network will make the measuring speed of the network very fast and can meet the requirements of some real-time applications.

## Reference

1. Weng, J., Cohen, P., and Herniou, M.: Camera calibration with distortion models and accuracy evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (1992) 965- 980
2. Tsai, R.Y.: An efficient and accurate camera calibration technique for 3D machine vision. *International Conference on Computer Vision and Pattern Recognition*. (1986) 364-374
3. Zhang Z. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. Vol. 22 No. 11 (2000) 1330-1334
4. Qing, G. W. and Hrizinger G.: Multisensory Visual Servoing by a Neural Networks. *IEEE Transactions on Systems, Man and Cybernetics*. (1999)
5. Zhao, Q. J. Sun, Z. X. and Lan, L.: Camera calibration technology using neural network. *Control and Decision-making*. Vol.17 No.13. (2002)
6. Chen, S. Y. and Li, Y. F.: Automatic Sensor Placement for Model-Based Robot Vision. *IEEE Transactions on Systems, Man and Cybernetics, Part B*. Vol. 34, No. 1, (2004) 393-408
7. Li, Y. F. and Chen, S. Y.: Automatic Recalibration of an Active Structured Light Vision System. *IEEE Transactions on Robotics and Automation*. Vol. 19 No.2 (2003) 259-268
8. Weng J., Cohen P., and Herniou M.: Camera calibration with distortion models and accuracy evaluation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 14 No. 10 (1992) 965--980
9. Thormählen T., Broszio H., Wassermann I.: Robust Line-Based Calibration of Lens Distortion from a Single View. *Proc. of Mirage 2003, INRIA Rocquencourt, France*, (2003) 105-112

# Evolutionary Design of a Brain-Computer Interface

G. Romero<sup>1</sup>, M.G. Arenas<sup>2</sup>, P.A. Castillo<sup>1</sup>, and J.J. Merelo<sup>1</sup>

<sup>1</sup> Department of Architecture and Computer Technology,  
University of Granada, ETSII, Periodista Daniel Saucedo Aranda s/n,  
18071 Granada, Spain

<sup>2</sup> Department of Computer Science, University of Jaén, EPS,  
Avda. de Madrid, 35 Ed. 50, 23071 Jaén, Spain  
gustavo@geneura.ugr.es

**Abstract.** This paper shows how Evolutionary Algorithm (EA) robustness help to solve a difficult problem with a minimal expert knowledge about it. The problem consist in the design of a Brain-Computer Interface (BCI), which allows a person to communicate without using nerves and muscles. Input electroencephalographic (EEG) activity recorded from the scalp must be translated into outputs that control external devices. Our BCI is based in a Multilayer Perceptron (MLP) trained by an EA. This kind of training avoids the main problem of MLPs training algorithms: overfitting. Experimental results produce MLPs with a classification ability better than those in the literature.

## 1 Introduction

Brain-Computer Interface (BCI) is a novel method by which humans and computers can interact. Research in this area has grown rapidly in the last 15 years often addressed to people with severe motor disorders. BCI relies in detecting brain activity to communicate or control cursor movement or prosthesis.

There are a wide variety of signals that can be measured in a human brain: chemical, electrical, magnetic, mechanical, metabolic and thermal. Among them, the most frequently used is scalp electroencephalogram (EEG). EEG signals let us detect tiny changes in synaptic electrical currents that occur when different areas of the brain are used. These currents are in the order of microamps. With EEG it is impossible to gain precise information about the firing of a specific neuron but general patterns and rhythms of the activity of millions of them.

When a region of the brain is not active, it has a tendency to synchronize its firing patterns with its neighbors, producing discernible rhythms of various frequencies in the EEG. For instance, when the visual cortex is inactive, the area is dominated by the  $\alpha$  rhythm, which is a repeating pattern at around 8Hz. The motor cortex creates the  $\mu$  rhythm, at about 10Hz, when it is in rest. When a subject plans a movement, the neurons in this region must start firing independently to perform the movement and the  $\mu$  rhythm disappears. This is

known as the Event Related Desynchronisation, and is often what a BCI system tries to detect in the EEG.

Several methods have been used to classify EEG signals and guess user intentions. The most important are Genetic Algorithms, Hidden Markov Models [1], Neural Nets [2, 3], Linear Discriminant Analysis, Probabilistic Methods [4], Spatial Filtering [5] and Support Vector Machines [6]. Every method has its advantages and disadvantages. Linear methods require linearly separable data but tend to be faster, simpler, and more robust. If the data is not linear or there is a strong noise, nonlinear methods are more appropriate, even though they are computationally more demanding.

For this work we choose a problem of self-regulation of slow cortical potentials (SCPs), of the **Ia** data set from **the BCI Competition 2003** ([7, 8]) as a test bed. The goal of this competition was to validate signal processing and classification methods for BCI. Thanks to the competition we can compare our method and results with several other people's work.

More exhaustive states of the art with detailed physiological information can be found in [9, 8]. Next sections describes the data set Ia, the experimental results and gives a conclusion and some clues on future work.

## 2 Self-regulation of Slow Cortical Potentials: Data Set Ia

This data set was provided by Niels Birbaumer from the Institute of Medical Psychology and Behavioral Neurobiology, University of Tübingen [10].

Data set Ia was taken from a healthy subject. The subjects were asked to move a cursor up or down on a computer screen, while their slow cortical potentials (SCPs) were recorded. The subjects received visual feedback of their SCPs (Cz-Mastoids) which were corrected for vertical eye movements. Cortical positivity (negativity) led to a downward (upward) movement of the cursor on the screen. Each trial lasted 6 seconds.

During each trial, the task to produce cortical negativity or positivity was visually presented by a highlighted goal at either the top or bottom of the screen from 0.5 s on. The visual feedback was presented from second 2 to second 5.5. For the competition, only this interval of every trial was provided for training and testing in order to avoid the classification of brain responses related to task presentation or reinforcement. Brain activity was recorded from the following 6 scalp positions at a sampling rate of 256 Hz. The sampling rate of 256Hz and the recording length of 3.5s results in 896 samples per channel for every trial. As 6 channels were recorded, the total length of every trial is 5376.

The training set were 268 trials recorded on two different days and mixed randomly. Of the total 268 trials, 168 originated from day 1 and the remaining 100 trials from day 2. Of these trials, 135 belong to class 0 and 133 to class 1. The test set contains 293 trials recorded on the second day.

For the BCI Competition 2003 ([7, 8]) every participant had to submit their estimated class ratings for every trial of the test set. The performance measure

was the correct response rate defined by the number of correctly classified trials divided by the total number of trials.

The competition winner was Brett Mensh from the MIT with an error rate of 11.3% using a linear discriminant analysis on the DC potentials and features from the spectral analysis of high  $\beta$  power band [11]. Error rates below 12% were also achieved by Gido Dornhege (Fraunhofer FIRST, IDA, Berlin) using regularized linear discriminant classifiers and Kai-Min Chung (National Taiwan University, Taipei) who applied a support vector machine classification on the data after downsampling to 25Hz. The best result using a MLP was 19.8% from Justin Sanchez (University of Florida). For a more detailed outcome of the competition see [7].

### 3 Experimental Results

#### 3.1 Algorithm Description

Our algorithm uses MLPs trained by an Evolutionary Algorithm (EA). The MLPs are fully connected and with any number of hidden layers. Optimum results, due to time constraints, were obtained with just one layer of one neuron, a simple Perceptron. Every MLP were represented by a floating point vector to let us apply some algorithms like blend crossover (BLX) [12] more easily. The EA is real coded and was implemented with EO [13].

The objective is to train a MLP to classify the training set, a matrix of 268 vectors of length 5376, into two classes. The test set is used after the training process to verify the quality of the MLP obtained.

Without much knowledge about the brain behaviour, our work focuses on extracting the maximum information possible from the training set.

Three genetic operators were used whose rates of application were experimentally determined by means of the run of a set of experiments. These operators were two kinds of mutation and blend crossover [12].

The first mutation operator was additive following a Gaussian distribution of center 0 and deviation 1. The second mutation operator was multiplicative

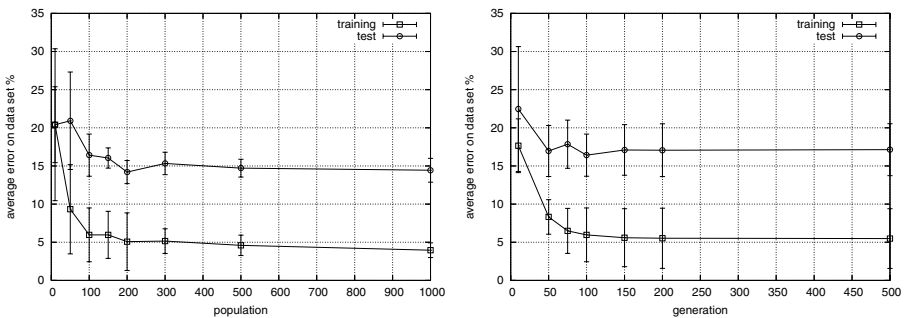


Fig. 1. EA parameters: population (left) and generation (right)

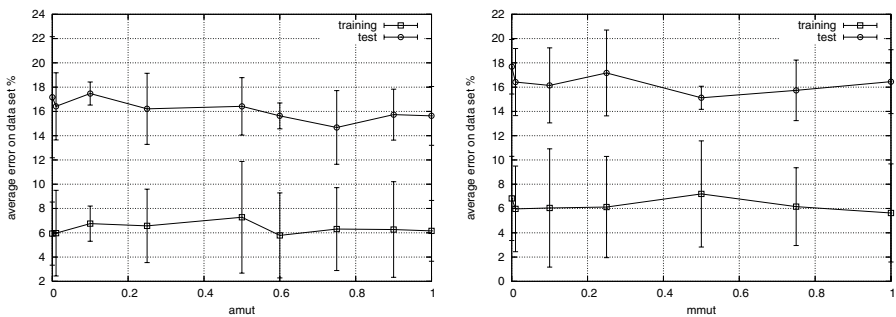
with a uniform distribution over the interval  $[-2, 2]$ . Both of them were applied to only one gene of an individual in every generation if it was due. The blend crossover or  $BLX-\alpha$  [12] is two parent based and its parameter values will be studied the section 3.2.

Other characteristics of our method are: As no local search method is used, our method is more resistant to the overfitting problem. No cross-validation method is needed because all the available samples can be used to validate the MLPs. For this reason the results of our algorithm doesn't depend on the partitions selected. The number of parameters of the algorithm is lower than similar methods as no local search algorithm is used to accelerate the training.

### 3.2 Parameter Selection

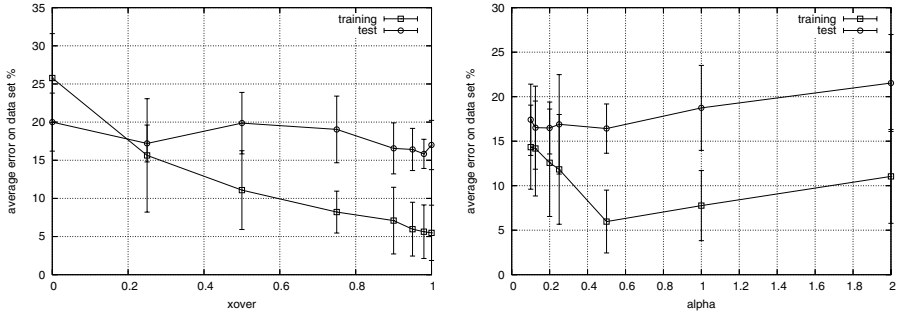
Our first task was to adjust the parameter values through the run of a series of experiments. For some parameters optimum values are easy to guess. Population sizes bigger than 200 individuals are very resource consuming and gives little additional improvements in solution quality (see figure 1).

For some others, as the number of generations, common sense is not always the best way to fix it. In most of the memetic methods, as [14], the classification error on the training set decreases with time, but on the test set this is only true for a limited period of time. After some number of generations the overfitting problem appears and the MLPs start to lose it's generalization ability. This is due to the use of some kind of cross-validation and back-propagation learning method (or any of it's more advanced versions as Quickprop or R-Prop). In figure 1 right, it can be seen that our algorithm doesn't suffer from this problem as generalization keeps constant once reached the maximum.



**Fig. 2.** EA parameters: **amut** (left): Additive Gaussian mutation of center 0 and standard deviation 1, **mmut** (right): Multiplicative uniform mutation in  $[-2, 2]$

The genetic operators application rate was calculated with two methods. The first is running three series of experiments. Application rates of additive mutation (**amut**, figure 2 left) and multiplicative mutation (**mmut**, figure 2 right) don't follow a clear trend so using low rates will help performance. Classification error seems to improve with higher crossover rates (**xover**, figure 3 left).



**Fig. 3.** EA parameters: **xover** (left): BLX- $\alpha$  crossover application rate, **alpha** (right):  $\alpha$  value for the BLX- $\alpha$  crossover

The second method is the use of a dynamic breeder able of self adapting the application rates based on the success of the new offspring generated. This method offers the best results, but with the added cost of more individual evaluations. After every application of an operator a individual has to be evaluated to adjust the statistics for that operator. The operator rates may change a 5% maximum per generation. The best values found by the dinamic breeder were  $\text{amut} = 0.34$ ,  $\text{mmut} = 0.5$  and  $\text{xover} = 0.98$ . Until this second method is fully tested it will be trusted only as a clue to find the optimal parameter values.

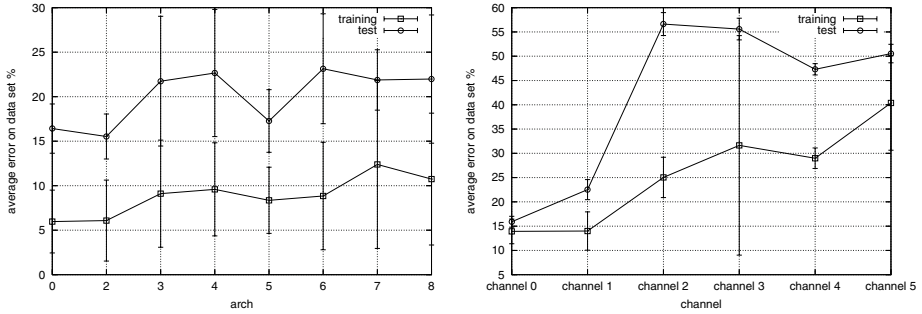
The rest of the parameters were determined by the run of series of experiments. Some were interesting as **alpha**, the  $\alpha$  value for the BLX operator. As can be seen in figure 3 right, the parameter has a great impact in the quality of solutions taking into account the training set, but over the test set, the difference is much lower. This lead us to think that the effect of this operator is similar to that of local search methods in memetic algorithms (in a more evident way than mutation operators).

Finally, the architecture of the net was selected with G-Prop [14]. The best results were obtained with a simple Perceptron of one neuron and with a MLP with 2 hidden neurons and one output neuron. For this parameter, *arch*, the results were very similar for both the training and test sets, see figure 4 left. More hidden layers add complexity to the net with no additional benefits in quality.

### 3.3 Preprocessing the Data Sets

From the 6 channels of the original data set, 0 and 1 are the more relevant, specially channel 0. To demonstrate this we try to classify the data sets just from one of the channels every time. The results are in figure 4 right. Channels 0 and 1 are equally effective to learn the training set. For the Test set, channel 0 is better. Using just one channel the training process can be speed up losing a 4.85% in classification quality on the training and gaining a 7.55% on the test set.





**Fig. 4.** EA parameters: **arch** (left): number of neurons of the hidden layer, **channel** (right): ICM data channels A1, A2, C3f, C3p, C4f and C4p

**Table 1.** Classification error % on test set

| Author     | Method | Training Set         |      | Test Set             |      |
|------------|--------|----------------------|------|----------------------|------|
|            |        | $\bar{x} \pm \sigma$ | best | $\bar{x} \pm \sigma$ | best |
| B.D. Mensh | SPC    |                      | 18.7 |                      | 11.3 |
| J. Sanchez | MLP    |                      |      |                      | 19.8 |
| G. Romero  | MLP+EA | 7.20±2.06            | 5.97 | 11.77±1.47           | 8.53 |

A preprocessing of the training and test sets also help to improve the quality of the solutions found. The best preprocessing found is a direct Fourier transform known as the discrete cosine transform ( $Y_k = X_0 + (-1)^k X_{n-1} + 2 \sum_{j=1}^{n-2} X_j \cos \frac{\pi j k}{n-1}$ ). The use of this preprocessing make the training set absolute classification accuracy go up a 3.10%. The improvement in the test set is a 7.04%.

### 3.4 Results

Experimental results are compared against the best from the BCI Competition 2003 [7, 8]. Results from the competition are the best classification error obtained. We also show the mean and standard deviation for a series of 100 experiments using the optimal parameter discovered and changing the random seed. Average results are similar of the best found in the literature and even better MLPs are found with our method.

## 4 Conclusions and Future Work

MLPs and simple Perceptrons are capable of classifying EEG signals very accurately. Not all the inputs are equally relevant. The channel 0 is more appropriate and lead to better training times and better classification capacity. Also the use

of some kind of preprocessing on the data can improve the results, as we have verified through the use of a discrete cosine transform.

The method presented on this paper avoids overfitting because no local search algorithm is used. As a consequence, a lower number of parameter is needed to use it and also the need for a division of the data set into training set and validation set, or any other kind of cross-validation, is avoided.

As a multiobjective problem, the size of the MLPs, the speed of training and the quality of the classification are in conflict. An implementation using fitness sharing [15] and with the help from a visual analysis of the data [16] can lead to better results in all or part of these subproblems.

## References

1. N. P. Hughes, S. J. Roberts, and L. Tarassenko. Markov Models for Automated ECG Interval Analysis. In *Advances in Neural Information Processing Systems 16 (NIPS 2003)*, 2003. in press.
2. A. Erfanian and M. Gerivany. EEG signals can be used to detect the voluntary hand movements by using an enhanced resource-allocating neural network. In *Conference of the IEEE Engineering in Medicine and Biology Society*, volume 23, Istanbul, Turkey, 2001.
3. S. Roberts P. Sykacek and M. Stokes. Adaptive bci based on variational bayes: an empirical evaluation. In *Brain-Computer Interface Workshop*, Albany, June 2002.
4. M. Stokes E. Curran M. Gibbs P. Sykacek, S. Roberts and L. C. Pickup. Probabilistic methods in bci research. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 11(2):192–195, 2003.
5. H. Ramoser, J. Müller-Gerking, and G. Pfurtscheller. Optimal spatial filtering of single trial EEG during imagined hand movement. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 8:441–446, December 2000.
6. Benjamin Blankertz, Gabriel Curio, , and Klaus-Robert Müller. Classifying single trial EEG: Towards brain computer interfacing. In S. Becker T. G. Diettrich and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems (NIPS 01)*, volume 14. MIT Press, 2002. to appear.
7. The BCI Competition 2003. <http://ida.first.fraunhofer.de/projects/bci/-competition>, 2003.
8. Benjamin Blankertz, Klaus-Robert Mller, Gabriel Curio, Theresa M. Vaughan, Gerwin Schalk, Jonathan R. Wolpaw, Alois Schlgl, Christa Neuper, Gert Pfurtscheller, Thilo Hinterberger, Michael Schrder, and Niels Birbaumer. The bci competition 2003: Progress and perspectives in detection and discrimination of eeg single trials. *IEEE Transactions on Biomedical Engineering*, 2004. accepted.
9. T.M. Vaughan et al. Brain-computer interface technology: A review of the second international meeting. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 11(2):94–109, June 2003.
10. T. Hinterberger I. Iversen B. Kotchoubey A. Kbler J. Perelmouter E. Taub H. Flor N. Birbaumer, N. Ghanayim. A spelling device for the paralysed. *Nature*, (398):297–298, 1999.
11. Brett D. Mensh, Justin Werfel, and H. Sebastian Seung. BCI Competition 2003 - Data Set Ia: Combining Gamma-Band Power with Slow cortical Potentials to Improve Sigle-Trial Classification of Electroencephalographic Signals. *IEEE Transactions on Biomedical Engineering*, manuscript TBME 00259-2003. R1, in press.

12. Larry J. Eshelman and J. David Schaffer. Real-coded genetic algorithms and interval schemata. In L. Darrell Whitley, editor, *Foundations of Genetic Algorithms*, volume 2, pages 187–202, Los Altos, CA, 1993. Morgan Kaufmann.
13. M. Keijzer, Juan-Julián Merelo-Guervós, G. Romero, and M. Schoenauer. Evolving Objects: a general purpose evolutionary computation library. In Pierre Collet, Cyril Fonlupt, Jin-Kao Hao, Evelyne Lutton, and Marc Schoenauer, editors, *Artificial Evolution, 5th International Conference, Evolution Artificielle, EA 2001, Le Creusot, France, October 29-31, 2001, Selected Papers*, pages 231–244, December 2003.
14. P.A. Castillo; Juan-Julián Merelo-Guervós; A. Prieto; I. Rojas and G. Romero. Statistical analysis of the parameters of a neuro-genetic algorithm. *IEEE Transactions on Neural Networks*, 13(6), November 2002.
15. D.E. Goldberg and J. Richardson. Genetics Algorithm with Sharing for Multimodal Function Optimization. In J.J. Grefenstette, editor, *Proceedings of the First International Conference on Genetic Algorithms*, pages 41–49, Hillsdale, NJ, 1987. Lawrence Erlbaum Associates.
16. G. Romero; M. García-Arenas; J. G. Castellano; P. A. Castillo; J. Carpio; Juan-Julián Merelo-Guervós; A. Prieto; V. Rivas. Evolutionary computation visualization: Application to G-PROP. Number 1917 in *Lecture Notes in Computer Science*, pages 902–912. Springer, 2000.

# Vision-Based Walking Parameter Estimation for Biped Locomotion Imitation

Juan Pedro Bandera Rubio<sup>1</sup>, Changjiu Zhou<sup>2</sup>,  
and Francisco Sandoval Hernández<sup>1</sup>

<sup>1</sup> Dpto. Tecnología Electrónica, E.T.S.I. Telecomunicación,  
Universidad de Málaga, Campus de Teatinos, 29071-Málaga, Spain

<sup>2</sup> School of Electrical and Electronic Engineering,  
Singapore Polytechnic, 500 Dover Road, Singapore 139651

**Abstract.** This paper proposes a new vision-based system that can extract walking parameters from human demonstration. The system uses only a non-calibrated USB webcam connected to a standard PC, and the human is only required to put three color patches on one of his legs and walk roughly in a perpendicular plane with respect to camera orientation. The walking parameters are then extracted in real time, using a local tracking system to follow the markers and a fast decision layer to detect the main features of the leg movement. As only one leg can be tracked properly using only one camera, we assume symmetric movement for left and right legs. Once extracted, the parameters have been successfully tested by generating walking sequences for both simulated and real Robo-Erectus humanoid robots.

## 1 Introduction

Biped Locomotion has become a key topic in robotic research over the last decade. Both hardware improvements and new software architectures have allowed the implementation of impressive humanoid robots such as Honda ASIMO, Sony QRIO or Fujitsu HOAP-2. Many universities have begun research in humanoid robotics, and some very interesting prototypes have appeared over the last years. Most of them use one of the two main approaches to biped locomotion: *static* or *dynamic*. Both approaches will be reviewed in Section 2. The main objective, in any case, is to produce a gait as natural and stable as possible.

One of the most popular and efficient methods to generate these natural gaits is imitation. As humanoid robots are inspired by the properties of a biological system -humans- it is not far-fetched to consider biologically inspired computation methods for them. The idea is that the movements of a humanoid robot will be more natural when produced through imitation of a human teacher. This process of imitation can be simplified by extracting suitable characteristics of the human motion [1].

If we focus in the walking process, a common assumption is to consider it as a periodic movement. The postures at the beginning and the end of each step have

to be identical to have a continuous and repeatable gait [3]. This requires the selection of specific initial conditions, constraint functions and their associated gait parameters. However, this gait will not usually be human-like. A common solution to this issue is the use of Human Motion Capture Data (HMCD) to drive the robot [5]. Nevertheless, some researchers show that the HMCD cannot be applied directly to humanoid robots due to kinematic and dynamic inconsistencies between the human subject and the humanoid. Kinematic corrections are usually required while calculating the joint angle trajectory [2].

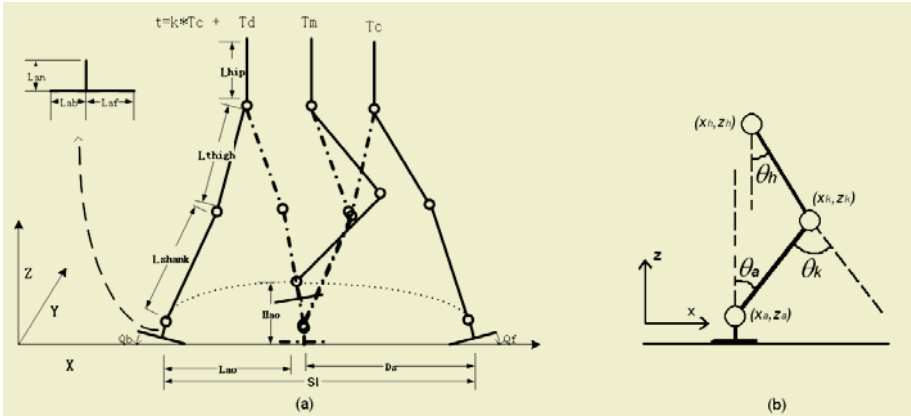
Adaptation of HMCD for a humanoid robot is commonly made with periodic joint motion corrections at selected joints to approximately match the desired Zero Moment Point (ZMP) trajectory. Constrained optimization methods can be used to maximize the dynamic stability against sliding during walking. In any case, it is very complex to adapt the trajectories from human to humanoid. The main problems arise from differences between human and humanoid bodies, stability, and noise during the perception stage. In order to get as good reference trajectories as possible, commercial HMCD systems can be used. However, most of them are very expensive and intrusive. They usually require special suits or markers, fine calibration and specific environments.

This paper introduces a new approach to the adaptation problem. The proposed method relies on parameter extraction to facilitate the gait conversion from human to humanoid. This means that perceived trajectories will not be used directly to create synthetic gait. Instead of this, the system analyzes the perceived motion and extracts the main walking parameters from it. These parameters are then used to create motion for the humanoid. If the set of extracted parameters is sufficient to characterize the gait, the generated movement will imitate correctly the perceived one [3][6]. The proposed method does not require a complex perception system to extract the base parameters. The HMCD used for our system was collected in the ARICC Lab at Singapore Polytechnic, using a non-calibrated USB webcam. The human performer only need to wear three color patches roughly placed on hip, knee and ankle.

## 2 Biped Locomotion and Key Walking Parameters

Biped motion is generally divided into a *Single Support Phase* (SSP), when only one foot is on the ground, and a *Double Support Phase* (DSP), when both feet are on the ground. In ordinary human gait, the DSP lasts for approximately 20% of the step cycle.

There are two types of biped gait: static and dynamic. Static walkers rely on the static equilibrium condition: maintain the Center of Gravity (CG) on the convex hull of the contact area with the ground. This approach denies inertial forces and therefore can be applied only if robot movements are very slow. Dynamic walkers achieve fast and natural walking motion following the principle of dynamic equilibrium: they use ZMP instead of CG, so that inertia components and gravity are considered. This is the approach adopted in this paper.



**Fig. 1.** a) Walking parameters and coordinates, b) Joint positions and angles

The process of gait generation can be simplified if we just generate a synthetic gait from key walking parameters. These parameters can be seen in Fig. 1.a and are deeply explained in [3]. There are four time-related parameters, related to step period, *SSP* and *DSP* intervals, and time of maximum ankle height. The detection of these key moments in the walking process will be one of the main objectives of the perception stage. The rest of parameters are related to lengths and distances.

### 3 System Overview

Our system uses a tracking stage to estimate joint positions (Fig. 1.b) in real-time. The outputs of this stage are the centroids of the detected markers. This information is used as input for a decision algorithm that determines the walking phase, and detects transitions between phases. The moments in which these transitions occur, and the joint positions at these moments, are used to extract the walking parameters. Finally, these parameters are tested by generating simulated and real robot walking sequences and comparing them with the perceived ones.

### 4 Perception System

Our perception system relies on a fast and efficient color-based tracking algorithm to extract the movement of a human leg. We instrument the human with three color patches located in the hip, knee and ankle of one of his legs. The tracked human is also supposed to walk more or less in a perpendicular straight line with respect to camera orientation.

The perception system is divided into two stages. The first stage tracks the color markers, and reduces drastically the amount of information the system is

using, from a video sequence to just the  $(x, z)$  positions of the three markers in each frame. The second stage uses these positions to determine the phase of the movement.

### 4.1 Tracking Algorithm

The algorithm we are using to track the color markers was proposed in [4], and it relies on a hierarchical representation of the regions of interest to track these regions in real time. The basis of the algorithm is to use a weighted template to follow each object. This template changes not only its position, but also its shape so that it can successfully handle partial occlusions, and perspective and physical deformations. The method represents both target and template using a Bounded Irregular Pyramid (BIP). This structure is a hierarchical approach to tracking that reduces very much the computational cost and allows the system to track the three leg markers in real time.

The only difference between the original tracking algorithm and our implementation is that we have used Hue-Saturation-Intensity (HSI) color space, more suitable to track saturated colors. See [4] for further details.

### 4.2 Decision Algorithm

The positions of hip, knee and ankle are used to extract walking parameters from the HMCD. As these parameters are related to walking phases, the proposed system must be able to detect these phases and, specially, the transitions between them.

Each step can be roughly divided into three stages: the first is the *Double Support Phase* (DSP). The second and third phases are contained in the *Single Support Phase* (SSP).

The decision stage will determine when the leg is changing from one phase, or state, to another (Fig. 2). More concretely, the decision will be related to ankle movement. A previous average filter is applied before decision to reduce noise. Alternative approaches were tried such as computing joint angle trajectory minima, and other measurements on relative joint distances. However these have appeared to be more sensitive to noise, as the inertia of markers and noise affects more the distance and angle information.

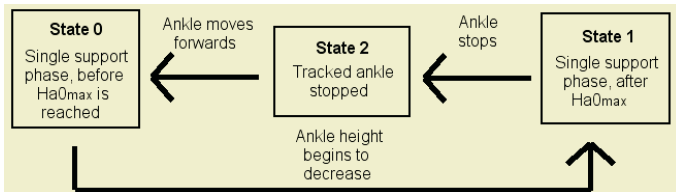


Fig. 2. Decision algorithm states

## 5 Extraction of Walking Parameters

Our approach to walking process considers it as a symmetric, periodic and smooth motion. The hip, knee and ankle trajectories, both in Cartesian and Joint spaces, can be considered as smooth curves characterized by a set of key points [3]. These key points should be identified in the transition moments where the state of the leg is changing. So changes detected in the decision stage become the key moments in which walking parameters can be extracted.

These parameters, already introduced in Sect. 2, can be divided into two main groups: *time* parameters and *position* parameters.

### 5.1 Time Parameters

Time parameters are directly obtained from the decision stage. We consider that each step begins with a *DSP* followed by a *SSP*. Being  $T(kab)$  the time instant, in step  $k$ , in which the leg moves from state  $a$  to state  $b$ , time parameters are obtained as detailed below:

- $T_s$ . The *SSP* period is obtained using this equation:

$$T_s = T(k12) - T(k20) \tag{1}$$

- $T_d$ . The system is tracking only one leg. This means that there is no direct way to obtain the *DSP* period as it is defined as the time between the *SSPs* of both legs. If we assume a symmetric walking pattern, then  $T_d$  can be obtained from the moment in which the tracked ankle stops:

$$T_d = (T(k20) - T((k - 1)12) - T_s)/2 \tag{2}$$

- $T_m$ . This is the moment in which the ankle reaches its maximum height, measured from the beginning of the previous *DSP*. This is easily obtained from the transition between states 0 and 1:

$$T_m = T(k01) - T(k20) + T_d \tag{3}$$

- $T_c$ . This is the step period,  $T_c = T_d + T_s$ .

### 5.2 Position Parameters

These parameters (Fig. 1.a) are obtained in the same way as time parameters, as the main information about joints motion is also related to transition moments:

- $S_l$ . The step length is the horizontal motion of the tracked ankle during states 0 and 1:

$$S_l = |x_a(k12) - x_a(k20)| \tag{4}$$

where  $x_a(kab)$  denotes the horizontal ( $x$ ) value for the ankle position in step  $k$ , in the transition from state  $a$  to state  $b$ .



- $Ha0$  and  $La0$ . Both parameters are related to transition between states 0 and 1:

$$Ha0 = |z_a(k01) - z_a(k20)|; La0 = |x_a(k01) - x_a(k20)| \quad (5)$$

- $Xsd$  and  $Xed$ . These parameters are distances between hip and ankle in key instants. The system extracts these parameters at the beginning and the end of the *SSP*:

$$Xsd = |x_a(k20) - x_h(k20)|; Xed = |x_a(k12) - x_h(k12)| \quad (6)$$

- $Hm_{max}$  and  $Hm_{min}$ . The extraction of these parameters is slightly different, as they are not related to transition moments [3].  $Hm_{max}$  and  $Hm_{min}$  are only required to produce a more natural motion in the simulator and real robot. The perception stage will obtain them as the maximum and minimum vertical hip positions during the human demonstration.

## 6 Experiments and Results

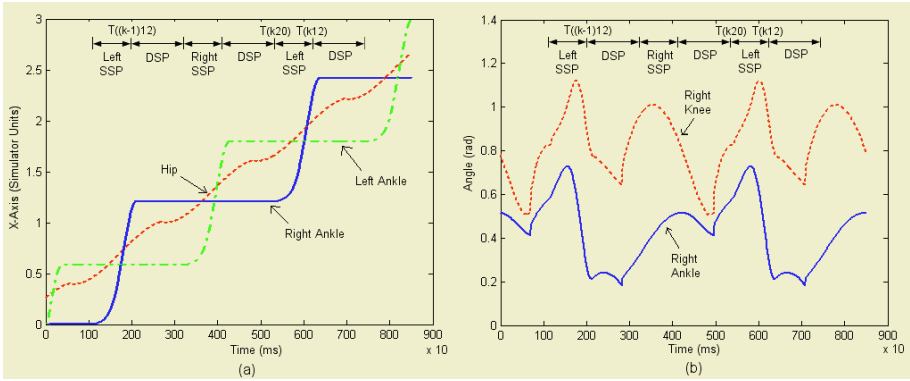
The proposed system has been tested in an indoor environment, using pink color patches as markers. The algorithm runs on a standard PC and the images are captured using a Ranger VCAM365 commercial webcam. Our approach does not require the camera to be calibrated. Besides, no specific illumination conditions nor specific background have been used.

The HMCD has been used to generate walking sequences over a Simulator developed at ARICC [6]. These sequences have been also successfully adapted to a real Robo-Erectus robot [7], also developed and built at ARICC, in Singapore Polytechnic. Table 1 shows the walking parameters extracted from a perceived walking motion. Time parameters are in seconds, position parameters are in pixels. After perception, all position parameters are normalized with respect to perceived shank length. When these parameters are translated to the robot simulator or the real robot, they are multiplied by the value of shank length that simulator or robot are using.

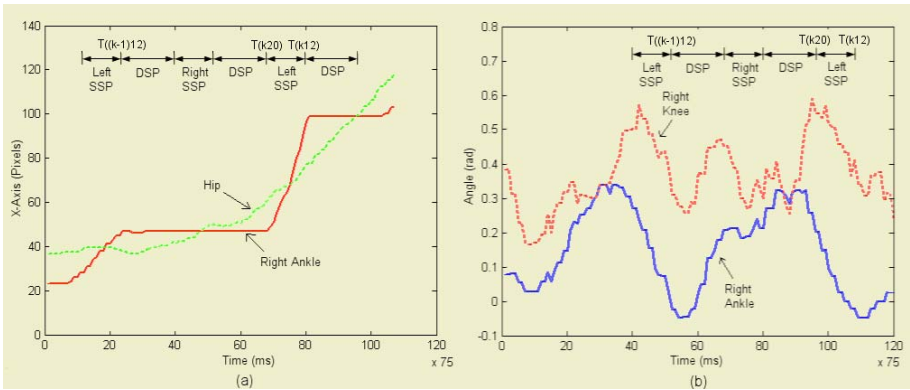
The parameters presented in Table 1 were used to generate a simulated walking sequence. The horizontal motion of hip and ankles in the simulator is shown

**Table 1.** Walking parameters extracted from human motion

| Parameter    | Value  | Normalized Value | Parameter  | Value | Normalized Value |
|--------------|--------|------------------|------------|-------|------------------|
| $T_c$        | 2.1325 | 2.1325           | $Ha0$      | 3     | 0.069767         |
| $T_s$        | 0.9690 | 0.9690           | $La0$      | 12    | 0.279070         |
| $T_d$        | 1.1635 | 1.1635           | $Xsd$      | 12    | 0.279070         |
| $T_m$        | 1.6165 | 1.6165           | $Xed$      | 8     | 0.186047         |
| <i>Shank</i> | 43     | 1                | $Hm_{max}$ | 71    | 1.651163         |
| $S_l$        | 26     | 0.604651         | $Hm_{min}$ | 66    | 1.534884         |



**Fig. 3.** Simulated walking cycle: a) Horizontal displacements of the hip and both ankles, and b) Joint angle trajectories



**Fig. 4.** Perceived walking cycle: a) Horizontal displacements of the hip and right ankle, b) Joint angle trajectories

in Fig. 3.a. By extracting only the relevant parameters and reconstructing the trajectories the system is able to synthesize a low noise smooth and stable motion. On the other hand, Fig. 3.b presents the trajectories in joint space for the previous movement. We can see here how it would be much more difficult to extract the walking parameters in this space, because angle trajectories are not as clearly associated with walking parameters as joint displacements.

Fig. 4 shows the perceived walking cycle. Although there are evident similarities between real and generated cycles (so that real and artificial movements will be very similar), the perceived walking sequence is much more noisy, and non periodical. Besides, we must also consider the possibility of losing markers. All this justifies the extraction of parameters as opposed to direct trajectory imitation.

## 7 Conclusions and Future Work

This paper has presented a novel approach to extraction of walking parameters from a perceived video sequence. The chosen parameterization allows to apply the extracted parameters on the simulator or real robot. The results show that generated sequences are rather similar to real ones. The proposed method requires only a standard, uncalibrated digital camera and three color patches.

Future work will address *on-line* balance compensation and imitation. We will also collect human walking data under various behavioral conditions. Parameter estimation techniques will be tested to extract more effectively human walking parameters for humanoid control and imitation. We will also conduct research on the use of periodic joint motion corrections at selected joints to dynamically match the desired ZMP trajectory.

## Acknowledgements

This work has been partially supported by the Spanish Ministerio de Educación y Ciencia (MEC), project number TIN2004-05961.

## References

1. Aggarwal, J.K., Slotin, J.: Human motion analysis: a review. *Computer Vision and Image Understanding*, **73**, **3**, (1999) 295-304
2. Dasgupta, A., Nakamura, Y.: Making feasible walking motion of humanoid robots from human motion capture data. In *Proc. IEEE Intl. Conf. on Robotics & Automation*, Detroit, Michigan, (1999) 1044-1049
3. Huang, Q., Yokoi, K., Kajita, S., Kaneko, K., Arai, H. Koyachi, N., Tanie, K.: Planning walking patterns for a biped robot. *IEEE Trans. Robot. Automat.*, **17**, (2001) 280-289
4. Marfil, R., Rodriguez, J.A., Bandera, A., Sandoval, F.: Bounded irregular pyramid: a new structure for color image segmentation. *Pattern Recognition*, **37**, **3**, (2004) 623-626
5. Safonova, A., Pollard, N., Hodgins, J. K.: Optimizing Human Motion for the Control of a Humanoid Robot. In *Proc. 2nd International Symposium on Adaptive Motion of Animals and Machines (AMAM2003)*, Kyoto, Japan, (2003)
6. Tang, Z., Zhou, C., Sun, Z.: Gait planning for soccer playing humanoid robots. *Lecture Notes in Control and Information Sciences*, Springer-Verlag, **299**, (2004) 241-262
7. Zhou, C., Yue, P.K.: Robo-Erectus: a low cost autonomous humanoid soccer robot. *Advanced Robotics*, **18**, **7**, (2004) 717-720

# Designing a Control System for an Autonomous Robot Using an Evolutionary Algorithm

P.A. Castillo, G. Romero, M.G. Arenas, J.J. Merelo,  
and A. Prieto

Department of Architecture and Computer Technology,  
University of Granada, Spain  
`pedro@atc.ugr.es`

**Abstract.** Hand-design of control systems for autonomous robots that act in dynamic or noisy environments is a complex task.

In this paper, a new technique for controller design, termed decision-vector, is presented. An evolutionary approach is proposed: the control systems (candidate solutions) are made up of the set of robot states with respect to the obstacles it can detect, and the corresponding actions to take on each one of those situations.

This initial work carries out the evolution of controllers in two environments, so that it is clear that, in spite of the simplicity of the proposed model, it is powerful enough to guide the robot to reach a target avoiding obstacles, and even, tracking a spread mark on the ground.

## 1 Introduction

Instead of hand-design a system to make a specific task, an evolutionary approach can set up a system whose behavior is specified, direct or implicitly, in the evaluation function (fitness function). It is convenient to use an evolutionary algorithm (AE) [3] as a method to search the space of the possible adaptations of the control system to the environment, not as a method to search for all the possible control systems.

In evolutionary computation, the members of a population must be evaluated throughout many generations. In the case of the evolution of control systems for robotics, several authors propose to evaluate robots in the real world [1, 5, 6]. In this paper, we propose to use a simulator to evaluate controllers. Then, once the control system has been tuned, it could be evaluated controlling a real robot. In any case, simulations must be made taking in account that the robot will act in a noisy environment and the sensors may introduce noise.

Designing a controller for an autonomous robot must be done using no details about the environment it is moving in, i.e., it must directly interact and explore the environment by means of its sensors.

In order to set up these kind of controllers, many authors have used fuzzy logic [7]. Nevertheless, in most cases, the design is made by hand or using architectures not suitable as the system complexity grows [8]. Other authors use genetic

programming to set up controllers that solve avoiding obstacles tasks [6]. Finally, some authors propose to evolve artificial neuronal networks (in their different configurations) for robot control [2]. The design of these control systems must be done automatically. However most of commented methods need the intervention of an expert, or some knowledge about the problem or the environment.

This paper proposes to automatically design a control system to guide a robot. An EA is used to establish the parameters of a simple architecture of control to successfully guide the robot through different environments.

The evaluation of an individual of the population (controller) consist of guiding the robot for a period of time using a simulator (developed for this purpose).

The rest of this paper is structured as follows: Section 2 presents the model of the robot and the control system. Section 3 describes two experiments (simulation environments). Simulations are intended to faithfully model physical characteristics of the robot and its interactions with the environment. Finally, section 4 shows obtained results, followed by a brief conclusion (sección 5).

## 2 Robot Model

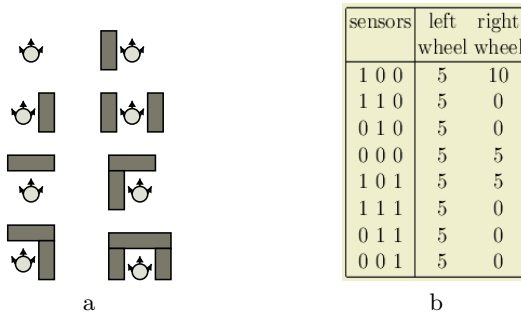
The navigation of an autonomous robot requires using several sensors to interact with its environment. End of race mechanical sensors ("bumpers") or ultrasounds sensors are commonly used. That way, the robot can measure the distance to the closest objects in its operational range.

Proposed model of the robot uses three sensors, two actuators (to move the wheels) and a control system. Ultrasounds sensors are used to measure the distance to the objects in the direction the sensor is oriented. 2D ray-tracing calculus are used to simulate the sensors operation and to measure distances. This way, the robot may detect an object, both when it moves forwards or makes a turn. Wheels are handled by two independent engines, that allows the robot to advance or to turn. Signals to active both engines are sent as real numbers in the interval  $[0,10]$ . Thus, to move forward, both wheels must advance an identical amount (sending the same value to both engines). In another case, the robot will turn towards the side whose engine received a smaller value of activation.

The simulator represents an environment of 400x400 pixels. Several obstacles (according to the problem) are included. The surface is supposed to be smooth. In order to make a movement, the robot must make the following steps:

1. Read the sensor activation values, obtaining the distance to the closest obstacles and the amount of mark on the ground (or "residues") in each sensor direction.
2. The controller receives the values and decides the movement to take.
3. The controller calculates how much the wheels must turn, and it sends these values to the engines.
4. The engines turn independently, making the robot either move forward or turn.

The simulator, both in the Linux and Win32 versions is available for download at the following web page: <http://atc.ugr.es/~pedro/robot>



**Fig. 1.** (a) Different configurations (states) of the obstacles with respect to the robot, taking into account the lateral and frontal sensors. (b) Movements to take to resolve a labyrinth following the wall on the left. First column shows the sensor values, "1" means that an obstacle is close in that direction (left, center or right sensor). Second column shows how much each wheel must turn (the action to take)

We propose to develop a controller that decides the next movement taking into account the obstacle position with respect to the sensors. A sensor may be detecting (or not) an object. Thus, there are eight possible configurations (states) of the obstacles with respect to the robot (see figure 1-a).

If a movement (action to take) is defined for each of the eight configurations (states), the controller will be able to guide the robot through the environment. The robot will react suitably on any object it faces, independently it is located.

As an example, figure 1-b defines a set of movements for the eight configurations, so the robot cross a labyrinth following the wall it has on the left.

The developed EA is specified in the following pseudocode:

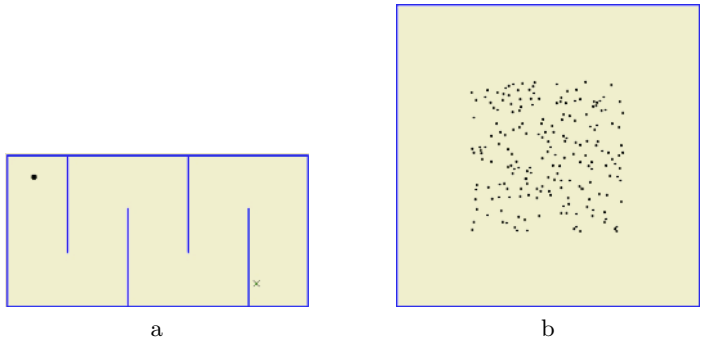
1. Generate an initial population of individual-controllers, initializing randomly the actions to take.
2. Repeat for  $g$  generations:
  - (a) evaluate the new individuals, obtaining their fitness value
  - (b) select the  $n$  best individuals, according to their fitness, to mate using the genetic operators to obtain the new ones
  - (c) replace the  $n$  worst individuals in the population by the new ones.

The mutation genetic operator randomly modifies the wheel turning values for some states that the individual codifies (depending on the mutation range). The crossover operator swaps some "movements" (action to take on a state) between two individual-controllers depending on the number of crossing points.

The fitness value of an individual (how the controller guides the robot during a number of steps) follows two criteria: first, the distance to the target is obtained. In the case of two individuals that guided the robot to the same distance, the one that collided least with the walls would be considered the best. In the case of being following a mark on the ground, the first criterium is the amount of tracked mark after moving during that number of steps.

### 3 Experiments

The experiments must be carried out using a set of dynamic environments, so that, a controller able to resolve several environments is more robust than another specialized in a concrete scene [4]. In this paper the resolution of two environments with different purposes and difficulties is proposed.



**Fig. 2.** (a) In the first experiment, the robot must cross the corridor without crashing against the walls. The starting point comes given by the point in the left part, while the target is pointed out by the X sign. (b) The second experiment consist of tracking a spread mark on the ground, avoiding the collisions with the walls

Experiments consisted of 10 simulations for each proposed environment (see figure 2). Due to the evaluation time needed for each controller (individual of the EA), experiments were set using 10 generations, and a population size of 20 individuals. To evaluate a controller, the robot is handled during a given number of steps. The evaluation finishes either if no more steps are left to take or the target is reached. The steps number was set to 200 (after some runs it was verified that that number is enough to reach the target). Each new evaluation, the robot is placed in a different place (close to the starting point), facing a new random direction to avoid the controller specializes and it always reaches the target starting from a very concrete point.

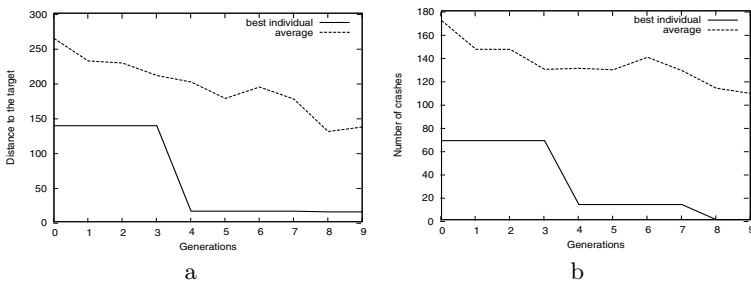
The first problem (see figure 2-a) consists of guiding the robot through a corridor, making turns both to the left and to the right, and avoiding collisions with the walls. The corridor presents the difficulty of the turns to the left and to the right, so that, controllers must resolve both to complete the route. Thus, a controller that only avoid obstacles by the left side, or only by the right side, or only by the front, will not be able to solve the problem.

In the second problem (see figure 2-b) the controller must handle the robot to track a spread mark on the ground (in the sense of cleaning some residues). It is mandatory to maximize the amount of tracked spot. This problem presents the difficulty that not only the robot must track the mark, but also it must avoid crashes against the walls.

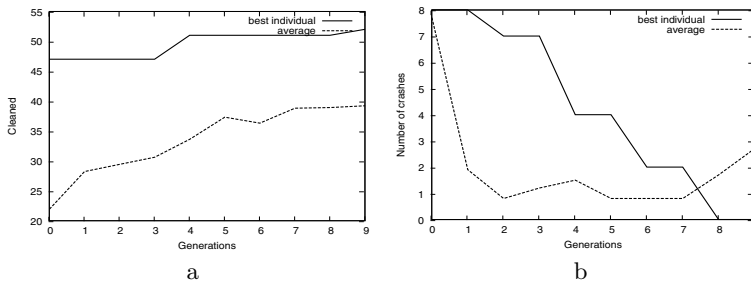
## 4 Results

Only in one run out of ten, the first problem could not be resolved. In the rest of the runs, a controller that handled the robot almost without collisions was obtained. Figure 3 shows the evolution of the robot distance to the target (a) and the number of collisions (b), in the case of the best individual in each generation and the average for all the controller population. As the simulation progresses, both the best individual and the population, on average, reduce the distance to the target and the number of crashes. Due to the difficulty of this problem, the average distance does not reduce in the same way that the best individual's distance to the target does, since not all the controllers are able to handle the robot through the corridor without crashes.

Figure 4 shows the amount of tracked mark (a) and the number of collisions (b) evolution. As the simulation progresses, the amount of spot grows, while the number of crashes reduces. Due to the absence of obstacles in the second problem environment, the controllers easily avoid to collide with the walls.



**Fig. 3.** Experiment 1: Distance to the target (a) and number of collisions (b) evolution during a typical simulation for the best individual and the population average. Both reduces as the simulation progresses



**Fig. 4.** Experiment 2: the amount of tracked mark (a) and the number of collisions (b) evolution during a typical simulation for the best individual and the population average. As the simulation progresses, the amount of spot maximizes, while the number of crashes is minimized



## 5 Conclusions and Future Work

Control system design is a complex task that can be carried out using an EA. This paper proposes a method based on an EA to design simple controllers that guide correctly an autonomous robot.

Two experiments are presented in which the controllers should guide the robot through two different environments. The first one consists of crossing a corridor, making turns both to the left and to the right, and avoiding collisions with the walls. The second one consists of tracking a mark on the ground, avoiding collisions with the walls. Simulations are set up the most realistic, introducing “noise” to modelize the system (sensor activations and wheels advances).

At the moment, we are improving the simulator and the algorithm to develop new controllers. A real robot is being developed, following the specifications given above: The hardware design consists of a parallel port board where the sensors and the actuators on the engines are connected. The computer reads the sensor activations and orders the movement to the wheels through the parallel port.

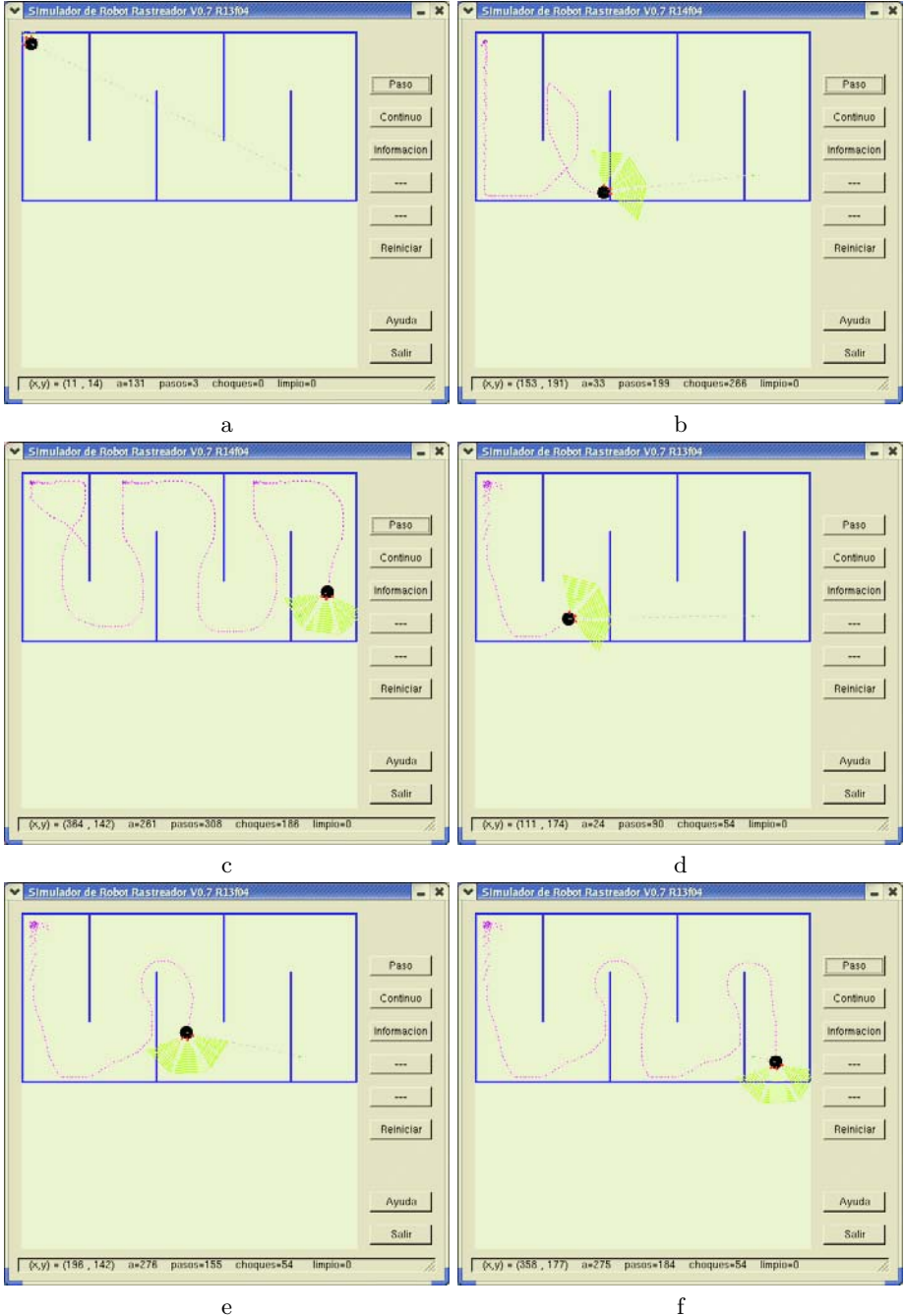
## Acknowledgements

This work has been supported by CICYT TIC2003-09481-C04-04 project.

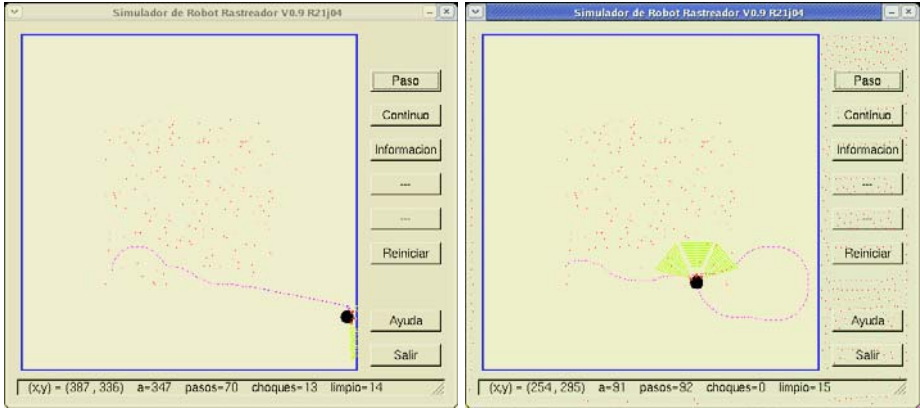
## References

1. R.A. Brooks. Artificial life and real robots. *In Proceedings of the First European Conference on Artificial Life*. MIT Press/Bradford Books, Cambridge, MA, 1992.
2. D. Cliff, I. Harvey, and P. Husbands. Explorations in evolutionary robotics. *Adaptive Behavior 2(1)*, pp.73-110, 1993.
3. D.E. Goldberg. Genetic algorithms in search, optimization and machine learning. *Addison-Wesley, Reading, MA*, 1989.
4. I. Harvey, P. Husbands, and D. Cliff. Issues in evolutionary robotics. *Proc. of the 2nd Intl. Conf. on Simulation of Adaptive Behaviour (SAB92)*, J.A. Meyer, H. Roitblat and S. Wilson editors, pp.364-373, 1993.
5. P. Husbands and I. Harvey. Evolution versus design: Controlling autonomous robots. *In Integrating Perception, Planning and Action, Proceedings of 3rd Annual Conf. on Artificial Intelligence, Simulation and Planning*, pp.139-146, 1992.
6. P. Nordin and W. Banzhaf. An on-line method to evolve behavior and to control a miniature robot in real time with genetic programming. *Adaptive Behaviour*, 5 (2) , pp.107-140, 1997.
7. H. Seraji and A. Howard. Behaviour-based robot navigation on challenging terrain: a fuzzy logic approach. *IEEE Transactions on Robotics and Automation*, vol.18, no.3, pp.308-321, 2002.
8. C. Ye, N.H.C. Yung, and D. Wang. A fuzzy controller with supervised learning assisted reinforcement learning algorithm for obstacle avoidance. *IEEE Transaction on Systems, Man and Cybernetics - Part B: Cybernetics*, pp.1-11, 2002.

**Table 1.** Typical situations during the simulations in the first experiment: (a) individual in the first stage of its evaluation; (b) bad controller that does not crosses the corridor; (c) the robot correctly crosses the corridor, colliding with the walls in some points; (d, e, f) three snapshots of the simulation where the best individual (controller) crosses the corridor

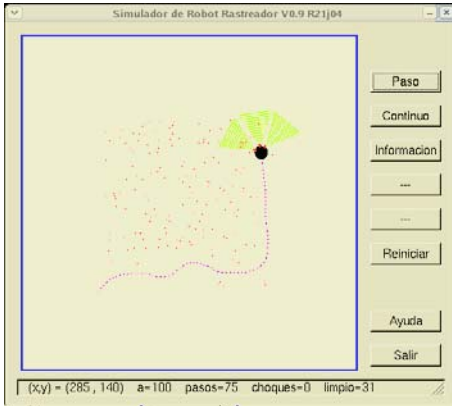


**Table 2.** Typical situations during the simulations in the second experiment: (a) bad individual that does not track the mark on the ground and crashes against the wall; (b) the robot loses the path, however, it is able to return to the spot; later on, it will evolve to give rise to better individuals; (c, d, e, f) it follows the contour of the mark and, if it loses the path, it return and to track the mark. As it follows the contour, it seems to move in spiral, reducing the spot

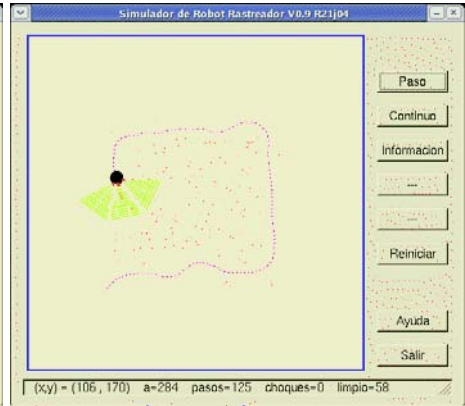


a

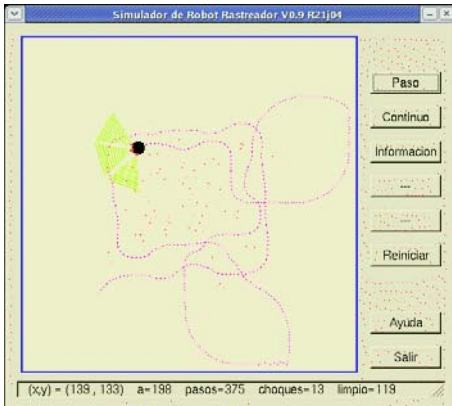
b



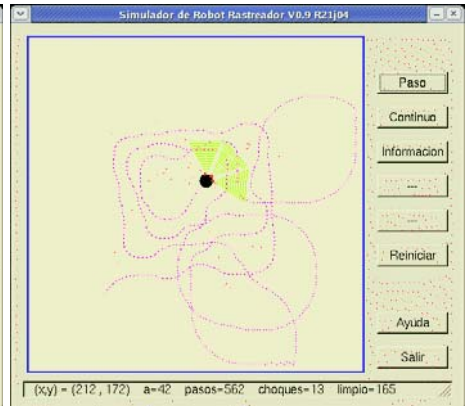
c



d



e



f

# Highly Modular Architecture for the General Control of Autonomous Robots

Ricardo A. Téllez, Cecilio Angulo, and Diego E. Pardo

Technical University of Catalonia, Jordi Girona,  
1-3, Barcelona, Spain  
r\_tellez@ouroboros.org  
cecilio.angulo@upc.edu  
diego.pardo@ben.colfuturo.org

**Abstract..** The implementation in a robot of the coordination between different sensors and actuators in order to achieve a task requires a high formulation and modelisation effort, specially when the number of sensors/actuators and degrees of freedom available in the robot is huge. This paper introduces a highly distributed architecture that is independent from the robot platform, capable of the generation of such a coordination in an automatic way by using evolutionary methods. The architecture is completely neural network based and it allows the control of the whole robot for, in principle, any type of task based on sensory-motor coordination. The article shows how the proposed architecture is capable of controlling an Aibo robot for the performance of three different difficult tasks (standing, standing up and walking) using exactly the same neural distribution. It is also expected that it will be directly scalable for higher levels of control and general design in evolutionary robotics.

## 1 Introduction

Creating a control system for a quadruped robot like Aibo is a very challenging task, since the number of degrees of freedom available is high and the sensorial information to process is huge. To obtain a coordination between the elements in order to perform a global robot task is difficult and is a problem not completely solved yet in a general sense. Furthermore, the solution happens to be more difficult when the control mechanism is completely neural based and the way the different modules should interact is not programmed by hand. Some works have been developed that use completely neural based controllers on wheeled robots [1][2]. Those generate sensory-motor patterns required for the task, mainly using behaviour-based systems, but there are none applied to robots as complicated as a quadruped. An exception could be those works that address the problem of walking in quadruped or biped robots by using neural oscillators and Central Pattern Generators (CPG) [3][4][5][6], like for example the use of CPGs for the control of several postures and movements [7], but those are always focused on CPG dependent tasks (walking, running, scratching) and not having on mind a general purpose sensory-motor task.

Having as a goal the complete control of the robot by artificial neural networks, together with the idea of generic control (any robot, any task), this paper introduces a general highly distributed architecture for the control of a robot on a sensory-motor coordination task. The idea is to have an architecture able to generate a coordination between different elements in a general way with independence of the task at hands and the robotic platform used. By taking as point of departure Minsky's idea of *society of mind* [8] and the idea of *modularity of mind* by Fodor [9], we see the robot's mind as a group of different modules each one in charge of its own device (sensor or actuator) that interacts with the rest of modules, where module is something similar to Fodor's definition: domain-specific processing systems, with their own proprietary transducers, and delivering non-conceptual outputs. The interaction of those modules produces as effect the accomplishment of the task by the whole robot.

In order to show the results of our research, the article has been organised as follows. Section 2 describes the architecture developed including the learning mechanism. Section 3 shows the results obtained when the architecture was implemented on an Aibo robot in both simulation and real robot. Section 4 generates conclusions from the results and points towards future work based on those results.

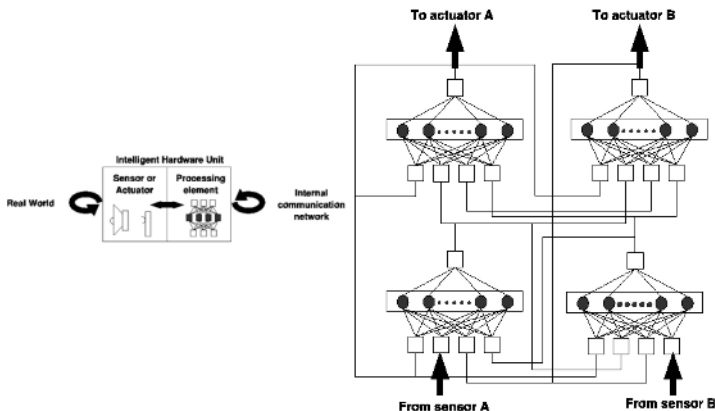
## 2 Architecture Definition

The architecture is based on several uniform modules, composed of neural networks, where each module is in charge of one element of the robot. Through the use of a neuro-evolutionary algorithm, modules learn how to cooperate between them and how to control its associated element, allowing the whole robot to accomplish the task at hands.

### 2.1 Hardware

Using Fodor's definition of module, we define the Intelligent Hardware Unit (IHU) as a module created around a physical device (sensor or actuator). Every IHU is composed by a sensor or by an actuator and a micro-controller implementing an artificial neural network (ANN) that process the information of its associated device (received sensor information for sensors, commands sent to the actuator for actuators). It is said that the ANN is in charge of its sensor/actuator. This means that is the neural net the one that decides which commands must be sent to the actuator, or how must it be interpreted a value received from a sensor, and under which circumstances. All IHUs are interconnected to each other in order to be aware of what the other IHUs are doing. So in some sense, the net is also in charge of deciding what to say to the other elements as well as to interpret what the others are saying. The structure of a IHU can be seen in the following figure, together with a neural controller for a simple system with two sensors and two actuators.

It should be stated that when put several IHU together on a control task, each element has its own particular vision of the situation because each one is



**Fig. 1.** Schematics of an IHU, and the connection schema of four ANNs from four IHUs controlling a simple robot

in charge of its own sensor or actuator. This leads to a situation where each unit knows what the others are doing but needs to select an action for its controller or sensor output, and based on its knowledge of the global situation and that of its particular device, decides what the next action will be.

Even though in the original definition a microprocessor was required for any IHU element, on the experiments presented here it has been simulated the existence of the micro-controllers linked to each device by allocating some processing time in the robot central processor for each IHU, since it was not physically possible to have one dedicated micro-controller for each IHU, neither in the simulations, nor in the real robot tests. It will be assumed that the results are not very different from the original idea.

## 2.2 Neuro-evolutionary Algorithm

To teach the networks the coordination required for the task a neuro-evolutionary approach has been selected. For the Co-evolution of the different networks and due to the necessity of evolving different ANNs for different roles on a common task, a co-evolutionary algorithm is required. By using such kind of algorithm it is possible to teach to the networks how they must cooperate to achieve a common goal, when every network has its own an different vision of the whole system.

The algorithm selected to evolve the nets is the ESP (Enforced Sub-Populations) [10][11], which has been proved to produce good results on distributed controllers [12]. This algorithm is also in principle free of bias for a special task. It is a general algorithm which produces good results in systems where several parts must interact to obtain the general view of the situation.

The chromosome codes in a direct way the weights of the network connections. A chromosome is generated for each IHU's network, and all are evolved at the same time over the same situation.

Usually in ESP, neurons in the hidden layer are also connected to themselves and to the other neurons of the same layer. This information is also coded in the chromosome. However, in the robot presented in this work and for the sake of simplicity, hidden neurons neither have been connected to themselves nor to the other neurons of the same layer.

### 3 Implementation

In order to validate the architecture explained in previous section, it was performed an implementation on a complex robot. Based on previous results where the architecture was tested on a very simple simulated robot [13], the architecture was later tested for the control of a complex real and simulated robot with several sensors and actuators while performing a task.

#### 3.1 Validation with a Complex Robot

For a validation of the architecture on real robots the Aibo robot was selected. This is a complex robot with several degrees of freedom and multiple sensors and actuators that requires a good coordination between them to achieve any simple task. The aim for this stage was to see if the architecture was capable of controlling such a complex robot in the realization of some simple tasks and a more complicated one. First task was to keep an up position; second was to stand up from a lying position, and third task was to generate a gait behaviour.

For the realization of those experiments a simulator and a real robot were used. The simulator selected was the Webots program by Cyberbotics, which allowed the easy evolution of the controller without having to use the real robot [14]. This introduced easiness in the experiments since the simulator had the possibility of letting the system running alone doing training. Also damaging the robot and problems with batteries were avoided by using the simulator. Once the controller was generated and tested on the simulator, a direct connection was established to the real robot, that allowed the test of the resulting neural controller on it. At this stage of the Webots simulator development (version 4.0.28beta) only direct connection of the simulator to the real robot is available, since the *cross-compilation of controller for real Aibo* feature was not working yet for the ERS-7 model. Direct connection means that the controller runs on the computer (under Webots) but controls the real robot via wireless connection.

For the control of the robot the following sensors and actuators were taken into account:

- Actuators: four leg upper joints (J1), four leg middle joints (J2), four leg lower joints (J3). These are all motors that move Aibo’s joints and determine its position on space.
- Sensors: four leg upper joints, four leg middle joints, four leg lower joints, four legs paw sensors. These are the sensors that indicate the state of the joint motors (actuators). Last four paw sensors indicate the state of the feet

paws. Those are switched on when the feet touches the ground or off when not touching. Furthermore, three accelerometer sensors (X,Y and Z) were used for determining Aibo's position.

This gives a total of 31 sensors and actuators, making the required number of ANNs to 31. All nets have the same number of inputs (31), outputs (1) and hidden units (8) for all the experiments. For the nets of the controllers, the inputs are connected to the outputs of all the other nets including itself. For the sensor networks are connected in the same way except that the entry from itself comes from the real sensor instead. For actuators, nets output indicate the kind of value that needs to be sent to the joint. For sensors, the output indicates the sensor value that is reported to the rest of IHU (including itself).

Values obtained from sensors were quantified, allowing a precision of one degree. This quantification generates a kind of dumping mechanism that prevents undesirable oscillations and never ending training. Network outputs of sensors were quantised in the same way. Network outputs of the actuators were quantized to provide only three possible values: move joint for 0.05 radians up, move joint for 0.05 radians down, or not move the joint.

Other required parameters for the evolution that were kept the same in all experiments are represented on table 1, and were selected based on previous experiences.

**Table 1.** Parameters used during neuro-evolution

| Parameter Name         | Parameter Value |
|------------------------|-----------------|
| Subpopulations         | 8               |
| Size of subpopulations | 40              |
| Mutation rate          | 0.4             |
| Stagnation             | 20              |

**First Test: Keep Standing Up Position.** The architecture was first tested to keep the robot on a standing up position . The robot was set up on an initial standing up position on the space and let free to act using its control networks. The goal was to keep up as high as possible with as less joint movements as possible. This seems a simple experiment but it is not because networks are directly connected to actuators and they keep on sending movement commands to the actuators all the time. This implies that joints will keep on moving unless the not-move-joint command is decided by their associated nets. This continuous movement of joints could lead to strange robot positions and eventually make it fall. In this task then, the robot must learn how to achieve a stable high position and keep it until the end of the evaluation time. For this purpose, the following fitness function was defined:

$$fitness = finalHeight / numberOfMovements$$



which evaluates the robot for its final height in opposition to the number of movement commands sent to the joints. Evaluation time for each phenotype was time steps.

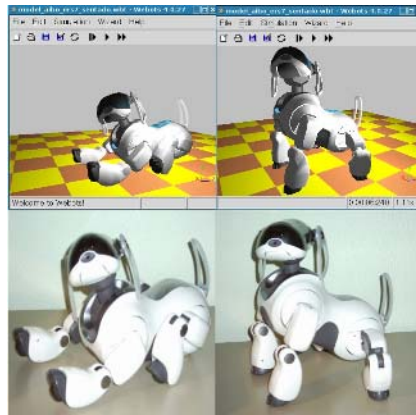
After 13 generations the robot learned how not to fall down and how to keep a good structure to stay stable, but still too many movements were realized. After generation 28, movements were reduced to a reasonable value while keeping the standing up position in both simulator and real robot.

**Second Test: Stand Up.** In this case the robot was required to change from a laying down position to a standing up one. For this purpose, the same architecture as in the previous experiment was used, been the only difference the fitness function and the time of evaluation of each phenotype, that in this case was of time steps.

The fitness function was defined as follows:

$$fitness = paws * finalHeight$$

Results showed that the robot was able to change its position to the desired one (see figure 2) after 77 generations in both simulator and real robot.



**Fig. 2.** Figures showing initial laying down position and final stand up obtained position of Aibo in both simulation and real robot

**Third Test: Walking.** This is by far the most difficult of the tests performed, and by hence, it was the one that gave more problems.

When designing controllers for robot walkers there are two main approaches: the first one uses a walking algorithm designed by hand by an engineer which indicates at any step the position of every joint of the robot [15][16][17]. Usually this approach has a set of parameters to be tuned, and they can be done of several ways (by hand, by try and modify, by genetic algorithms, etc...). The

resulting gait is one perfectly engineered that behaves exactly in the way the algorithm says.

The second approach consists of using CPGs. CPGs are non-linear oscillators made of neural nets. CPGs are coupled between them to generate the control signal for the actuators taking or not into account sensors information. The architecture developed here is very similar to that of CPGs, since it implements a group of neurons connected to the actuators, but it implements more connectivity and receives information from all sensors.

In this case, the architecture was implemented using recurrent neural nets instead of simple feedforward ones, and the fitness function was developed to conditionate the robot to acquire a determined gait style, based on the observance of Sony's walking style.

$$fitness = height * distance$$

Rule 1: fitness = 0 if J1 joints out of (-0.698,0.261) radians

Rule 2: fitness = 0 if J3 joints out of (0.349,1.658) radians

Rule 3: fitness = 0 if less than three paws down

Rule 4: fitness = 0 if after one front paw up does not follow a back paw up

For this experiment, an incremental evolutionary approach was taken, where the robot was let to neuro-evolve the controller for a few generations with rule 1 activated. After a determined number of evaluation steps rule 2 was activated, and so happens with rule 3 and 4. After applying four rules for several generations a basic walking pattern arise.

## 4 Discussion

Results presented here show that the neural architecture proposed is able to control a complex robot, and generate the required coordination mechanisms between sensors and actuators to perform complex tasks like walking or standing up. Coordination is learned during training phase and then used to solve the task during test phase with a completely neural based controller.

Experiments were performed in a general way, it is, there was no special consideration neither for the robot being controlled nor for the task being developed. Having implemented the architecture in such a way, it could be suggested that the architecture would be capable of the generation of any sensory-motor coordination required for any other task on any other robot, been the only limitations those imposed by evolutionary robotics. We admit thought, that the architecture has only been tested on two simulation robots (see author's previous work [13]) and only one real robot. Nevertheless, some research is been performed at present by the authors to apply this architecture to different types of robots on different tasks.

We think this work is a step closer to the final goal of obtaining a completely neurally controlled robot were more deliberative tasks could be performed.

## Acknowledgements

Authors would like to thank Olivier Michel for their support and big help on Webots issues, and Auke Ijspeert for insightful suggestions and comments on walking robots.

## References

1. Nolfi, S., Floreano, D.: *Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines*. The MIT Press (2000)
2. Floreano, D., Mondada, F.: Evolutionary neurocontrollers for autonomous mobile robots. *Neural Networks* **11** (1998) 1461–1478
3. Hiroshi Kimura, S.A., Sakurama, K.: Realization of dynamic walking and running of the quadruped using neural oscillator. *Autonomous Robots* **7** (1999) 247–258
4. Hiroshi Kimura, Y.F., Konaga, K.: Adaptive dynamic walking of a quadruped robot by using neural system model. *Advanced Robot* **15** (2001) 859–876
5. Collins, J., Richmond, S.: Hard-wired central pattern generators for quadrupedal locomotion. *Biological Cybernetics* **71** (1994) 375–385
6. Reeve, R.: *Generating walking behaviours in legged robots*. PhD thesis, University of Edinburgh (1999)
7. Billard, A., Ijspeert, A.J.: Biologically inspired neural controllers for a motor control in a quadruped robot. *Proceedings of the International Joint Conference on Neural Network* (2000)
8. Minsky, M.: *The Society of Mind*. Touchtone Books (1988)
9. Fodor, J.: *The modularity of mind*. Cambridge, MA: MIT Press (1983)
10. Gmez, F., Miikkulainen, R.: Solving non-markovian control tasks with neuroevolution. In: *Proceedings of the IJCAI99*. (1999)
11. Gomez, F., Miikkulainen, R.: Incremental evolution of complex general behavior. Technical Report AI96-248, University of Texas (1996)
12. Yong, H., Miikkulainen, R.: Cooperative coevolution of multiagent systems. Technical Report AI01-287, Department of computer sciences, University of Texas (2001)
13. Tllez, R., Angulo, C.: Evolving cooperation of simple agents for the control of an autonomous robot. In: *Proceedings of the 5th IFAC Symposium on Intelligent Autonomous Vehicles*. (2004)
14. Michel, O.: Webots: Professional mobile robot simulation. *Journal of Advanced Robotics Systems* **1** (2004) 39–42
15. Hornby, G.S., Fujita, M., Takamura, S., Yamamoto, T., Hanagata, O.: Autonomous evolution of gaits with the sony quadruped robot. In Banzhaf, W., Daida, J., Eiben, A.E., Garzon, M.H., Honavar, V., Jakiela, M., Smith, R.E., eds.: *Proceedings of the Genetic and Evolutionary Computation Conference*. Volume 2., Orlando, Florida, USA, Morgan Kaufmann (1999) 1297–1304
16. Hornby, G., Takamura, S., Hanagata, O., Fujita, M., Pollack, J.B.: Evolution of controllers from a high-level simulator to a high DOF robot. In: *ICES*. (2000) 80–89
17. Rfer, T.: Evolutionary gait-optimization using a fitness function based on proprioception. *Lecture Notes in Artificial Intelligence* (2005)

# Using PSOMs to Learn Inverse Kinematics Through Virtual Decomposition of the Robot

Vicente Ruiz de Angulo and Carme Torras

Institut de Robòtica i Informàtica Industrial (CSIC-UPC),  
Llorens i Artigas 4-6, 08028-Barcelona, Spain  
{ruiz, torras}@iri.upc.edu  
www-iri.upc.es

**Abstract.** We propose a technique to speed up the learning of the inverse kinematics of a robot manipulator by decomposing it into two or more virtual robot arms. Unlike previous decomposition approaches, this one does not place any requirement on the robot architecture and, thus, it is completely general. Parametrized Self-Organizing Maps (PSOM) are particularly adequate for this type of learning, and permit comparing results obtained directly and through the decomposition. Experimentation shows that time reductions of up to two orders of magnitude are easily attained.

## 1 Introduction

Neural networks have proved useful for learning the inverse kinematics of robot manipulators, either lacking a well-defined model or needing on-line recalibration while functioning. The main shortcoming is the large number of training samples (i.e., robot movements) required to attain an acceptable precision [2, 3].

Several attempts have been made at reducing the number of required samples, among them the use of hierarchical networks [4, 10], the learning of only the deviations from the nominal kinematics [5], and the use of a continuous representation by associating a basis function to each knot [9].

In [6, 7] we proposed a practical trick that can be used in combination with all the methods above. It consists in decomposing the learning of the inverse kinematics into several independent and much simpler learning tasks. This was done at the expense of sacrificing generality: the procedure works only for some robot models subject to certain types of deformations. Specifically, the procedure assumes that the last three robot joints cross at a point, a condition satisfied by some classic robot architectures.

Here we present another decomposition technique for learning inverse kinematics that is not limited by the above assumption. While being more general, it still retains the main advantage of the trick above: The input dimensionality of each of the tasks resulting from the decomposition is half that of the original one. Thus, for a given desired accuracy, if the number of training samples required to learn inverse kinematics directly is  $O(n^d)$ , through the decomposi-

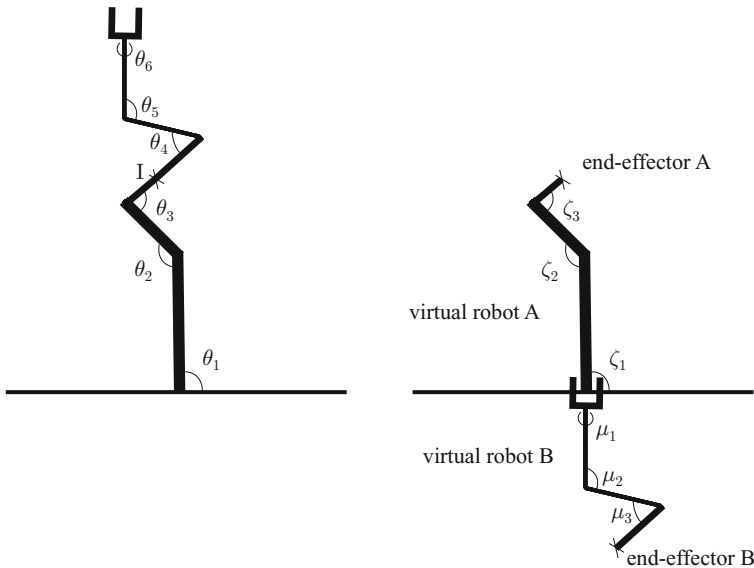
tion it reduces to  $O(n^{d/2})$ . This yields an enormous reduction in the number of samples required for high-precision applications.

The paper is structured as follows. In the next section we describe the proposed decomposition of the inverse kinematics mapping. Section 3 explains how the workings of two parameterized self-organizing maps (PSOMs) encoding the kinematics of the component virtual robot robots can be combined to provide the inverse kinematics of the original robot. The following two sections are devoted to the training scheme and the way to retrieve the kinematics from the component PSOMs, respectively. In Section 6, some illustrative experimental results of learning with and without the decomposition are presented, permitting to quantify the savings obtained. Finally, some conclusions and prospects for future work are put forth in Section 7.

## 2 Kinematics Decomposition

The technique described here is based on the idea of decomposing the kinematics of a serial manipulator into those of several “virtual robots”. The advantage of the approach is that, since the component robots are much simpler than the original one, the learning of their inverse kinematics requires less sampling points to be acquired.

We will explain the technique using only two virtual robots (see Fig. 1). The extension to more virtual robots is straightforward. Let  $\theta = (\theta_1, \theta_2 \dots \theta_n)$  and



**Fig. 1.** Decomposing the robot manipulator (left) into two virtual robot arms (right)

$T_1, T_2, \dots, T_n$  be the joints and the associated transformation matrices, respectively, of the real robot. To connect the two robots we select a reference frame  $\mathcal{A}$  rigidly linked to  $A_k$ , the reference frame of joint  $k$ . Thus,  $\mathcal{A} = A_k A_c$ , where  $A_c$  is a constant matrix. It can be, for example, a reference frame centered on any point  $I$  of link  $k$ . Ideally  $k = n/2$ .

The first virtual robot, or robot A, has  $k$  joints  $\zeta = (\zeta_1, \zeta_2, \dots, \zeta_k)$ , and their associated transformation matrices are  $T_1, T_2, \dots, T_{k-1}, T_k A_c$ . The second robot, or robot B, is composed of  $n-k$  joints  $\mu = (\mu_1, \dots, \mu_{n-k})$  with associated reference matrices  $T_n^{-1}, T_{n-1}^{-1}, \dots, T_{k+2}^{-1}, (T_{k+1} A_c)^{-1}$ .

We could consider that we have virtually broken the original robot into two pieces, exactly at point  $I$  of link  $k$ . Robot A is the first piece of the robot and has its end-effector at the extreme of the broken link. Robot B is the other piece of the original robot, the base of robot B being the original end-effector (translated and rotated to the origin of coordinates), and the end-effector of robot B, the extreme of the other half of the broken link. The second robot can also be seen as the remaining of the original robot, inverted and translated to the reference frame.

By  $\theta = (\zeta, \mu)$  we mean

$$\theta_i = \zeta_i, \quad \forall i = 1 \dots k \quad (1)$$

$$\theta_i = \mu_{n-i+1}, \quad \forall i = k + 1 \dots n \quad (2)$$

We denote  $DK_A(\zeta)$  and  $DK_B(\mu)$  the direct kinematics of robots A and B, respectively. It is easy to see that  $\theta = (\zeta, \mu)$  is a valid inverse kinematics solution for a given position  $X$  and orientation  $\Omega$  of the real robot iff

$$DK_A(\zeta) = TR(X, \Omega) DK_B(\mu), \quad (3)$$

where  $TR(X, \Omega)$  is the matrix transformation yielding a translation  $X$  and an orientation  $\Omega$ .

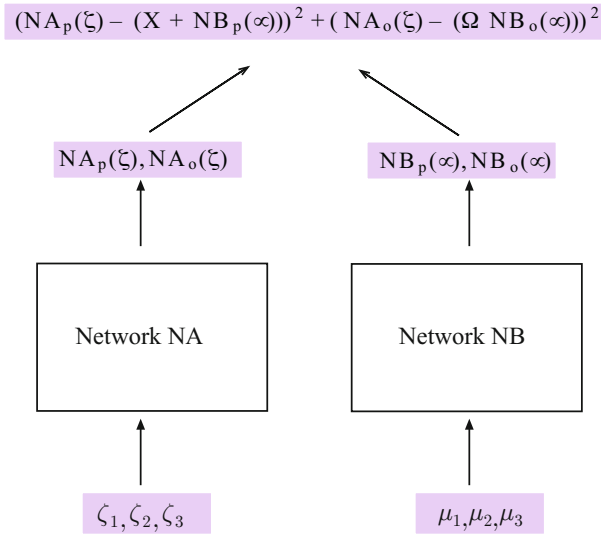
### 3 Kinematics Composition

The approach consists in creating two neural networks (or any other interpolators)  $NA$  and  $NB$  approximating the functions  $DK_A(\zeta)$  and  $DK_B(\mu)$  (see Fig. 2). When the joint values of pose  $(X, \Omega)$  are required, a search in the inputs of the two networks is carried out to find values of  $\zeta$  and  $\mu$  satisfying (3) as much as possible. This can be done by imposing a common cost function to be minimized in  $(\zeta, \mu)$  such as

$$(NA(\zeta) - TR(X, \Omega) NB(\mu))^2, \quad (4)$$

or by decomposing the output of the networks into two components: position  $(NA_p, NB_p)$  and orientation  $(NA_o, NB_o)$ , and then minimizing:

$$(NA_p(\zeta) - (X + NB_p(\mu)))^2 + (NA_o(\zeta) - (\Omega NB_o(\mu)))^2. \quad (5)$$



**Fig. 2.** The workings of the two networks are linked through the cost function at the top

To facilitate the search we require the output of the neural networks to be differentiable with respect to the input. We consider that the memory-based neural networks are especially well suited to our application, since they use stored function points to build the approximation of the function. On the one hand, they allow a quick search among the stored points to find a good starting point for continuous minimization. On the other hand, we can apply  $TR(X, \Omega)$  to the stored points of network  $NB$ , so that the whole approximation of the function gets translated and rotated, becoming  $NB'$ . In this way, the function to be minimized becomes  $(NA(\zeta) - NB'(\mu))^2$ , whose derivatives are more easily obtained.

A Parametrized Self-Organized Map (PSOM) [9] is the type of network better suited to our requisites. It approximates a function using a regular grid of sampled points. Because of its excellent interpolation capabilities, the required number of points is very small. Of particular interest to us is that PSOMs treat input and output variables in the same way. This means that it is as natural to ask which output corresponds to a given input as asking which input correspond to a given output. Therefore, our search in the input variables is naturally addressed and embedded in the framework of these networks.

### 4 Learning the Inverse Kinematics of the Virtual Robots

Usual inverse kinematics learning requires the capability to observe the position and orientation of the robot end-effector, represented by the transformation ma-

trix  $M$ . Our method requires also knowing the position and orientation of the point  $I$ , encapsulated in the transformation matrix  $M_I$ .

Every time the robot performs a movement (even during working operation), a sampling point for each of the virtual robots can be obtained. The learning amounts to supplying virtual robot A with a sampling point consisting of an input  $\zeta_i = \theta_i, i = 1 \dots k$  and an output  $M_I$ . For robot B the sampling point has as input  $\mu_i = \theta_{n-i+1}, i = 1 \dots n - k$  and as output  $M^{-1}M_I$ . We could understand this as moving the whole robot B “frozen” in its current configuration to the place it is supposed to be, before extracting its kinematics sample point.

When using PSOMs to learn the kinematics of the virtual robots, the movements are generated following a regular grid in the space of joint angles.

## 5 Computing Kinematics with PSOMs

Once trained, a PSOM works by putting some constraints on a subset of the variables of the system (input or output), for example fixing them to a desired value. The system then carries out a quick optimization aimed to find a point of the approximated input-output manifold satisfying the constraints or, if impossible, the closest one to satisfying them. The starting point of the process is the stored point that best satisfies the constraints. From it, an iterative minimization procedure is launched, which finishes in a few steps.

For PSOMs trained on the kinematics of a robot, to get the inverse kinematics we simply fix the position and orientation variables and we let the minimization get the point in the interpolating surface with the desired pose values, the remaining components of the point are taken to be the result. To obtain the inverse of the real robot using the PSOMs for the virtual robots, we first transform the points stored in  $NB$  with the desired pose, as explained in Section 3. Afterwards, we look for a good starting point for the minimization by finding the closest pair (in pose space) between the points stored in  $NA$  and the transformed points in  $NB$ . Let  $(A_0, B_0)$  be this closest pair. A minimization step is then carried out in  $NA$  with  $B_0$  as target pose, and another step is done in  $NB$  with  $A_0$  as desired pose. The result of the step in  $NA$  and  $NB$  are two points whose pose components are  $A_1$ , and  $B_1$ , respectively. These points will be the starting point for the following steps in which the desired poses for  $NA$  and  $NB$  will be  $B_1$  and  $A_1$ , respectively. More iterations are performed in the same way, until  $A_i$  and  $B_i$  are closer than a certain threshold. Then we extract the inverse kinematics of the real robot by concatenating the joint components of the last obtained points.

## 6 Experiments

The experiments have been executed in a new general simulation environment developed at our institute, which allows the visualization of any serial manipulator. The only input needed for the simulator is a Denavit-Hartenberg table,



**Table 1.** Classic algorithm

| number of movements | position mean error | position stdev. error | orientation mean error | orientation stdev. error |
|---------------------|---------------------|-----------------------|------------------------|--------------------------|
| 64                  | 476                 | 229                   | 0.927                  | 0.635                    |
| 729                 | 46                  | 21                    | 0.101                  | 0.049                    |
| 4096                | 11                  | 17                    | 0.012                  | 0.027                    |

**Table 2.** Decomposition algorithm

| number of movements | position mean error | position stdev. error | orientation mean error | orientation stdev. error |
|---------------------|---------------------|-----------------------|------------------------|--------------------------|
| 8                   | 377                 | 236                   | 0.770                  | 0.653                    |
| 27                  | 48                  | 42                    | 0.092                  | 0.045                    |
| 64                  | 10                  | 35                    | 0.016                  | 0.059                    |
| 125                 | 3.6                 | 27                    | 0.005                  | 0.049                    |
| 216                 | 2.1                 | 8.3                   | 0.002                  | 0.011                    |
| 343                 | 1.6                 | 6.4                   | 0.002                  | 0.014                    |
| 512                 | 0.9                 | 2.9                   | 0.002                  | 0.021                    |

from which the graphical model is created using a uniform link and joint representation.

We used a PSOM variant known as LPSOM. This model builds a PSOM extracting for each query a subgrid of the sampling grid, which is centered on the closest point to the query. The representation of pose orientation has been thoroughly studied and different alternatives have been compared experimentally [8]. There exist many possible representations, but none is completely satisfactory. For example, the Euler representation is very compact, but lacks continuity. This drawback affects also other in principle good candidate representations such as quaternions. The classical  $3 \times 3$  rotation matrix is continuous but not compact. The solution was to select a subset of elements of the standard rotation matrix that determine it. The five elements in the last column and row are good in general, although not perfect because the matrix is not determined in one point (when the common element of the last row and last column takes a value of 0). Therefore, it is safer to use 6 elements, the last two columns of the rotation matrix, which completely determine it.

We have chosen the well-known PUMA robot to validate our technique. The experiments were carried out using a very large workspace, allowing ranges for the six joints [1] as follows: [-150,-35],[-215,-100],[-35,80],[-110,5],[-100,15],[-100,15]. We trained one LPSOM in a classical way, by generating samples of the kinematics of the robot in a regular grid in the joint space covering the workspace above. Then we moved the robot to the different configurations represented in the grid to obtain the associated positions and orientations. Thus, each knot in the grid requires one movement. The results are shown in Table 1. The units are millimeters and radians.

In the experiment to test our decomposition approach, we used two smaller PSOMs, one for each of the two virtual robots A and B. The corresponding regular grids were also generated. In this case, with only one movement of the robot, we get the required information for one knot of robot A and another knot of robot B. Table 2 shows the results. The comparison of both tables reveals that, for the only number of points in common (64), the averages in position and orientation are around 50 times more precise for the decomposition algorithm. We note also that the limits of physical accuracy of the manipulator are approximately reached with 512 movements with the decomposition algorithm, whereas it was impossible with our computer memory resources (allowing grids of up to 262,144 points) to reach precisions under 1 mm and .01 radians with the classic procedure.

## 7 Concluding Remarks and Future Work

The purpose of this paper is to propose a technique to learn inverse kinematics (IK) with a reasonable number of movements when high accuracy is required.

Unlike our previous work on IK learning through function decomposition [6, 7], the technique here proposed doesn't place any restriction on the type of robot architecture to which it can be applied. The kinematics of any serial manipulator undergoing whatever deformation can be learned with this technique. However, a new "sensorial" requisite must be fulfilled: the reference frame attached to a point in an intermediate link of the robot must be known using some sensing system. We think that this is not a shortcoming, since learning IK with any procedure requires anyway a sensorial system to determine the position and orientation of the gripper.

One of the most promising applications of our technique is to flexible robots. Since it reduces the dimensionality of the functions to be learned from 6 to 3, it is still affordable to include the load change as an extra variable and still have quick learning.

In addition to learning efficiency, our technique has other advantages over classic IK learning. For instance, it allows the robot to learn to move in the complete workspace without actually moving everywhere, and to approach risk zones only after learning has been successfully completed.

Among the tasks left for future work, we can mention testing the extension of this framework to more than two virtual robots. Also, we think that appropriately weighting the learning of the position and orientation of the two virtual robots can further improve the results. The inaccuracies in the interpolated position of the virtual subrobots are simply added (vectorially) in the composed robot. Instead, inaccuracies in the orientation components of the subrobots result in orientation inaccuracies of the same order in the composed robot, but also add a possibly large error component in position.

An open issue common to all the approaches to IK learning is the representation of orientation. We think that the goodness of representations for orientation can be evaluated with respect to three criteria: 1) compactness, 2) continuity, and

3) whether interpolated representations are proper representations. Compactness saves memory (especially in memory-based models) and should influence positively generalization. But continuity (two close orientations in the representation space should also be close as regards to robot motion) has a more radical influence in the quality of the interpolation. Finally, it is desirable that every interpolated representation corresponds to a true orientation. Otherwise, one has the problem of how to map interpolated values onto the representation space, as it happens with rotation matrices. By choosing as representation six elements of the rotation matrix, we have given priority to the continuity criterion, while trying to maximize compactness. Interpolated representations do not correspond necessarily to points inside the representation space, but this does not seem a big problem in practice.

**Acknowledgements.** This work was supported by the I+D project DPI 2004-07358 of the Spanish Ministry of Education.

## References

1. K.S. Fu, R.C. González and C.S.G. Lee: *Robotics: Control, Sensing, Vision, and Intelligence*. New York: McGraw-Hill (1987)
2. B.J.A. Kröse and P.P. van der Smagt: *An Introduction to Neural Networks* (5th edition), Chapter 7: "Robot Control". University of Amsterdam (1993)
3. T.M. Martinetz, H.J. Ritter and K.J. Schulten: Three-dimensional neural net for learning visuomotor coordination of a robot arm. *IEEE Trans. on Neural Networks* **1** (1990) 131-136
4. H. Ritter, T. Martinetz and K.J. Schulten: *Neural Computation and Self-Organizing Maps*. New York: Addison Wesley (1992)
5. V. Ruiz de Angulo and Torras C.: Self-calibration of a space robot. *IEEE Trans. on Neural Networks* (1997) **8** 951-963
6. V. Ruiz de Angulo and C. Torras: Learning inverse kinematics via cross-point function decomposition. *Proc. Intl. Conf. on Artificial Neural Networks (ICANN-02)*, *Lecture Notes in Computer Science* (2002) **2415** 856-861
7. V. Ruiz de Angulo and C. Torras: Speeding up the learning of robot kinematics through function decomposition. *IEEE Trans. on Neural Networks* (to appear)
8. D. Saune Sánchez: Recalibración de un brazo robot mediante técnicas de descomposición de la cinemática. Proyecto final de carrera, Departament LSI, Universitat Politècnica de Catalunya (2003)
9. J. Walter and H. Ritter: Rapid learning with parametrized self-organizing maps. *Neurocomputing* (1996) **12** 131-153
10. J. Walter and K.J. Schulten: Implementation of self-organizing neural networks for visuo-motor control of an industrial arm. *IEEE Trans. on Neural Networks* (1993) **4**

# Highly Modular Architecture for the General Control of Autonomous Robots

Ricardo A. Téllez, Cecilio Angulo, and Diego E. Pardo

Technical University of Catalonia, Jordi Girona,  
1-3, Barcelona, Spain  
r\_tellez@ouroboros.org  
cecilio.angulo@upc.edu  
diego.pardo@ben.colfuturo.org

**Abstract..** The implementation in a robot of the coordination between different sensors and actuators in order to achieve a task requires a high formulation and modelisation effort, specially when the number of sensors/actuators and degrees of freedom available in the robot is huge. This paper introduces a highly distributed architecture that is independent from the robot platform, capable of the generation of such a coordination in an automatic way by using evolutionary methods. The architecture is completely neural network based and it allows the control of the whole robot for, in principle, any type of task based on sensory-motor coordination. The article shows how the proposed architecture is capable of controlling an Aibo robot for the performance of three different difficult tasks (standing, standing up and walking) using exactly the same neural distribution. It is also expected that it will be directly scalable for higher levels of control and general design in evolutionary robotics.

## 1 Introduction

Creating a control system for a quadruped robot like Aibo is a very challenging task, since the number of degrees of freedom available is high and the sensorial information to process is huge. To obtain a coordination between the elements in order to perform a global robot task is difficult and is a problem not completely solved yet in a general sense. Furthermore, the solution happens to be more difficult when the control mechanism is completely neural based and the way the different modules should interact is not programmed by hand. Some works have been developed that use completely neural based controllers on wheeled robots [1][2]. Those generate sensory-motor patterns required for the task, mainly using behaviour-based systems, but there are none applied to robots as complicated as a quadruped. An exception could be those works that address the problem of walking in quadruped or biped robots by using neural oscillators and Central Pattern Generators (CPG) [3][4][5][6], like for example the use of CPGs for the control of several postures and movements [7], but those are always focused on CPG dependent tasks (walking, running, scratching) and not having on mind a general purpose sensory-motor task.

Having as a goal the complete control of the robot by artificial neural networks, together with the idea of generic control (any robot, any task), this paper introduces a general highly distributed architecture for the control of a robot on a sensory-motor coordination task. The idea is to have an architecture able to generate a coordination between different elements in a general way with independence of the task at hands and the robotic platform used. By taking as point of departure Minsky's idea of *society of mind* [8] and the idea of *modularity of mind* by Fodor [9], we see the robot's mind as a group of different modules each one in charge of its own device (sensor or actuator) that interacts with the rest of modules, where module is something similar to Fodor's definition: domain-specific processing systems, with their own proprietary transducers, and delivering non-conceptual outputs. The interaction of those modules produces as effect the accomplishment of the task by the whole robot.

In order to show the results of our research, the article has been organised as follows. Section 2 describes the architecture developed including the learning mechanism. Section 3 shows the results obtained when the architecture was implemented on an Aibo robot in both simulation and real robot. Section 4 generates conclusions from the results and points towards future work based on those results.

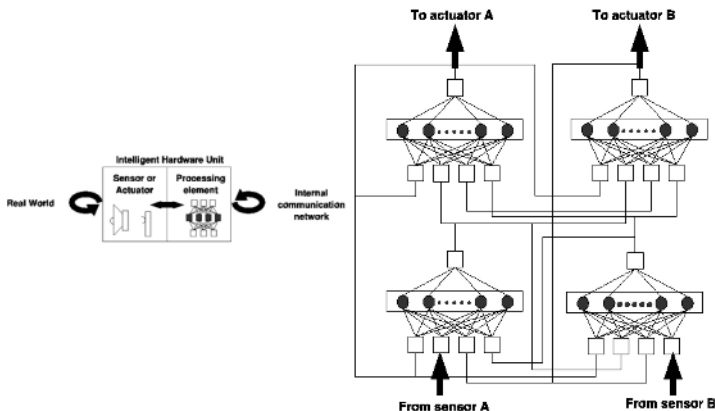
## 2 Architecture Definition

The architecture is based on several uniform modules, composed of neural networks, where each module is in charge of one element of the robot. Through the use of a neuro-evolutionary algorithm, modules learn how to cooperate between them and how to control its associated element, allowing the whole robot to accomplish the task at hands.

### 2.1 Hardware

Using Fodor's definition of module, we define the Intelligent Hardware Unit (IHU) as a module created around a physical device (sensor or actuator). Every IHU is composed by a sensor or by an actuator and a micro-controller implementing an artificial neural network (ANN) that process the information of its associated device (received sensor information for sensors, commands sent to the actuator for actuators). It is said that the ANN is in charge of its sensor/actuator. This means that is the neural net the one that decides which commands must be sent to the actuator, or how must it be interpreted a value received from a sensor, and under which circumstances. All IHUs are interconnected to each other in order to be aware of what the other IHUs are doing. So in some sense, the net is also in charge of deciding what to say to the other elements as well as to interpret what the others are saying. The structure of a IHU can be seen in the following figure, together with a neural controller for a simple system with two sensors and two actuators.

It should be stated that when put several IHU together on a control task, each element has its own particular vision of the situation because each one is



**Fig. 1.** Schematics of an IHU, and the connection schema of four ANNs from four IHUs controlling a simple robot

in charge of its own sensor or actuator. This leads to a situation where each unit knows what the others are doing but needs to select an action for its controller or sensor output, and based on its knowledge of the global situation and that of its particular device, decides what the next action will be.

Even though in the original definition a microprocessor was required for any IHU element, on the experiments presented here it has been simulated the existence of the micro-controllers linked to each device by allocating some processing time in the robot central processor for each IHU, since it was not physically possible to have one dedicated micro-controller for each IHU, neither in the simulations, nor in the real robot tests. It will be assumed that the results are not very different from the original idea.

## 2.2 Neuro-evolutionary Algorithm

To teach the networks the coordination required for the task a neuro-evolutionary approach has been selected. For the Co-evolution of the different networks and due to the necessity of evolving different ANNs for different roles on a common task, a co-evolutionary algorithm is required. By using such kind of algorithm it is possible to teach to the networks how they must cooperate to achieve a common goal, when every network has its own an different vision of the whole system.

The algorithm selected to evolve the nets is the ESP (Enforced Sub-Populations) [10][11], which has been proved to produce good results on distributed controllers [12]. This algorithm is also in principle free of bias for a special task. It is a general algorithm which produces good results in systems where several parts must interact to obtain the general view of the situation.

The chromosome codes in a direct way the weights of the network connections. A chromosome is generated for each IHU's network, and all are evolved at the same time over the same situation.

Usually in ESP, neurons in the hidden layer are also connected to themselves and to the other neurons of the same layer. This information is also coded in the chromosome. However, in the robot presented in this work and for the sake of simplicity, hidden neurons neither have been connected to themselves nor to the other neurons of the same layer.

### 3 Implementation

In order to validate the architecture explained in previous section, it was performed an implementation on a complex robot. Based on previous results where the architecture was tested on a very simple simulated robot [13], the architecture was later tested for the control of a complex real and simulated robot with several sensors and actuators while performing a task.

#### 3.1 Validation with a Complex Robot

For a validation of the architecture on real robots the Aibo robot was selected. This is a complex robot with several degrees of freedom and multiple sensors and actuators that requires a good coordination between them to achieve any simple task. The aim for this stage was to see if the architecture was capable of controlling such a complex robot in the realization of some simple tasks and a more complicated one. First task was to keep an up position; second was to stand up from a lying position, and third task was to generate a gait behaviour.

For the realization of those experiments a simulator and a real robot were used. The simulator selected was the Webots program by Cyberbotics, which allowed the easy evolution of the controller without having to use the real robot [14]. This introduced easiness in the experiments since the simulator had the possibility of letting the system running alone doing training. Also damaging the robot and problems with batteries were avoided by using the simulator. Once the controller was generated and tested on the simulator, a direct connection was established to the real robot, that allowed the test of the resulting neural controller on it. At this stage of the Webots simulator development (version 4.0.28beta) only direct connection of the simulator to the real robot is available, since the *cross-compilation of controller for real Aibo* feature was not working yet for the ERS-7 model. Direct connection means that the controller runs on the computer (under Webots) but controls the real robot via wireless connection.

For the control of the robot the following sensors and actuators were taken into account:

- Actuators: four leg upper joints (J1), four leg middle joints (J2), four leg lower joints (J3). These are all motors that move Aibo’s joints and determine its position on space.
- Sensors: four leg upper joints, four leg middle joints, four leg lower joints, four legs paw sensors. These are the sensors that indicate the state of the joint motors (actuators). Last four paw sensors indicate the state of the feet

paws. Those are switched on when the feet touches the ground or off when not touching. Furthermore, three accelerometer sensors (X,Y and Z) were used for determining Aibo's position.

This gives a total of 31 sensors and actuators, making the required number of ANNs to 31. All nets have the same number of inputs (31), outputs (1) and hidden units (8) for all the experiments. For the nets of the controllers, the inputs are connected to the outputs of all the other nets including itself. For the sensor networks are connected in the same way except that the entry from itself comes from the real sensor instead. For actuators, nets output indicate the kind of value that needs to be sent to the joint. For sensors, the output indicates the sensor value that is reported to the rest of IHU (including itself).

Values obtained from sensors were quantified, allowing a precision of one degree. This quantification generates a kind of dumping mechanism that prevents undesirable oscillations and never ending training. Network outputs of sensors were quantised in the same way. Network outputs of the actuators were quantized to provide only three possible values: move joint for 0.05 radians up, move joint for 0.05 radians down, or not move the joint.

Other required parameters for the evolution that were kept the same in all experiments are represented on table 1, and were selected based on previous experiences.

**Table 1.** Parameters used during neuro-evolution

| Parameter Name         | Parameter Value |
|------------------------|-----------------|
| Subpopulations         | 8               |
| Size of subpopulations | 40              |
| Mutation rate          | 0.4             |
| Stagnation             | 20              |

**First Test: Keep Standing Up Position.** The architecture was first tested to keep the robot on a standing up position . The robot was set up on an initial standing up position on the space and let free to act using its control networks. The goal was to keep up as high as possible with as less joint movements as possible. This seems a simple experiment but it is not because networks are directly connected to actuators and they keep on sending movement commands to the actuators all the time. This implies that joints will keep on moving unless the not-move-joint command is decided by their associated nets. This continuous movement of joints could lead to strange robot positions and eventually make it fall. In this task then, the robot must learn how to achieve a stable high position and keep it until the end of the evaluation time. For this purpose, the following fitness function was defined:

$$fitness = finalHeight / numberOfMovements$$



which evaluates the robot for its final height in opposition to the number of movement commands sent to the joints. Evaluation time for each phenotype was time steps.

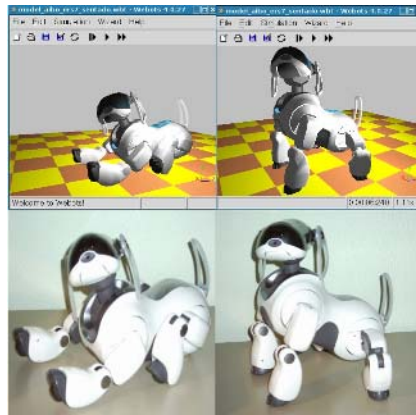
After 13 generations the robot learned how not to fall down and how to keep a good structure to stay stable, but still too many movements were realized. After generation 28, movements were reduced to a reasonable value while keeping the standing up position in both simulator and real robot.

**Second Test: Stand Up.** In this case the robot was required to change from a laying down position to a standing up one. For this purpose, the same architecture as in the previous experiment was used, been the only difference the fitness function and the time of evaluation of each phenotype, that in this case was of time steps.

The fitness function was defined as follows:

$$fitness = paws * finalHeight$$

Results showed that the robot was able to change its position to the desired one (see figure 2) after 77 generations in both simulator and real robot.



**Fig. 2.** Figures showing initial laying down position and final stand up obtained position of Aibo in both simulation and real robot

**Third Test: Walking.** This is by far the most difficult of the tests performed, and by hence, it was the one that gave more problems.

When designing controllers for robot walkers there are two main approaches: the first one uses a walking algorithm designed by hand by an engineer which indicates at any step the position of every joint of the robot [15][16][17]. Usually this approach has a set of parameters to be tuned, and they can be done of several ways (by hand, by try and modify, by genetic algorithms, etc...). The

resulting gait is one perfectly engineered that behaves exactly in the way the algorithm says.

The second approach consists of using CPGs. CPGs are non-linear oscillators made of neural nets. CPGs are coupled between them to generate the control signal for the actuators taking or not into account sensors information. The architecture developed here is very similar to that of CPGs, since it implements a group of neurons connected to the actuators, but it implements more connectivity and receives information from all sensors.

In this case, the architecture was implemented using recurrent neural nets instead of simple feedforward ones, and the fitness function was developed to conditionate the robot to acquire a determined gait style, based on the observance of Sony's walking style.

$$fitness = height * distance$$

Rule 1: fitness = 0 if J1 joints out of (-0.698,0.261) radians

Rule 2: fitness = 0 if J3 joints out of (0.349,1.658) radians

Rule 3: fitness = 0 if less than three paws down

Rule 4: fitness = 0 if after one front paw up does not follow a back paw up

For this experiment, an incremental evolutionary approach was taken, where the robot was let to neuro-evolve the controller for a few generations with rule 1 activated. After a determined number of evaluation steps rule 2 was activated, and so happens with rule 3 and 4. After applying four rules for several generations a basic walking pattern arise.

## 4 Discussion

Results presented here show that the neural architecture proposed is able to control a complex robot, and generate the required coordination mechanisms between sensors and actuators to perform complex tasks like walking or standing up. Coordination is learned during training phase and then used to solve the task during test phase with a completely neural based controller.

Experiments were performed in a general way, it is, there was no special consideration neither for the robot being controlled nor for the task being developed. Having implemented the architecture in such a way, it could be suggested that the architecture would be capable of the generation of any sensory-motor coordination required for any other task on any other robot, been the only limitations those imposed by evolutionary robotics. We admit thought, that the architecture has only been tested on two simulation robots (see author's previous work [13]) and only one real robot. Nevertheless, some research is been performed at present by the authors to apply this architecture to different types of robots on different tasks.

We think this work is a step closer to the final goal of obtaining a completely neurally controlled robot were more deliberative tasks could be performed.

## Acknowledgements

Authors would like to thank Olivier Michel for their support and big help on Webots issues, and Auke Ijspeert for insightful suggestions and comments on walking robots.

## References

1. Nolfi, S., Floreano, D.: *Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines*. The MIT Press (2000)
2. Floreano, D., Mondada, F.: Evolutionary neurocontrollers for autonomous mobile robots. *Neural Networks* **11** (1998) 1461–1478
3. Hiroshi Kimura, S.A., Sakurama, K.: Realization of dynamic walking and running of the quadruped using neural oscillator. *Autonomous Robots* **7** (1999) 247–258
4. Hiroshi Kimura, Y.F., Konaga, K.: Adaptive dynamic walking of a quadruped robot by using neural system model. *Advanced Robot* **15** (2001) 859–876
5. Collins, J., Richmond, S.: Hard-wired central pattern generators for quadrupedal locomotion. *Biological Cybernetics* **71** (1994) 375–385
6. Reeve, R.: Generating walking behaviours in legged robots. PhD thesis, University of Edinburgh (1999)
7. Billard, A., Ijspeert, A.J.: Biologically inspired neural controllers for a motor control in a quadruped robot. *Proceedings of the International Joint Conference on Neural Network* (2000)
8. Minsky, M.: *The Society of Mind*. Touchtone Books (1988)
9. Fodor, J.: *The modularity of mind*. Cambridge, MA: MIT Press (1983)
10. Gmez, F., Miikkulainen, R.: Solving non-markovian control tasks with neuroevolution. In: *Proceedings of the IJCAI99*. (1999)
11. Gomez, F., Miikkulainen, R.: Incremental evolution of complex general behavior. Technical Report AI96-248, University of Texas (1996)
12. Yong, H., Miikkulainen, R.: Cooperative coevolution of multiagent systems. Technical Report AI01-287, Department of computer sciences, University of Texas (2001)
13. Tllez, R., Angulo, C.: Evolving cooperation of simple agents for the control of an autonomous robot. In: *Proceedings of the 5th IFAC Symposium on Intelligent Autonomous Vehicles*. (2004)
14. Michel, O.: Webots: Professional mobile robot simulation. *Journal of Advanced Robotics Systems* **1** (2004) 39–42
15. Hornby, G.S., Fujita, M., Takamura, S., Yamamoto, T., Hanagata, O.: Autonomous evolution of gaits with the sony quadruped robot. In Banzhaf, W., Daida, J., Eiben, A.E., Garzon, M.H., Honavar, V., Jakiela, M., Smith, R.E., eds.: *Proceedings of the Genetic and Evolutionary Computation Conference*. Volume 2., Orlando, Florida, USA, Morgan Kaufmann (1999) 1297—1304
16. Hornby, G., Takamura, S., Hanagata, O., Fujita, M., Pollack, J.B.: Evolution of controllers from a high-level simulator to a high DOF robot. In: *ICES*. (2000) 80–89
17. Rfer, T.: Evolutionary gait-optimization using a fitness function based on proprioception. *Lecture Notes in Artificial Intelligence* (2005)

# Complex Behaviours Through Modulation in Autonomous Robot Control

J.A. Becerra, F. Bellas, J. Santos, and R.J. Duro

Grupo de Sistemas Autónomos, Universidade da Coruña, Spain  
{ronin, fran, santos, Richard}@udc.es

**Abstract.** Combining previous experience and knowledge to contemplate tasks of increasing complexity is one of the most interesting problems in autonomous robotics. Here we present an ANN based modular architecture that uses the concept of modulation to increase the possibilities of reusing previously obtained modules. A first approximation to the modulation of the actuators was tested in a previous paper where we showed how it was useful to obtain more complex behaviours than those obtained using only activation / inhibition. In this paper we extend the concept to sensor modulation, which enables the architecture to easily modify the required behaviour for a module, we show how both types of modulation can be used at the same time and how the activation / inhibition can be seen as a particular case of modulation. Some examples in a real robot illustrate the capabilities of the whole architecture.

## 1 Introduction

When designing a behaviour based architecture in a bottom up fashion [4][2] two main problems have to be considered [12]: the decomposition of a robot control system into subparts is not evident and the interactions between modules grow exponentially as the complexity of the system grows. So, it's necessary to establish a scalable method to perform the decomposition and to obtain the interactions between modules.

In hierarchical architectures, the global behaviour is decomposed into lower level behaviours that will be implemented in particular controllers and organized in a hierarchy where a controller can be activated or inhibited by other higher level controller. The behaviours can be obtained individually and then the interconnection between them established. It is also possible to reuse the behaviours. The problem that arises is that the decomposition is not clear in every case, as it implies a specific knowledge of what sub-behaviours must be employed. This, in general, implies a greater participation of the designer in the process of obtaining a global controller. Examples are Brooks [4] and Colombetti, Dorigo and Borghi [5].

In distributed architectures, all the controllers compete at the same level for control of the actuators each instant of time, leading to less participation of the designer. They also preserve the level of behaviour reuse. However, as a drawback, they induce a higher level of difficulty when obtaining the relations between modules. Distributed architectures are exemplified by Arbib [1] and Arkin [2].

An alternative classification of control architectures is to differentiate between competitive and cooperative coordination. In the former only one process writes its output to the actuators each moment of time. Examples are [4] and [13]. In cooperative coordination, the outputs of two or more processes that control the same actuators are combined into a single output to be sent to the actuators, usually through summation with different degrees, as the case of [1][2].

We propose modulation as a way to bridge the gap between hierarchical architectures, usually competitive, and distributed architectures, usually cooperative. In that line, we show in [3] that actuator modulation permits to obtain a graceful and continuous transition between behaviours that were obtained for other problems and which, through the adaptation of their outputs, can be set to perform together more complex tasks. In this paper, we will see that sensor modulation, i.e. the modification of how the modules “see the world”, is useful to adapt controllers in order to value the environment in a different way. We will also present the whole control architecture where both types of modulation can coexist and where the activation / inhibition can be seen as a particular case of modulation.

The concept of modulation has not been employed often in the literature on autonomous robot behaviour controllers. Among the authors that have done some work in this line, we can cite Husbands and col. [8] and Ishiguro and col. [9], which use modulation within a single ANN. From the point of view of multi-module architectures, some work has been carried out by Meyer and col. [11], who apply modulation in the neural nodes of the modules, and Duro et al. [6], who establish a base for a direct modulator-modulated structure in which a network or networks modulate the behaviour of other networks influencing their connection weights.

Taking into account that the concept of modulating the operation of an ANN implies adapting the values that result from it through some external action, from a systematic point of view, modulating structures can be classified into three groups: structures that act on the input, structures that act on the network nodes or synapses and structures that act on the outputs of the network. Most of the work presented can be framed on the first two classes. The second one, which may be referred as functional modulation, is the most powerful of the three as it permits actually modifying the function the network is implementing. However, it presents many implementation problems, especially due to exponential growth of the modulating networks as the modulated networks grow. In addition, the modulated networks or, in general, behaviour modules have to contemplate their modulation by others, which precludes the use of legacy controllers. On the other hand, input or output modulation take the modulated controller as a black box and the modulator just acts over its inputs or outputs in order to force the desired behaviour or its adaptation.

## 2 Constructing Control Architectures with Modulation

In [7] we established a hierarchical, modular ANN based, control architecture for autonomous robots where both modules and connections are obtained incrementally in an evolutionary process. In this control architecture, if the behaviour is very simple, it's obtained as a whole in an evolutionary process. Otherwise, the designer tries to

identify sub-behaviours that may be useful to generate the global behaviour required. There is no problem if useless behaviours are included in this preliminary set. Behaviours, after being obtained separately, are provided to the system as modules and an evolutionary process will obtain a higher level module that select those lower level behaviours from the initial set (activating just one in each instant of time) that are useful and will ignore the rest. This process can be repeated as many times as desired, leading to a controller with a hierarchical structure. To prevent the problem of the designer having to completely determinate all the necessary lower level behaviours, a higher level behaviour may be coevolved with a lower level behaviour.

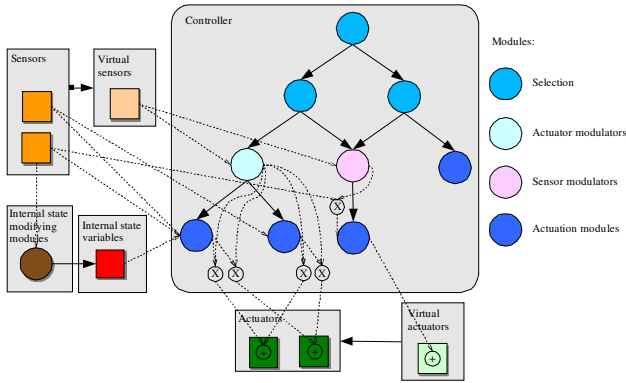
The control flows in this architecture from the top (root node) to the bottom (leaf nodes). Every module is an ANN where its inputs are sensors, but only the lower level modules can take control of actuators through their outputs. Other modules take part in the decision process of what lower level module has to be activated in the end. The same module may appear several times in a controller, as it may be needed for different sub-behaviours. In that case, even if we only plot it once for the sake of simplicity, the module will be replicated in the hierarchy as many times as necessary.

As indicated before, in [3] we presented an initial output modulation strategy. This was done by defining a new type of high level module, called actuator modulator, that, instead of activating only one lower level module, it activates every lower level module connected to it but modifying the way the lowest modules take control of actuators. As more than one module is allowed to take control of the actuators, all of these controllers cooperate by adding their outputs to each actuator.

The work presented in this paper develops the concept and provides new ways to modulate behaviours. We are now adding a new high level module, called sensor modulator. This module activates also every lower level module connected to it, in this case modifying how these lower level modules perceive the environment through their sensors.

Both modulators are also ANNs and are obtained through evolution. Because we are not modifying anything inside the modulated ANN, any ANN can be modulated. These are the main definitions in the architecture we present:

- A module X is an ancestor of a module Y if there is a path from X to Y.
- X is a descendant of Y if there is a path from Y to X.
- X is a direct descendant if there is a path of length 1 from Y to X.
- X will be called a Root node (denoted as R) if it has no ancestors.
- X is an actuator node (A) if its outputs establish values for the actuators.
- X is a selector node (S) if its output selects one of its direct descendants as the branch to follow, shortcircuiting the others.
- X is an actuator modulating node (AM) if its outputs modify (multiplying with a value between 0 and 2) the outputs of its descendant nodes of type A. The modulations propagate through the controller hierarchy until they reach the actuator nodes in such a way that if between R and A there is more than one AM that modulates one output of A, the resulting modulating value will be the product of the individual modulations in the path. Assuming that AM wishes to modulate the values of  $n$  actuators, its number of outputs must necessarily be  $n \cdot \text{number of direct descen}$



**Fig. 1.** Example of a controller with all the elements of the architecture

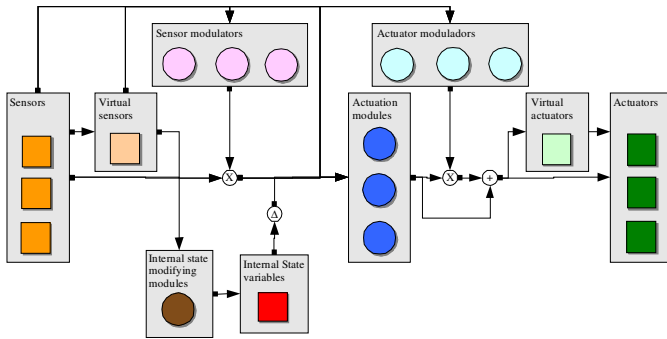
dants, as the modulation propagated to each descendant is usually different. When more than one node A provides values for the same actuator, the actuator receives the sum of these values. An AM does not necessarily modulate all the actuators over which the set of nodes acts, just any subset of them.

- X is a sensor modulating node (SM) if its outputs modify (multiplying with a value between 0 and 2) the inputs of its descendant nodes. The modulations propagate through the controller hierarchy until they reach the actuator nodes in such a way that if between R and Y there is more than one SM that modulates one input of Y, the resulting modulating value will be the product of the individual modulations in the path. Assuming that SM wishes to modulate the values of  $n$  sensors, its number of outputs must necessarily be  $n \cdot \text{number of direct descendants}$ , as the modulation propagated to each descendant is usually different. An SM does not necessarily modulate all the sensors over which the set of nodes acts, just any subset of them.

The use of actuator modulators leads to a continuous range of behaviours for the transitions between those determined by the individual controllers. This is due to the fact that actuators values can now be a linear combination of those produced by every low level module.

The sensor modulators permit changing how a module reacts under a given input pattern transforming it in a different one. This way, it is very easy to make changes in that reaction for already learned input patterns. We will show an example of this later.

In addition to increasing the architecture’s possibilities, modulation results in a very interesting secondary effect: there can be more than a sub-tree being executed simultaneously in the controller. So, the architecture is not really different from a distributed architecture where modules are categorized into different groups, because actuator modulators can be put together in the same level and sensor modulators can be set aside of the hierarchy and attached to the appropriate inputs where necessary. This is true even if we have selectors, because it is evident that a selector can be replaced by an actuator modulator that modulates every actuator with “1” in the descendant



**Fig. 2.** Alternative representation for the architecture (arrows represent blocks of connections)

tree activated and with “0” in the others<sup>1</sup>. In the figure 2 we display an alternative general representation for a controller taking this equivalence into account.

To a human designer it may be easier to understand a controller using the first representation, as the equivalent controller with the second representation has more connections, but the existence of this second representation is important because it shows that the architecture can be taken, if desired, as a distributed one where the modules are just classified into very general groups. The second representation is also useful to implement the controller in hardware or in a parallel environment because all of the modules’ outputs can be calculated simultaneously.

Just to provide a flavour of what can be achieved with modulation, in what follows we will present two experiments. The controllers were obtained using SEVEN [7], an environment that integrates the simulation of different robot platforms with the evolutionary processes, and later tested on the real robot (a Pioneer 2-DX).

### 3 Sensor Modulation

In this example we will see how sensor modulation may be useful to quickly adapt behaviours to new requisites that imply a different reaction when dealing with already learnt input patterns.

We start from a controller obtained previously to achieve wall following, at a given distance from the wall, with the Pioneer 2-DX. The new desired behaviour is for the robot to follow walls at a distance that depends on light intensity (measured with a couple of added light sensors): the more intensity, the closer it must follow the walls.

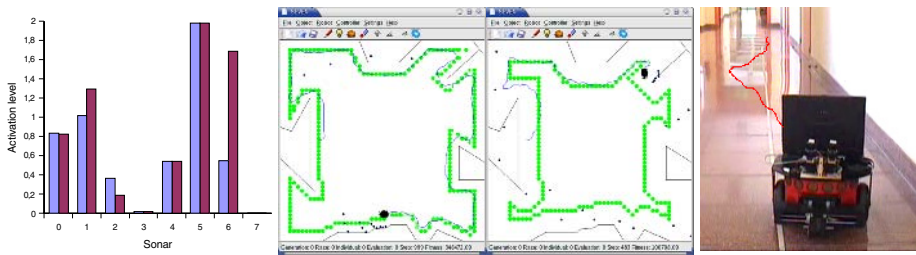
The controller only uses two modules: the original wall following and the sensor modulator that is being evolved. The ANN being evolved has 2 inputs (light sensors), a hidden layer with 11 neurons and 8 outputs (each one modulates one of the 8 inputs of the wall following module, corresponding to frontal sonar sensors). The connections between neurons are gaussian synopsis. The evolutionary algorithm used was a macroevolutionary algorithm [10] with 800 individuals, 100 evaluations and 1000

<sup>1</sup> This replacement can be carried out if we replicate common modules in different branches, as we indicated before.



steps of life in each evaluation per individual, and 100 generations. Two different light conditions were present in the evolution and the robot had to follow the wall at different distances. We could be tempted to think the problem is not complicated, but we have to realize that the solution is not as easy as modulating every input value by the same factor, because the same change in input values does not necessarily lead to the same change in the activation of connected neurons.

In the figure 3 we display the sonar activation values for both light conditions and the obtained behaviour. The evolution has led to a sensor modulator that achieves its task by playing mainly with two sonar sensors (1 and 6). As an additional advantage, it prunes the wall following module making sonars 3 and 7 unnecessary.



**Fig. 3.** Sensor modulation. Left: sonar activation level for two different light conditions (left bar = more light, right bar= less light). Center: behaviour in the simulator. Right: behaviour in the real robot

## 4 Sensor and Actuator Modulation Combined

The next example illustrates the use of both types of modulation together and the kind of rich behaviour that can be obtained with only two low level modules. The behaviour we want to obtain can be described as follows. The robot has to catch a prey avoiding a predator. Prey and predator are also mobile objects. The robot gains energy as it goes near the prey and loses energy as it goes near the predator. There are two internal values that affect also the robot energy: the energetic cost associated to each step and the anxiety level of the robot that makes it loses energy (the higher the anxiety level, the higher energy loss). The objective is to catch the prey maximizing energy. That means that if the energetic cost in each step or the anxiety is very high the robot will go straight to the prey, without worrying about getting near the predator. In other case, the robot will try to catch the prey avoiding predator by a long distance.

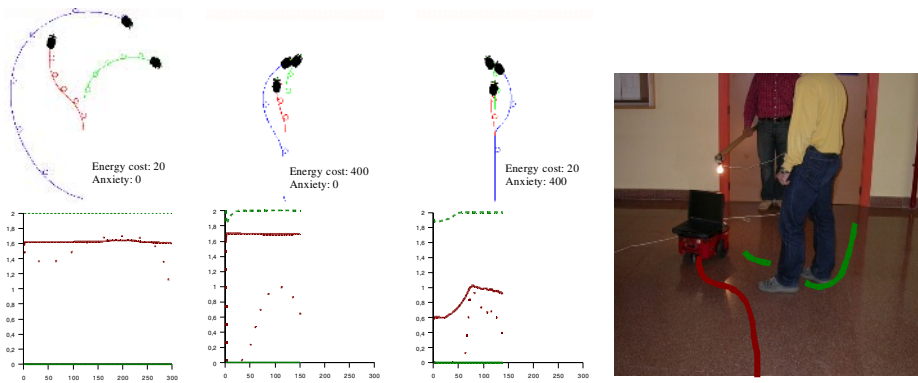
The controller has 4 modules: 2 low level modules (go straight to the prey and escape straight from the predator), 1 actuator modulator (which modulates outputs of the 2 low level modules depending on distance to prey and predator) and 1 sensor modulator (which modulates inputs of the actuator modulator depending on anxiety).

The ANN of the sensor modulator has 2 inputs, 2 hidden layers with 6 neurons each one and 2 outputs. The ANN of the actuator modulator has 3 inputs (two of them modulated by the sensor modulator), 2 hidden layers with 6 neurons each one and 2 outputs. To simplify the problem, the only output of the low level modules is the angular velocity (which will be between  $-40^\circ/\text{sec}$  and  $40^\circ/\text{sec}$ ). Linear velocity is set to a

fixed value (200 mm/sec). Predator and prey follow a predefined motion with a linear velocity of 70 mm/sec and a maximum angular velocity of  $2^\circ/\text{sec}$ . The prey escapes from the robot and the predator tries to get between the prey and the robot.

The construction of the controller without the sensor modulator was explained in [3]. So, we only obtained the new sensor modulator. The evolutionary algorithm employed was a macroevolutionary algorithm with a population of 100 individuals which evolved 100 generations. Each individual was evaluated 200 times and it had 300 steps of life each time.

Figure 4 shows some examples. As the actuator modulator is the same as the one in [3] the behaviour is similar: it only plays with the “escape from predator” module and it maintains the modulation of the “go to prey” module constant. The sensor modulator evolves in a similar way and it maintains the modulation of the sensor that indicates the distance to the prey constant and only modulates the sensor that indicates the distance to the predator. That “constant modulation” applied to the “distance to prey” sensor is in fact a “0”, so the ANN corresponding to the actuator modulator can be pruned. This makes sense, the actuator modulator activates by default the “go to prey” module and only when it is near the predator activates as necessary the “escape from predator” module. It is also really interesting that the sensor modulator only changes the actuator modulator behaviour when anxiety is higher than the amount of energy lost in each step, otherwise the actuator modulator already behaves correctly.



**Fig. 4.** “Catch prey and avoid predator” behaviour. Graphics represent modulation levels. Dashed lines = actuator modulator outputs, continuous lines = sensor modulator outputs

## 5 Conclusions

This paper deals with a complete modulation based architecture for robot controllers. The inclusion of sensor modulators allows the architecture to easily modify behaviours changing how they consider input patterns. The reaction of a module to a given input pattern can be very complex, so it is easier to use a sensor modulator to adapt that reaction than to obtain other module from scratch. This modulation can be used together with actuator modulation and the other components of the architecture to obtain many different behaviours from a small set of controllers in a structured manner.

An interesting result from behaviour design point of view is that, due to the nature of its components, the architecture can be seen as hierarchical or distributed with modules grouped into a few different classes. Thus, it presents advantages of both types of architectures.

Both types of modulation individually and the complete architecture have been tested on real robots in real operating conditions and the behaviours obtained correspond with the desired ones.

## Acknowledgements

This work was funded by Xunta de Galicia under project PGIDIT03TIC16601PR.

## References

1. Arbib, M.A. (1992), "Schema Theory", in *The Encyclopedia of Artificial Intelligence*, 2nd ed., S. Shapiro (Ed.), Wiley, New York, N.Y., pp. 1427-43.
2. Arkin, R.C. (1998), *Behavior Based Robotics*, MIT Press, Cambridge, MA.
3. Becerra, J. A., Santos J. and Duro R. J. (2003), "Multimodule Artificial Neural Network Architectures for Autonomous Robot Control Through Behavior Modulation", *Lecture Notes in Computer Science*, vol. 2687, pp. 169-176.
4. Brooks, R.A. (1986), *Achieving Artificial Intelligence through Building Robots*, A.I. Memo 898, MIT, AI Lab.
5. Colombetti, M., Dorigo, M., and Borghi, G. (1996), "Behavior Analysis and Training - A Methodology for Behavior Engineering", *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 26, No. 3, pp. 365-380.
6. Duro, R.J., Santos, J., and Gómez, A. (1995), "Synaptic Modulation Based Artificial Neural Networks", *From Natural to Artificial Neural Computation*, J. Mira, F. Sandoval (Eds.), LNCS, Vol. 930, Springer-Verlag, Berlin, pp. 31-36.
7. Duro, R. J., Santos, J., Becerra, J. A. (2000), "Evolving ANN Controllers for Smart Mobile Robots", *Future Directions for Intelligent Systems and Information Sciences*, Springer-Verlag, pp. 34-64.
8. Husbands, P., Smith, T., O'Shea, M., Jakobi, N., Anderson, J., Philippides, A. (1998), "Brains, Gases and Robots", *Perspectives Neural Comp.*, V. 2, pp. 51-63.
9. Ishiguro, A., Otsu, K., Fujii, A., and Uchikawa, Y. (2000), "Evolving an Adaptive Controller for a Legged-Robot with Dynamically-Rearranging Neural Networks", *Proc. Supp. 6th Int. Conf. on Simulation of Adaptive Behavior*, J-A Meyer, A. Berthoz, D. Floreano, H.L. Roitblat, S.W. Wilson (Eds.), pp. 235-244.
10. Marín, J. and Solé, R.V. (1999), "Macroevolutionary Algorithms: A New Optimization Method on Fitness Landscapes", *IEEE Transactions on Evolutionary Computation*, Vol. 3, No. 4, pp. 272-286.
11. Meyer, J-A., Doncieux, S., Filliat, D., and Guillot, A., (2002), "Evolutionary Approaches to Neural Control of Rolling, Walking, Swimming and Flying Animats or Robots", R.J. Duro, J. Santos and M. Graña (Eds.), *Biologically Inspired Robot Behavior Engineering*, Vol. 109, Physica-Verlag, pp. 1-43.
12. Harvey, I. (1996), "Artificial Evolution and Real Robots", *Proceedings of International Symposium on Artificial Life and Robotics (AROB)*, Masanori Sugisaka (Ed.), Beppu, Japan, pp. 138-141.
13. Urzelai, J., Floreano, D., Dorigo, M., and Colombetti, M. (1998), "Incremental Robot Shaping", *Connection Science Journal*, Vol. 10, No. 384, pp. 341-360.

# Explorative Data Analysis Based on Self-organizing Maps and Automatic Map Analysis

Marc Franzmeier, Ulf Witkowski, and Ulrich Rückert

Heinz Nixdorf Institute, University of Paderborn,  
Paderborn, Germany  
{franzmeier, witkowski, rueckert}@hni.upb.de

**Abstract.** In the field of explorative data analysis self-organizing maps have been used successfully for a lot of applications. In our case, we apply the self-organizing map for the analysis of semiconductor fabrication data by training recorded high dimensional data sets. Usually, the training result is displayed by using appropriate visualization techniques and the results are evaluated manually. Especially for large data sets an automated post-processing of the training result is essential. In this paper an automatic training result analysis based on specific image processing is introduced. Dependencies between components maps are calculated by structure overlapping analysis based on the segmentation of component maps. This novel method has been integrated into the data analysis software DanI, that simulates self-organizing maps for data analysis with several pre-processing and post-processing capabilities.

## 1 Introduction

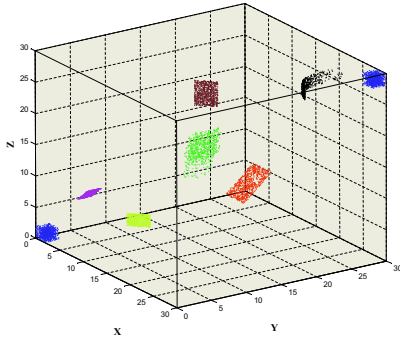
The range of applications where self-organizing maps (SOMs)[3] are used is extremely wide. Applications like machine vision, optical character recognition and, signal processing are typical for SOMs [7]. In cooperation with an industrial partner we use self-organizing maps for the analysis of semiconductor fabrication data. Aim in this field of explorative data analysis is the optimization of the fabrication process in terms of quality and yield [5]. During the data collection phase various parameters are captured from the fabrication process. One type of data is physical parameters like film thickness, implantation dose, equipment settings etc. that are collected during the fabrication process. The second type of data is a measurement result of electrical test structures like voltages, currents and derived data. These different types of data are summarized up to high dimensional data sets containing several ten thousands of vectors with several hundred of vector components. Due to the large number of vectors a manual analysis is not possible. By applying the data set to a self-organizing map during a training phase a mapping of the high dimensional data set to a mostly two dimensional map is achieved [3][5] and the training result may be interpreted visually in a qualitative way. But in general an automatic post-processing of the trained SOM

is desirable to get quantitative dependencies between the various components in the data set and to reduce time for analysis. Therefore, an automated analysis of the trained SOM has been developed. In this paper, we introduce an automated structure overlapping analysis, that performs an image segmentation on the visualizations of self-organizing maps like component maps or U-Matrices [8]. An image segmentation algorithm based on the Discrete Curve Evolution Algorithm [4] extracts the structure of the input data after being mapped to the SOM. Size, shape and position of the extracted structures of all component maps of the SOM are compared to identify similarities. The resulting level of overlapping gives information about dependencies in the data set. The SOM algorithm as well as the analysis of the trained data are implemented in the data analysis tool DanI. We demonstrate the capabilities of the new approach on an artificial test data set containing three-dimensional vectors. In section 2 the data analysis using SOMs and the segmentation of self-organizing maps is discussed. The three dimensional artificial data set is introduced in section 3. Furthermore, the applied image processing and the Discrete Curve Evolution algorithm as well as its necessary extension for an automated segmentation of histograms is explained. In section 4 the data analysis software DanI is introduced. The discussed topics are summarized in a conclusion in section 5.

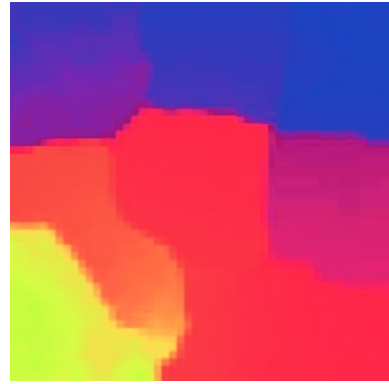
## 2 Segmentation of Self-organizing Maps

The unsupervised learning algorithm of self-organizing maps is used in the field of explorative data analysis by the authors, because of its nonlinear projection of a high dimensional input space onto a two-dimensional map of neurons. After projection to the two-dimensional map of neurons the training result has to be examined. Different visualization techniques, like U-Matrix and component maps, have become popular. All this visualizations are capable to visualize specific properties of the map representing the input data. What they are suffering from is the necessary manual analysis of the visualized structure of the data. Manual analysis of high dimensional data is not only time consuming, but also a very complex task. Additionally, manual data analysis often results in qualitative results only. When quantitative results are demanded more reliable methods are necessary. In order to allow a quantitative comparison between different component maps it is necessary to identify the structure of the maps using appropriate algorithms. Component maps and U-Matrix are two-dimensional graphical representations of the trained SOM. These two-dimensional visualization techniques can be treated as ordinary images. Therefore, image processing methods can be applied to analyze them.

The principle procedure shown in the following is illustrated by using an artificial three-dimensional data set (see figure 1). Different types of clusters are included in the data set. Linear and non linear dependencies between two components of the data are included as well as randomized dependencies. In total the data set contains 6000 three-dimensional vectors. This data set is trained on a two-dimensional, rectangular SOM of 50 by 50 neurons.



**Fig. 1.** Three-dimensional data set containing 6000 data points



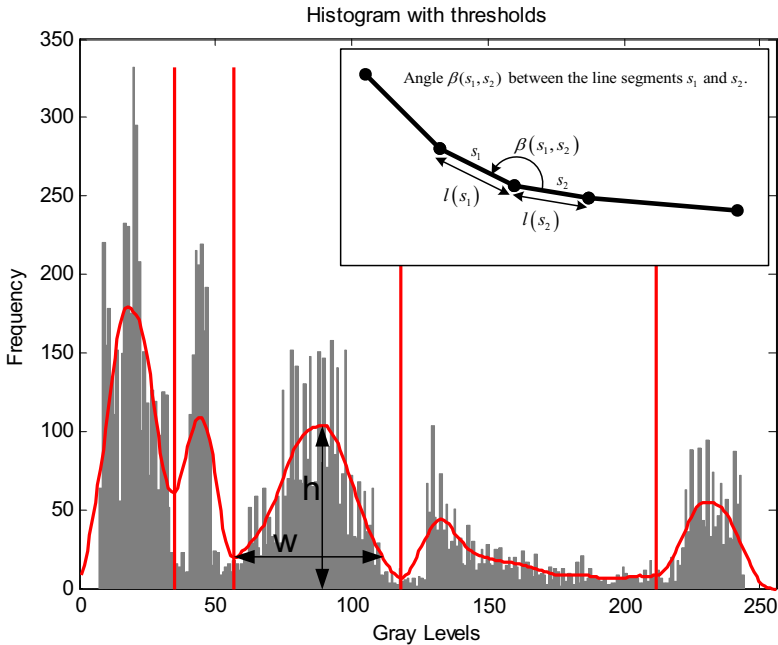
**Fig. 2.** Component map of the y-component of the data set

### 3 Image Processing

For the segmentation of component maps and U-Matrices we handle the visualizations as ordinary digital images, where one neuron is represented by one pixel in the image. Histogram based segmentation is used for further image processing. Firstly, the coloured image is converted into a grey level image with 256 levels and the grey level histogram is calculated. In this histogram separated accumulations of grey levels with high rates correspond to areas with similar colours in the original image. Replacing these accumulations of grey levels by one grey level in the original image, leads to a segmented image. The task is to find the appropriate separators in the histogram. In real world images the scene is often illuminated in a way that only one fixed separator is necessary. Generally, in the field of data mining more then one separator is needed as well as their position of them is application specific. Because of dealing with unknown data the separation of grey level accumulations in histograms is a non trivial process. In case of analyzing data from a semiconductor production process the captured data is afflicted with noise. Measurement impreciseness and small differences in the properties of the fabricated device under test are responsible for this noise. Noisy histograms include a large number of entries, which are not forming the shape of the histogram. Before further computation the histogram is smoothed, because the qualitative shape of the histogram is needed only. Suitable smoothing algorithms are moving averages, local smoothing, low pass filters, or other methods that reduce noise in the histogram [6]. Especially local maxima should be reduced by the smoothing algorithm, because this reduces the expense to identify the important peaks in the histogram later on. The number of points to represent the histogram can then be reduced without destroying the shape of the histogram. Latecki et al. have introduced a shape simplifying algorithm, the Discrete Curve Evolution [1],[4]. Treating the histogram as a polygon of connected line segments a relevance function  $K(s_1, s_2)$  can be defined as:

$$K(s_1, s_2) = \frac{\beta(s_1, s_2) \cdot l(s_1) \cdot l(s_2)}{l(s_1) + l(s_2)} \quad (1)$$

In (1) the relevance  $K(s_1, s_2)$  of the kink between two adjacent line segments  $s_1$  and  $s_2$  is calculated.  $\beta(s_1, s_2)$  is the enclosed angle between the segments,  $l(s)$  is the length of the line segment  $s$  (cf. figure 3). The relevance  $K(s_1, s_2)$  is calculated for every kink in the histogram. The kink with minimal relevance is deleted. Then the algorithm starts from the beginning. After several iterations the minimal points representing the shape of the histogram are left. Using a fixed number of iterations is not suitable in the field of explorative data analysis. Removing too many kinks may delete the shape of the histogram. Removing to few kings leads to a non minimal histogram representation, which aggravates the search of the separators in the histogram more than necessary. The Discrete Curve Evolution introduced by Latecki has been enhanced with a dynamic stop criterion by the authors. In figure 3 the grey level histogram corresponding to



**Fig. 3.** Grey level histogram based on fig. 2 and definition of  $\beta$  (small figure) in (1)

figure 2 is shown. The histogram is smoothed, because the qualitative shape (see black line) is necessary for further computation, not the quantitative one. The dynamic stop criterion stops the Discrete Curve Evolution Algorithm one step before a kink is deleted, that is forming the major shape of the histogram. As a result, the histogram is reduced to the main shape forming peaks. This leads directly to the question what is an important peak in a histogram? In figure

3 a human observer would identify five peaks that characterize the histogram. A relevance measurement is needed, to specify the importance of a peak in a histogram. It is defined as:

$$S_{Peak} = w \cdot h = Width \cdot Height \quad (2)$$

The size of a peak  $S_{Peak}$  is defined heuristically as the product of the opening width  $w$  of a peak and its height  $h$  (cf. figure 3). After each iteration of the Discrete Curve algorithm the size of all peaks in the histogram is calculated and averaged in  $\bar{S}_{Peak}$ . The relation between the peaks is defined by a heuristic weight function  $W_t$ , that is initialized with zero.

$$S_{Peak} < 0.25 \cdot \bar{S}_{Peak} : W_{t+1} = 0.5 \cdot (W_t - 1) \quad (3)$$

$$S_{Peak} \geq 0.25 \cdot \bar{S}_{Peak} : W_{t+1} = 0.5 \cdot (W_t + 1) \quad (4)$$

$S_{Peak}$  is the average peak size, which is calculated after each iteration. The weight of a peak is increasing, if its current size is longer than twenty five percent of the average peak size, otherwise the weight is decreased. If shape of the histogram is reduced to the minimal number of peaks, the separators are placed at the local minima between two peaks. If there is a plain line between two maxima, the middle of the line is chosen as a position for the separator. In figure 3 vertical separators show the sectioning of the histogram, calculated with the method described above. Based on the separation in the histogram the image of the component map can be segmented. We use standard segmentation algorithms for separating the structures of the original image into binary sub images. I. e. we select one section of the histogram and set all pixels in a binary copy of the original image to one, if they have the corresponding colour range of the section in the histogram. After storing this binary image, we proceed with the next section in the histogram. Out of these binary images stored for every section in the histogram the structures can be extracted very easily. Because one section in the histogram may correspond to more than one structure in the binary image, we use conventional region growing in the binary image to extract the structures. Figures 4 and 5 are showing the segmented y and z component maps of the SOM for the data set depicted in figure 1.

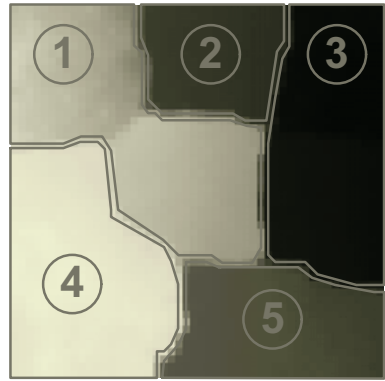
### 3.1 Structure Overlapping Analysis

If the extraction of the structures of all components in a data set has been performed, an overlapping analysis of the structures can be generated automatically. Size, shape and position are compared between the structures of all component maps. Minimal size of the structures as well as the minimal similarity of the structures are parameters of the process, that can be adjusted manually. The minimal size is used as a relevance measure. It avoids comparing large (important) structures with smaller ones (less important). More important than the size is the similarity between two structures. The similarity is defined as the percentage of overlapping between two structures of the analyzed components.

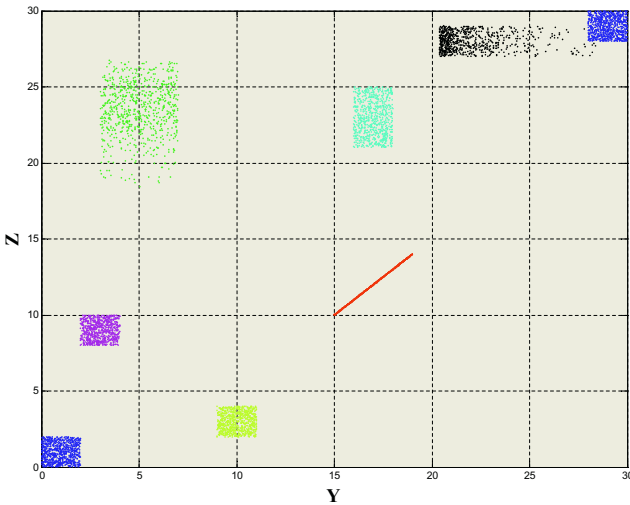




**Fig. 4.** Segmented y-component map of the SOM trained with data set from fig. 1



**Fig. 5.** Segmented z-component map of the SOM trained with data set from fig. 1



**Fig. 6.** yz-projection of data set depicted in figure 1

The capabilities of the new method will be demonstrated in the following. Based on the artificial data set of figure 1 the relations between the y and z-component (cf. figure 6) of the data set will be discussed. Therefore, the new method has been applied on the y and z-component of the artificial data set. Figures 4 and 5 are showing the segmentation of the component maps, mentioned above. Obviously both component maps are separated in five structures. Brighter grey levels in the figures correspond to larger values in the input data. E. g. combining the structures 2 and 5 in figure 4 leads to a structure that is very similar to structure 4 in figure 5. Structure 2 has a size of 52.4% and structure 5 has a size of 46.2% of structure 4 in figure 5. Although there is only one structure in z-

component map, there are two structures in y-component map. This corresponds to the two topmost right clusters in figure 6. The z-values of these clusters can not be separated, because they partially share the same interval. In contrast, these clusters are separated in the y-component and in z-component map of the SOM. Although the new segmentation method and the structure extraction haven been applied to component maps, both can be applied on the U-Matrix. As typical U-Matrices have clearly separated structures and large areas with nearly the same colours, standard image segmentation using one separator in the histogram produces satisfying results.

### 4 Implementation of Self-organizing Maps

In cooperation with an industrial partner the software DanI (**D**ata **a**nalysis for the **I**ntegrated circuit fabrication) has been developed (figure 7). It integrates pre-processing of recorded data, training of the SOM, visualization, recall and analysis functionality into one powerful tool. Input interfaces like MS Excel allow comfortable import of data. If necessary, data pre-processing algorithms can be applied on the data in order to process missing, faulty or alphanumeric data. The training of the data itself can be performed on the local processor of

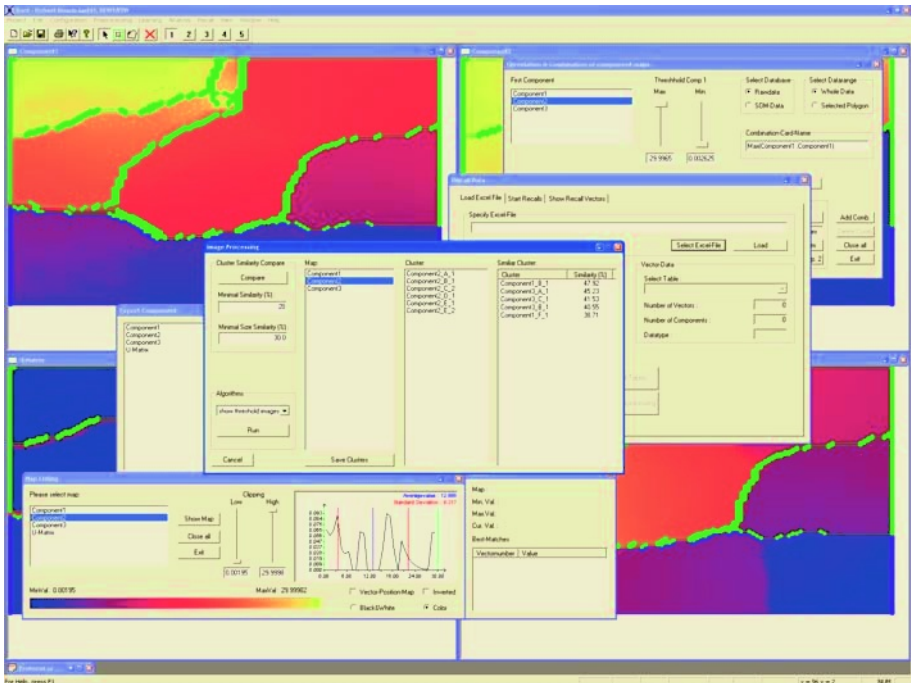


Fig. 7. DanI-Software: Data analysis for the IC-fabrication

a personal computer or speeded up using an accelerator [2], developed by the authors. After training, the SOM can be visualized using component maps and U-Matrix. Additionally, correlation calculations can be executed on the SOM training results as well as on the raw data. For the analysis of high dimensional data the structure overlapping analysis has been implemented. The structures of a key component are compared with the clusters of all other components automatically. As a result the coverage of the structures is presented to the user in percentage rate. The extracted structures can also be used for correlation computation.

## 5 Conclusion

It has been shown that SOM based explorative data analysis can be automated by applying an image segmentation on visualization of the SOM. A multi threshold histogram based segmentation has been developed to identify and extract the structures of component maps. The structure overlapping analysis, presented in this paper, analyses the extracted structures and returns quantitative dependencies between vectors of the input data. Structure overlapping analysis as well as the automated image segmentation have been combined to an automated data analysis method that has been integrated into the data analysis software DanI. The features of the software, like pre-processing, training, visualization, recall and analysis, have been presented. The principles of the automated analysis have been demonstrated using an artificial data set.

## Acknowledgements

This work was partly developed in the course of the Collaborative Research Center 614 - Self-Optimizing Concepts and Structures in Mechanical Engineering - University of Paderborn, and was funded by the Deutsche Forschungsgemeinschaft.

## References

1. T. Barkowsky, L. J. Latecki, and K. Richter. Schematizing maps: Simplification of geographic shape by discrete curve evolution. *LN in Computer Science*, 1849, 2000.
2. M. Franzmeier, C. Pohl, M. Porrmann, and U. Rückert. Hardware accelerated data analysis. In *Proceedings of the 4th Int. Conf. on Parallel Computing in Electrical Engineering (PARELEC 2004)*, Dresden, Germany, 7 - 10 September 2004.
3. T. Kohonen. Speedup of som computation. In *Triennial Report 1997 - 1999*, pages 46–49. Neural Networks Research Center, Laboratory of Computer and Information Science, Helsinki University of Technology, 1999.
4. L.J. Latecki, R. Venugopal, M. Sobeland, and S. Horvath. Tree-structured partitioning based on splitting histograms of distances. In *IEEE Int. Conference on Data Mining (ICDM'03)*, Melburne, USA, November 2003.

5. S. Rüping and J. Müller. Analysis of IC fabrication processes using self-organizing maps. In *ICANN99, Proc. of the Ninth International Conference on Artificial Neural Networks*, pages 631–636, Edinburgh, 1999.
6. S. W. Smith. *The Scientist and Engineer's Guide to Digital Signal Processing*. California Technical Publishing, San Diego, California, 1997.
7. V. Tryba, K. Marks, U. Rückert, and K. Goser. Selbstorganisierende Karten als lernende klassifizierende Speicher. In *ITG-Fachbericht*, number 102, pages 407–419, Berlin, Germany, 1988. VDE-Verlag.
8. A. Ultsch and H. P. Siemon. Exploratory data analysis: Using Kohonen networks on transputers. Technical Report 329, Univ. of Dortmund, Germany, Dec. 1989.

# A Novel Optimization of Profile HMM by a Hybrid Genetic Algorithm

Lifang Liu, Hongwei Huo, and Baoshu Wang

Xidian University, School of Computer Science and Technology,  
Xi'an, P.R.China 710071

lfliu@mail.xidian.edu.cn, liulifang2008@hotmail.com

**Abstract.** Profile Hidden Markov Models (Profile HMM) are well suited to modelling multiple alignment and are widely used in molecular biology. Usually, heuristic algorithms such as Baum-Welch are used to estimate the model parameters. However, Baum-Welch has a tendency to stagnate on local optima. A more involved approach is to use some form of stochastic search algorithm that ‘bumps’ Baum-Welch off from local maxima. In this paper, a hybrid genetic algorithm is presented for training profile HMM (hybrid GA-HMM training) and producing multiple sequence alignment from groups of unaligned protein sequences. The quality of the alignments produced by hybrid GA-HMM training is compared to that by the other Profile HMM training methods. The experimental results prove very competitive with and even better than the other tested profile HMM training methods. Analysis of the behavior of the algorithm sheds light on possible improvement.

## 1 Introduction

Hidden Markov Models (HMMs) are a class of probabilistic models that are generally applicable to time series or linear sequences. HMMs have been most widely applied to recognizing words in digitized sequences of the acoustics of human speech [1]. HMMs were introduced into computational biology in the late 1980s, and have been a preferred choice of method when solving problems. Profile HMM is a particular type of HMM well suited to modelling multiple alignment. Profile HMM can be used to detect potential membership in a family by obtaining significant matches of a sequence to the profile HMM, to give an alignment of a sequence to the family or more precisely to add it into the multiple sequence alignment of the family, and to classify protein families [2].

The standard algorithm for training HMM from initially unaligned example sequences are hill-climbing algorithms, such as gradient descent [3] or Baum-Welch expectation maximization [2], which are iterative algorithms in which the likelihood (or the posterior probability) increases in each iteration. A serious problem with any hill-climbing optimization technique is that it often ends up in a local maximum. To deal with this problem, noise injection during HMM re-estimation or stochastic methods can be used to avoid local maxima. Noise

injection was introduced in Krogh et al.[2], an alternative and very elegant simulated annealing algorithm was described in Eddy [4]. Genetic algorithm was also used to estimate HMM. Kwong et al. [5] applied an GA to estimate the left-right models, and used small population sizes (30 individuals) and low mutation (0.0001) and crossover (0.01) probabilities. Thomsen [6] presented an evolutionary algorithm capable of evolving both the topology and the model parameters of HMM, and the applicability of the method was exemplified on a secondary structure prediction problem. In this paper, we present a hybrid genetic algorithm for training profile HMM, and producing multiple alignments to test the applicability of the introduced method.

## 2 Profile HMM

Krogh et al.[2] introduced a profile HMM architecture that was well suited for representing profiles of multiple sequence alignment. The basic model architecture is shown in Fig.1.

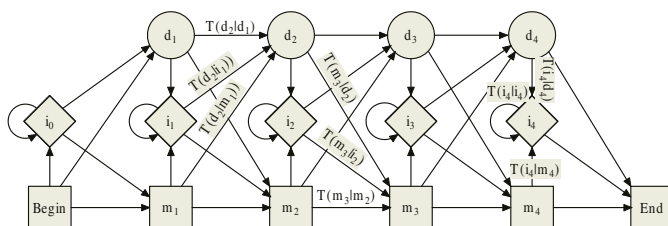


Fig. 1. Profile hidden Markov model

A HMM describes a series of observations by a ‘hidden’ stochastic process - a Markov process. Each position in the model has three states. A state shown as a rectangular box is a match state that models the distribution of letters in the corresponding column of an alignment. Diamond-shaped states model insertions of letters between two alignment positions, and circular states model a deletion, corresponding to a gap in an alignment. States of neighboring positions are connected, as shown by lines. For each of these lines there is an associated *transition probability*, which is the probability of going from one state to the other. For additional details on profile HMM, see [2].

Sequence alignment under probabilistic models can be scored using the log-odds ratio [7]. The log-odds score asserts whether or not the probability that the sequence  $s$  was generated by the model  $m$  is larger than the probability that the sequence was generated by the null model  $\phi$ :

$$score(s) = \log_z \frac{P(s|m)}{P(s|\phi)} \tag{1}$$

where the logarithm can be to any base  $z$ , most often to base 2, in which case the score is reported in bits. The higher the score, the better. The log-odds score has been used by HMMER [4], HMMER is a freely distributable implementation of profile HMM software for protein sequence analysis. In this paper, we also use the log-odd score to score sequence alignment.

### 3 Methods and Algorithm

#### 3.1 Representation

For a profile HMM of length  $M$ , here  $M$  is the number of match states in the profile HMM rather than the total number of states, which is  $3M+3$  for the profile HMM architecture in Fig.1. There are  $M$  positions in the model, each position has three states(match, delete, and insert). Each of these  $M$  match states ( $m_k, k = 1...M$ ) can generate a letter  $x$  from the 20-letter amino acid alphabet according to a distribution  $P(x|m_k), k = 1...M$ , and all emission probabilities in each match state must satisfy the following equation:

$$\sum_{i=1}^{20} P(x_i|m_k) = 1 \quad (2)$$

The notation  $P(\cdot|m_k), k = 1...M$  means that each of the match states  $m_k, k = 1...M$ , have distinct distributions, where ‘.’ indicates any possible letters from the 20-letter amino acid alphabet. There are a total of  $M+1$  insert states which generate amino acids in exactly the same way as the match states, but use probability distributions  $P(\cdot|i_k), k = 0...M$ . From each state, there are three possible transitions to other states, as shown in Fig.1. The transition probability from state  $q$  to state  $r$  is called  $T(r|q)$ , and for each state, transition probabilities must satisfy the following equation:

$$\sum_{i=1}^3 T(r_i|q) = 1 \quad (3)$$

where  $r_i$  denotes any possible next state. The notations  $T(\cdot|d_k), k = 1...M, T(\cdot|i_k), k = 0...M, T(\cdot|m_k), k = 0...M$  mean that each of the states (delete states  $d_k, k = 1...M$ , insert states  $i_k, k = 0...M$ , match states  $m_k, k = 0...M$ ) have distinct transition distributions, where ‘.’ indicates any possible next state. At position  $M$ , the three states have no transition to delete state. For a model of length  $M$ , the total number of *state transition probabilities* is  $9M+3$ , and the total number of *symbol emission probabilities* is  $40M+20$ , if the profile HMM describes a family of proteins.

The goal of hybrid GA-HMM training is to find a model, i.e. a proper model length  $M$  and probability parameters, that accurately describes a family of proteins by assigning large probabilities to sequences in that family. In hybrid GA-HMM training, the profile HMM is encoded into a string of real numbers that acts as a chromosome (see Fig.2).

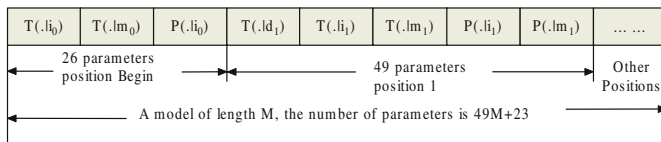


Fig. 2. Genetic representation of profile HMM

### 3.2 Population Initialization

The population of initial parent profile HMMs was generated as follows: i)The initial number of model length  $M$  was set to the average length of all the training sequences  $M_{avg}$ . ii)The probabilities were randomly initialized (between 0.0 and 1.0) and normalized to satisfy Equations (2) and (3).

### 3.3 Crossover and Mutation

The crossover operators used are the standard *arithmetic crossover*, *one-point crossover*, and *two-point crossover*, which take two parent profile HMMs and create two offsprings. For parent profile HMMs of different length, crossover operator acts on the domain  $[1, M_{min}]$ , as shown in Fig.3, and  $M_{min} = \min(M_{parent1}, M_{parent2})$ . After crossover, the probabilities in each chromosome must be normalized to satisfy Equations (2) and (3).

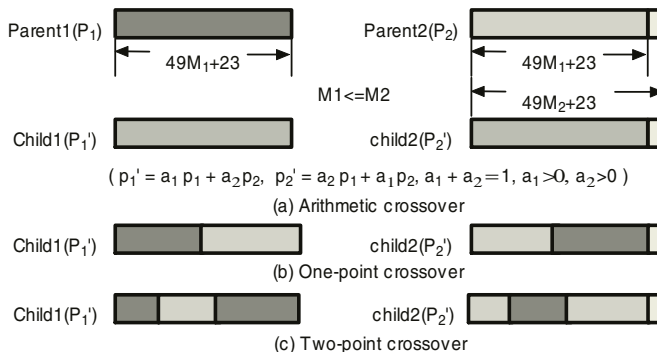


Fig. 3. Crossover operators

Three mutation operators are used in this paper:

The *insertPosition* operator adds a new position into the model. This position is selected randomly between 1 and  $M$ , and the 49 probabilities associated with the new states are assigned randomly and normalized. After this change, the model length become  $M+1$ .

The *deletePosition* operator deletes a position from the model. This position is chosen randomly between 1 and  $M$ , and must ensure a valid profile HMM. After this change, the model length become  $M-1$ . Together with the *insertPosition* operator, the model architecture is re-estimated, much like model surgery (for additional details on model surgery, see[2,7]).



The **modifyProbabilities** operator selects probabilities from parent chromosome randomly and the selected probabilities are replaced by random generated real numbers; after this change, the affected probabilities are normalized.

### 3.4 Fitness Evaluation

The average log-odds score  $l_m$  of the profile HMM solution  $m$  generates the training observation sequences  $O_1, \dots, O_n$  must be calculated as references for the fitness evaluation:

$$l_m = \left( \sum_{i=1}^n \log_2 \frac{P(O_i|m)}{P(O_i|\phi)} \right) / n \quad (4)$$

where  $O_i, (i = 1 \dots n)$  is a protein sequence of  $L_i$  letters, and denoted  $x_1 \dots x_{L_i}$ . The log likelihood score  $\log_2 P(O_i|m)$  is calculated by forward procedure[1], and  $\log_2 P(O_i|\phi) = \sum_{i=1}^{L_i} \log_2 P(x_i|\phi)$ . Since the probabilities of solution  $m$  were randomly initialized,  $l_m$  may be negative, so to guarantee positive, the fitness value  $f_m$  of solution  $m$  will be calculated as follows:

$$f_m = \begin{cases} l_m + C_{min} & \text{when } l_m + C_{min} > 0, \\ 0 & \text{Otherwise.} \end{cases} \quad (5)$$

where  $C_{min}$  is the log-odds score obtained by Baum-Welch training.

### 3.5 The Hybrid Genetic Algorithm

Since GA has a global search ability and heuristics have a local search ability, their hybridization will possibly form a more powerful search.

In our hybrid GA, Baum-Welch algorithm is employed. Crossover is applied with probability  $P_c$  using one of the three mentioned crossover operators (the choice is made randomly with equal probability for all three crossover operators). Mutation is applied with probability  $P'_m$  using one of the three mentioned mutation operators (the choice is made randomly with equal probability for all three mutation operators).  $P'_m$  is decreased with current generation  $g$  increased using the formula:  $P'_m = P_m(1.0 - g/g_{max})$ , where  $g_{max}$  is the predefined maximal generation. An offspring replaces the parent only if it is fitter. The offspring is refined using Baum-Welch algorithm [1,7], which re-estimated all the parameters ( $\hat{P}(x|m_k)$  and  $\hat{T}(r|q)$ ). To avoid over-fitting, Dirichlet mixture priors for regularization [2,6,7] is used, for more details on Dirichlet mixture priors, see [8]. The algorithm terminates either after a predefined number of generations  $g_{max}$  or after a predefined number of non-improved generations  $g_{unimproved}$ . All the steps can be summarized by the following pseudo-code:

#### Procedure Hybrid GA-HMM

**BEGIN**

1. Initialize population.
  - Set models' length to  $M_{avg}$ ;
  - $P(\cdot|\cdot) = random()$ ;
  - $T(\cdot|\cdot) = random()$ ;

```

2. Evaluate.
 Calculate f_{m_i} , $i=1..population\ size$;
WHILE ($g_{current} \leq g_{max}$ and $g_{current\ unimproved} \leq g_{unimproved}$)
BEGIN
 WHILE ($p_{current\ size} \leq population\ size$)
 BEGIN
 3. Tournament selection.
 4. Crossover.
 IF($random() \leq P_c$)
 Select one of the three crossover operators with equal probability;
 Apply the selected crossover operator on parents;
 5. Mutation.
 IF($random() \leq P'_m$)
 Select one of the three mutation operators with equal probability;
 Apply the selected mutation operator on offsprings;
 6. Apply Baum-Welch on offsprings.
 Calculate $\hat{P}(\cdot|\cdot)$, $\hat{T}(\cdot|\cdot)$;
 7. Apply regularization on offsprings.
 8. Evaluate.
 END
END
END

```

## 4 Experimental Results

The modeling was first tested on the globins, a large family of heme-containing proteins involved in the storage and transport of oxygen that have different oligomeric states and overall architecture. The globin protein sequences used for the training set were taken from the file `globin50.fa` of HMMER1.8.4, which contains 50 randomly selected unaligned globin sequences. We validated this model from the alignments it produced by the Viterbi algorithm [1,7], and compared to the performance of the `hmmt` program of HMMER1.8.4, `hmmt` can build an HMM from initially unaligned training sequences, and allows a choice of approaches, simulated annealing (SA), the Viterbi approximation of the Baum-Welch (Viterbi), and the full Baum-Welch (BW) implementations. The alignment accuracy was assessed by the sum-of-pairs score (SPS) and the column score (CS), SPS indicates the ratio of pairs correctly aligned while CS shows the ratio of columns correctly aligned, for how to calculate them see Thompson et al. [9]. The alignment of seven representative globins from Bashford et al. [10] was used as the reference alignment, and the `BAliscore` program [9] was used to calculate SPS and CS score. In the test, we used Dirichlet mixture priors [8] for regularization, a ten-component mixture Dirichlet prior for match emissions, and single component Dirichlet priors for insert emissions and transitions, the data was taken from the file `BrownHaussler.pri` of HMMER1.8.4. All the tests were performed on the same precondition.

We used the following parameters in the hybrid GA-HMM training: *population size*=10,  $P_c=0.7$ ,  $P_m=0.1$ ,  $g_{max}=100$ ,  $g_{unimproved}=10$ . For Viterbi HMM

**Table 1.** Results obtained from the experiments

| training methods |               | SPS/CS      | log-odds score |
|------------------|---------------|-------------|----------------|
| Viterbi          | don't surgery | 0.195/0.000 | 250.45         |
|                  | surgery       | 0.283/0.000 | 257.32         |
| BW               | don't surgery | 0.598/0.190 | 292.32         |
|                  | min           | 0.523/0.190 | 286.85         |
| SA               | max           | 0.738/0.474 | 304.24         |
|                  | avg           | 0.595/0.319 | 300.05         |
|                  | min           | 0.535/0.207 | 291.62         |
| hybrid GA        | max           | 0.841/0.716 | 328.94         |
|                  | avg           | 0.671/0.412 | 306.13         |

**Table 2.** Results obtained from the experiments when *populationsize* = 10, 20, 30, 40, 50 respectively

| population size | 10          | 20          | 30          | 40          | 50          |
|-----------------|-------------|-------------|-------------|-------------|-------------|
| min             | 0.647/0.353 | 0.580/0.267 | 0.702/0.474 | 0.718/0.440 | 0.606/0.276 |
| SPS/CS          |             |             |             |             |             |
| max             | 0.766/0.586 | 0.810/0.578 | 0.872/0.741 | 0.858/0.716 | 0.899/0.819 |
| avg             | 0.689/0.416 | 0.713/0.456 | 0.797/0.603 | 0.777/0.571 | 0.779/0.584 |
| min             | 307.09      | 310.86      | 316.24      | 314.06      | 312.15      |
| log-odds        |             |             |             |             |             |
| max             | 314.27      | 318.49      | 324.72      | 324.35      | 322.26      |
| avg             | 308.53      | 314.23      | 318.12      | 320.44      | 322.17      |

**Table 3.** Average SPS/CS obtained from the experiments, for hybrid GA, *populationsize* = 10

| training methods | BW          | SA          | hybrid GA   |
|------------------|-------------|-------------|-------------|
| reference2       | 0.659/0.101 | 0.662/0.112 | 0.738/0.163 |
| reference3       | 0.382/0.038 | 0.439/0.094 | 0.491/0.148 |

training, model surgery [7] was applied, but for SA HMM training, model surgery did not seem to improve the accuracy in this test, so these results were not listed. We ran SA and hybrid GA 50 times respectively. The minimal, the maximal, and the average (avg) scores were listed in the Table 1.

In order to improve the performance of hybrid GA, set *populationsize* equal to 10,20,30,40,50 respectively. In each instance ran the program ten times. The minimal, the maximal, and the average (avg) scores were listed in the Table 2.

The modeling was also tested on the sequences from BALiBASE reference2 and reference3, for additional details on BALiBASE, see [11]. Each alignment in reference2 and reference3 tested only once, and the average score is given. Table 3 shows the results of the experiments.

We built a profile HMM from the alignment of seven representative globins from Bashford et al.[10] using the MAP construction algorithm[6], and encoded this model as an individual for the population initialization, the performance of hybrid GA is improved, here *populationsize* = 10, the result is (average score of 10 runs): log-odds score=315.37, SPS=0.718, CS=0.485. For complicated HMMs,

the parameter space may be complex, with many spurious local optima that can trap a training algorithm. Large collections of protein structural alignments are now available, so build the model from a multiple sequence alignment, and further refining it using one of these methods (BW,SA,GA etc) can greatly improve the accuracy of the model.

## 5 Conclusion

In our study, Baum-Welch algorithm was employed, other heuristic algorithms for training HMMs can also be employed. We use *insertPosition* operator and *deletePosition* operator slightly change the model architecture, that much like model surgery, and the change is stochastic. In our future work, architecture learning operators will be added.

## References

1. Rabiner,L.R.(1989) Tutorial on hidden Markov models and selected applications in speech recognition. *Proc. IEEE*, 77,257-286.
2. Krogh,A.,Brown,M.,Mian,I.S.,Sjolander,K.and Haussler,D.(1994) Hidden Markov models in computational biology: Applications to protein modeling. *Journal of Molecular Biology*, 235, 1501-1531.
3. Baldi,P.,Chauvin,Y.,Hunkapiler,T.,and McClure,M.A.(1994) Hidden Markov modes of biological primary sequence information. *Proc. Natl. Acad. Sci. USA*, 91, 1059-1063.
4. Eddy,S.(1995) Multiple alignment using hidden Markov models. *Proc. Int. Conf. on Intelligent Systems for Molecular Biology*, 114-120, Cambridge, England:AAAI/MIT Press.
5. Kwong,S.,Chau,C.W.(1997) Analysis of parallel genetic algorithms on HMM based speech recognition system *IEEE Transactions on Consumer Electronics*, 43, 1229-1233.
6. Thomsen, R.(2002) Evolving the Topology of Hidden Markov Models using Evolutionary Algorithms. *Proceedings of the Seventh International Conference on Parallel Problem Solving from Nature - PPSN VII*, 861-870, 2002.
7. Durbin,R.,Eddy,S.R.,Krogh,A. and Mitchison,G.J.(1998) Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids. Cambridge University Press, Cambridge, UK.
8. Brown,M.P,Hughey,R.,Krogh,A.,Mian,I.S.,Sjölander,K., and Haussler,D.(1993) Using Dirichlet mixture priors to derive hidden Markov models for protein families. *Proc. of First Int. Conf. on Intelligent Systems for Molecular Biology*, 47-55, Menlo Park, CA:AAAI/MIT Press.
9. Thompson,J.D.,Plewniak,F. and Poch,O.(1999) A comprehensive comparison of multiple sequence alignment programs. *Nucleic Acids Research*, 27(13), 2682-2690.
10. Bashford,D.,Chothia,C. and Lesk,A.M. (1987) Determinants of a protein fold: unique features of the globin amino acid sequence. *Journal of Molecular Biology*, 196, 199-216.
11. Thompson,J.D.,Plewniak,F. and Poch,O.(1999) BALiBASE: a benchmark alignment database for the evaluation of multiple alignment programs. *Bioinformatics*, 15, 87-88.

# Heuristic Search over a Ranking for Feature Selection

Roberto Ruiz, José C. Riquelme, and Jesús S. Aguilar-Ruiz

Department of Computer Science, University of Seville, Spain  
{rruiz, riquelme, aguilar}@lsi.us.es

**Abstract.** In this work, we suggest a new feature selection technique that lets us use the wrapper approach for finding a well suited feature set for distinguishing experiment classes in high dimensional data sets. Our method is based on the relevance and redundancy idea, in the sense that a ranked-feature is chosen if additional information is gained by adding it. This heuristic leads to considerably better accuracy results, in comparison to the full set, and other representative feature selection algorithms in twelve well-known data sets, coupled with notable dimensionality reduction.

## 1 Introduction

In recent years, there has been an explosion in the rate of acquisition of data in several domains. A typical data set may contain thousands of features. Theoretically, having more features should give us more discriminating power. However, this can cause several problems: increase computational complexity and cost; too many redundant or irrelevant features; and estimation degradation in the classification error.

Most of the feature selection algorithms approach the task as a search problem, where each state in the search specifies a distinct subset of the possible attributes [1]. The search procedure is combined with a criterion in order to evaluate the merit of each candidate subset of attributes. There are a lot of possible combinations between each procedure search and each attribute measure [2]. Feature selection is grouped in two ways according to the attribute evaluation measure: depending on the type (filter or wrapper techniques) or on the way that features are evaluated (individual or subset evaluation). The filter model relies on general characteristics of the data to evaluate and select feature subsets without involving any mining algorithm. The wrapper model requires one predetermined mining algorithm and uses its performance as the evaluation criterion. It searches for features better suited to the mining algorithm, aiming to improve mining performance, but it also is more computationally expensive [3] than filter model. Feature ranking (FR), also called feature weighting [1, 4], assesses individual features and assigns them weights according to their degrees of relevance, while the feature subset selection (FSS) evaluates the goodness of each found feature subset. (Unusually, some search strategies in combination

with subset evaluation can provide a ranked list). In the FR algorithms category, a subset of features is often selected from the top of a ranking list. This approach is efficient for high-dimensional data due to its linear time complexity in terms of dimensionality. In the FSS algorithms category, candidate feature subsets are generated based on a certain search strategy. Different algorithms address these issues distinctively. In [2], a great number of selection methods are categorized. We found different search strategies, namely exhaustive, heuristic and random search, combined with several types of measures to form different algorithms. The time complexity is exponential in terms of data dimensionality for exhaustive search and quadratic for heuristic search. The complexity can be linear to the number of iterations in a random search [5], but experiments show that in order to find best feature subset, the number of iterations required is usually at least quadratic to the number of features [6]. The most popular search methods in machine learning ([7, 8]) can not be applied to these data sets due to the large number of features. One of the few used search techniques in these domains is sequential forward [9, 10, 11] (also called hill-climbing or greedy).

The limitations of both approaches, ranking and subset selection, clearly suggest that we should pursue a hybrid model. Recently, a new framework for feature selection has been used, where several above-mentioned approaches are combined. The process of selection involves two phases due to the high number of attributes: Algorithms begin with a phase where attributes are individually evaluated, and provide a ranking according to a filter criterion. In the next step, a feature subset evaluator (filter or wrapper) is applied to a fixed number of attributes from the previous ranking (greater than a threshold value, or the first  $k$  features) following a search strategy. The method proposed by Xing et al. [12], the one proposed by Yu and Liu [9], and another by Guyon et al. [13] are among the most referenced works at present following this framework.

Our paper is organized as follows. In section 2, we present the concept of relevance and redundancy at the same time used in our wrapper approach. Algorithm is described in section 3. Experimental results are shown in Section 4, and the most interesting conclusions are summarized in section 5.

## 2 Wrapper Approach over Feature Ranking

Feature ranking makes use of a scoring function  $S(i)$  computed from the values  $x_{k,i}$  and  $y_k$  ( $k = 1, \dots, m$  examples and  $i = 1, \dots, n$  features). By convention, we assume that a high score is indicative of high relevance and that features are sorted in decreasing order of  $S(i)$ . We consider ranking criteria defined for individual features, independently of the context of others. In feature subset selection, it is a fact that two types of attributes are generally perceived as being unnecessary: attributes that are irrelevant to the target concept, and attributes that are redundant given other attributes. We now formally define incremental ranked (IR) usefulness in order to devise an approach to explicitly identify relevant features and do not take into account redundant features. In other words,

learning can be achieved more efficiently and effectively with just relevant and non-redundant features.

**Definition 1.** *Let  $R$  be a set of  $M$  features sorted in decreasing order of  $S(i)$ , given a sample of data  $D$ , a learning algorithm  $L$ , and a subset of selected features  $F$ , feature  $F_i$  is incrementally useful to  $L$  with respect to  $F$  if the accuracy of the hypothesis that  $L$  produces using the group of features  $\{F_i\} \cup F$  is better significantly (denoted by  $\succ$ ) than the accuracy achieved using just the subset of features  $F$ , in this case  $F_i$  is added to  $F$ . Note that the process starts from the first feature in  $R$ , and continues with the next ranked attribute.*

Wrapper subset evaluates attribute sets by using a learning scheme. Five cross validation is used to estimate the accuracy of the learning scheme for a set of features. We conduct Student's paired two-tailed t-test in order to evaluate the statistical significance (at 0.1 level) of the difference between the previous best subset and the candidate subset. This last definition allows us to select features from the ranking, but only those that increase the classification rate significantly. Although the size of the sample is small (5 cross validation), our search method use a t-test. We want to obtain an heuristic not to do an accurate population study. However, on the one hand it must be noted that it is an heuristic based on an objective criterion, to determine the statistical significance degree of difference between the accuracy of each subset. On the other hand, the confidence level has been relaxed from 0.05 to 0.1 due to the small size of the sample. Statistically significant differences at the  $p < 0.05$  significance level would not permitted us to add more features, because it would be difficult to obtain significant differences between the accuracy of each subset by the test. Obviously, if the confidence level is increased, more features can be selected, and vice versa. Then, the user can adjust the confidence level.

### 3 Algorithm

There are two phases in the algorithm shown in Figure 3: Firstly, the features are ranked according to some evaluation measure (line 1–4). In second place, we deal with the list of attributes once, crossing the ranking from the beginning to the last ranked feature (line 5–12).

Consider the situation depicted in Figure 2; an example of feature selection process by IR. It shows the attributes ranked according to some evaluation measure. We obtain the classification accuracy with the first feature in the list (f5). In the second step, we run the classifier with the first two features of the ranking (f5,f7), and a paired t-test is performed to determine the statistical significance degree of the differences. As it is lower than 0.1, f7 is not selected. The same occurs with the two next subsets (f5,f4 and f5,f3), but feature f1 is added, because the accuracy obtained is significantly better than that obtained with only f5, and so on. In short, the classifier is run nine times to select, or not, the ranked features (f5,f1,f2): once with only one feature, four times with two features, three with three features and once with four features. The same situation occurs in high-dimensional data.

---

Input: E training, U--measure, W--classifier  
Output: BestSubset

---

```

1 list l = {}
2 for each $F_i \in F$
3 $S(i) = \text{compute}(f_i, U)$
4 position F_i into l according to $S(i)$
5 BestClassification = 0
6 BestSubset = \emptyset
7 for each $F_i \in l$
8 TempSubset = BestSubset $\cup F_i$
9 TempClassification = WrapperClassification(TempSubset, W)
10 if (TempClassification > BestClassification)
11 BestSubset = TempSubset
12 BestClassification = TempClassification

```

---

Fig. 1. IR Algorithm

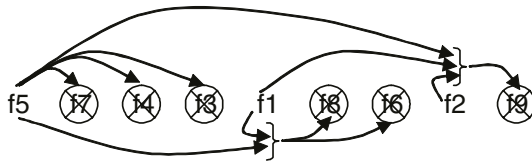


Fig. 2. Example of feature selection process by IR

## 4 Experiments and Results

The aim of this section is to evaluate our approach in terms of classification accuracy, degree of dimensionality and speed on selected features, in order to see how IR fare in situations where there are large numbers of features. The comparison was performed with two representative groups of high-dimensional data sets: Three data sets are selected from the UCI Repository<sup>1</sup>, and three selected from the NIPS 2003<sup>2</sup> feature selection benchmark. The main characteristic of these data sets is the great number of features. The full characteristics of all the data sets are summarized in Table 1. In order to compare the effectiveness of feature selection, attribute sets chosen by each technique were tested with two learning algorithms, a probabilistic (naive Bayes) and a decision tree learner (*c4.5*). These two algorithms were chosen because they represent two quite different approaches to learning.

As already mentioned, the proposed search is realized over a ranking of attributes, and any evaluation measure can be used for it. In the experiments, we use two criteria: one belongs to wrapper model and one to filter model. In the wrapper approach, denoted by  $IR_W$ , we order attributes according to their indi-

<sup>1</sup> <http://www.ics.uci.edu/mllearn/MLRepository.html>

<sup>2</sup> <http://clopinet.com/isabelle/Projects/NIPS2003/>



**Table 1.** Data sets

| Data                         | Acrr. | Feat. | Inst. | Classes |
|------------------------------|-------|-------|-------|---------|
| Musk <sup>(1)</sup>          | MK    | 166   | 6598  | 2       |
| Arrhythmia <sup>(1)</sup>    | AR    | 279   | 452   | 16      |
| Madelon <sup>(2)</sup>       | MA    | 500   | 2000  | 2       |
| Multi-feature <sup>(1)</sup> | MF    | 649   | 2000  | 10      |
| Arcene <sup>(2)</sup>        | AC    | 10000 | 100   | 2       |
| Dexter <sup>(2)</sup>        | DE    | 20000 | 300   | 2       |

**Table 2.** Accuracy of nb on selected features. The symbol ” + ” and ” – ” respectively identify statistically significant, at 0.1 level, wins or losses over  $IR_W$

| Data | Wrapper |     |        |     |        | Filter |            |       |              | Full  |       |       |       |
|------|---------|-----|--------|-----|--------|--------|------------|-------|--------------|-------|-------|-------|-------|
|      | $IR_W$  |     | $IR_F$ |     | $SF_W$ |        | $CFS_{SF}$ |       | $FOCUS_{SF}$ |       |       |       |       |
|      | Acc     | Att | Acc    | Att | Acc    | Att    | Acc        | Att   | Acc          |       | Att   |       |       |
| MK   | 84.59   | 1   | 84.59  | 1   | N/A    | 65.75  | –          | 10    | 83.37        | –     | 11    | 83.86 |       |
| AR   | 73.01   | 7   | 73.02  | 8   | 74.35  | 15     | 69.69      | –     | 25           | 69.03 | –     | 21    | 61.74 |
| MA   | 63.00   | 4   | 62.65  | 3   | 62.75  | 6      | 60.90      | 6     | 59.15        | –     | 15    | 58.40 |       |
| MF   | 97.30   | 15  | 97.85  | 19  | N/A    | 97.10  | 86         | 93.65 | –            | 7     | 93.35 | –     |       |
| AC   | 90.00   | 22  | 93.00  | 19  | 83.00  | 4      | N/A        | 60.00 | –            | 4     | 70.00 | –     |       |
| DE   | 88.67   | 14  | 88.00  | 15  | 84.67  | 11     | N/A        | 90.33 | 23           | 88.67 |       |       |       |

vidual predictive power, using as criterion the performance of the target classifier built with a single feature. In the filter approach, a ranking is provided using non-linear correlation measure. We choose symmetrical uncertainty (denoted by  $IR_F$ ), based on entropy and information gain concepts.

Due to the high-dimensional data, we limit our comparison to sequential forward (SF) techniques (see Introduction section). We choose three representative subset evaluation measures in combination with SF search engine. One, denoted by  $SF_W$ , uses a target learning algorithm to estimate the worth of attribute subsets; the other two are subset search algorithms which exploit sequential forward search and utilize correlation measure (variation of CFS algorithm [8]) or consistency measure (variation of FOCUS [7]) to guide the search, denoted by  $CFS_{SF}$  and  $FOCUS_{SF}$  respectively (both of them used in [9]).

The experiments are conducted using the WEKA’s implementation of all these existing algorithms and our algorithm is also implemented in the WEKA environment [14]. For each data set, we run  $CFS_{SF}$  and  $FOCUS_{SF}$  algorithms (both of them are independent of the learning algorithm), and for each data set and each classifier, we run the wrapper feature selection algorithms,  $IR_W$ ,  $IR_F$  and  $SF_W$ . We record the running time and the number of selected features for each algorithm. We then apply the two classifiers (nb and c4) on the original data set as well as on each newly obtained data set containing only the selected features from each algorithm and record overall accuracy by a 10-fold cross-validation.

**Table 3.** Accuracy of c4 on selected features

| Data | Wrapper |     |                    |     |                    |     | Filter             |     |                    |     | Full               |
|------|---------|-----|--------------------|-----|--------------------|-----|--------------------|-----|--------------------|-----|--------------------|
|      | $IR_W$  |     | $IR_F$             |     | $SF_W$             |     | $CFS_{SF}$         |     | $FOCUS_{SF}$       |     |                    |
|      | Acc     | Att | Acc                | Att | Acc                | Att | Acc                | Att | Acc                | Att |                    |
| MK   | 96.83   | 7   | 96.30              | 7   | 96.44              | 6   | 95.54 <sup>-</sup> | 10  | 95.04 <sup>-</sup> | 11  | 96.88              |
| AR   | 74.32   | 6   | 73.02              | 5   | 74.10              | 8   | 69.04 <sup>-</sup> | 25  | 71.67              | 21  | 64.38 <sup>-</sup> |
| MA   | 83.50   | 9   | 80.20 <sup>-</sup> | 23  | 80.80 <sup>-</sup> | 11  | 74.55 <sup>-</sup> | 6   | 78.20 <sup>-</sup> | 15  | 70.35 <sup>-</sup> |
| MF   | 95.70   | 13  | 94.55              | 10  | 95.70              | 17  | 94.45 <sup>-</sup> | 86  | 91.40 <sup>-</sup> | 7   | 94.75              |
| AC   | 91.00   | 6   | 94.00              | 9   | 95.00              | 7   | N/A                |     | 77.00 <sup>-</sup> | 4   | 74.00 <sup>-</sup> |
| DE   | 88.00   | 12  | 88.33              | 17  | 90.33              | 12  | N/A                |     | 89.33              | 23  | 76.00 <sup>-</sup> |

Tables 2 and 3 report accuracy and number of features selected from nb and c4 respectively by each feature selection algorithm and the full set. We conduct an Students paired two-tailed t-test in order to evaluate the statistical significance of the difference between two averaged accuracy values: one resulted from  $IR_W$  and the other resulted from one of  $IR_F$ ,  $SF_W$ ,  $CFS_{SF}$ ,  $FOCUS_{SF}$  and the full set. The symbol " + " and " - " respectively identify statistically significant, at 0.1 level, wins or losses over  $IR_W$ . And Table 4 records the running time for each feature selection algorithm, showing two results for each wrapper approach, depending on the learning algorithm chosen.

Before we compare our technique with the others. Note the similarity between the results obtained with the two approaches of our algorithm, one based on a ranking-wrapper ( $IR_W$ ) and the other on a ranking-filter ( $IR_F$ ). As we can see from Table 2 and 3, in all the cases, except for one data set (MA) with c4 classifier, these accuracy differences are not statistically significant. the number of attributes selected are similar but  $IR_F$  is a little bit faster than  $IR_W$  because of the time needed to build the ranking for the wrapper-ranking approach.

Apart from the previous comparison, we study the behavior of  $IR_W$  comparing in three way: with respect to a whole set of features; with respect to another wrapper approach; and with respect to two filter approaches.

Classification accuracies obtained with the whole feature set are statistically lower than those obtained with our wrapper approach. As we can see from the last column in Table 2 and 3,  $IR_W$  wins in most of the cases, except in two data sets (MK and DE) and two data sets (MK and MF) for nb and c4 respectively. These accuracy differences are especially relevant in two data sets (AR and AC) and four (AR, MA, AC and DE) for nb and c4 respectively. We notice that the number of selected features is drastically low as regards the whole set.

For the two classifiers, no statistical significant differences are shown, except for c4 in MA data set, between the accuracy of our wrapper approach and the accuracy of the sequential forward wrapper procedure ( $SF_W$ ). On the other hand, the advantage of  $IR_W$  with respect to the  $SF_W$  for nb and c4 is clear. We can observe (see Table 4) that  $IR_W$  is consistently faster than  $SF_W$ . The time savings from  $IR_W$  become more obvious when the computer-load necessities of the mining algorithm increases. In many cases the time savings are in

**Table 4.** Running time (seconds) for each feature selection algorithm

| Data | Wrapper |        |        |        |        |        | Filter     |              |
|------|---------|--------|--------|--------|--------|--------|------------|--------------|
|      | nb      |        |        | c4     |        |        | $CFS_{SF}$ | $FOCUS_{SF}$ |
|      | $IR_W$  | $IR_F$ | $SF_W$ | $IR_W$ | $IR_F$ | $SF_W$ |            |              |
| MK   | 334     | 72     | N/A    | 2400   | 2700   | 10277  | 10         | 77           |
| AR   | 251     | 140    | 4089   | 291    | 245    | 2400   | 2          | 14           |
| MA   | 156     | 96     | 825    | 2460   | 5100   | 18000  | 5          | 52           |
| MF   | 1984    | 2643   | N/A    | 6502   | 5280   | 72000  | 73         | 45           |
| AC   | 1020    | 660    | 1027   | 1121   | 945    | 5820   | N/A        | 35           |
| DE   | 3300    | 2622   | 20280  | 9240   | 20880  | 86400  | N/A        | 1320         |

degrees of magnitude, and in two cases,  $SF_W$  did not report any results: for nb in MF data set  $SF_W$  did not produce any results after forty eight hours running (hence, neither selected features nor accuracy results); and in MK data set for nb classifier, results are not shown because the accuracy obtained with each individual feature is lower than without feature. These results verify the superior computational efficiency of incremental search applied by  $IR_W$  over greedy sequential search applied by  $SF_W$ , with similar number of attributes and without statistical significant differences.

In general, the computer-load necessities of filter procedures can be considered as negligible with respect to wrapper ones (Table 4), except for  $FOCUS_{SF}$  in DE data set. However, accuracies obtained with  $IR_W$  are notably better for nb and c4. Firstly, for the last two data sets (AC and DE) results were not produced by  $CFS_{SF}$  because the program ran out of memory after a period of considerably long time due to its quadratic space complexity. Secondly, in the rest of data sets,  $IR_W$  either improves or maintains the accuracy of both  $CFS_{SF}$  and  $FOCUS_{SF}$ . From Table 2 and 3, it can be seen that apart from the two last data sets,  $IR_W$  improves  $CFS_{SF}$  on two and four data sets for nb and c4 respectively. And  $IR_W$  improves  $FOCUS_{SF}$  on five and four data sets and no statistical significant differences on the rest.

## 5 Summary and Future Work

The success of many learning schemes, in their attempts to construct data models, hinges on the reliable identification of a small set of highly predictive attributes. The inclusion of irrelevant, redundant and noisy attributes in the model building process phase can result in poor predictive performance and increased computation. The most popular search methods in machine learning can not be applied to these data sets due to the large number of features. However, in this paper, we have proposed a new feature selection technique that lets us use a wrapper approach for finding a well suited feature set for classification. We use the incremental ranked usefulness definition to decide at the same time, whether a feature is relevant and non-redundant or not (non-relevant or redundant). The technique extracts the best non-consecutive features from the ranking, trying to

statistically avoid the influence of unnecessary attributes on the later classification. This new heuristic, named IR, shows an excellent performance comparing to the traditional sequential forward search technique, not only regarding the classification accuracy with respect to filter approaches, but also the computational cost with respect to the wrapper approach. By way of comparison, a rough estimate of the time required by the SF wrapper approach to choose this many features is on the order of thousands of hours, assuming the method does not get caught in a local minima first and prematurely stops adding attributes as a result.

## Acknowledgements

The research was supported by the Spanish Research Agency CICYT–Feder under grant TIN 2004-00159 and TIN 2004-06689-C03-03.

## References

1. Blum, A., Langley, P.: Selection of relevant features and examples in machine learning. In Greiner, R., Subramanian, D., eds.: *Artificial Intelligence on Relevance*. Volume 97. (1997) 245–271
2. Liu, H., Yu, L.: Toward integrating feature selection algorithms for classification and clustering. *IEEE Trans. on Knowledge and Data Engineering* **17** (2005) 1–12
3. Kohavi, R., John, G.: Wrappers for feature subset selection. *Artificial Intelligence* **1-2** (1997) 273–324
4. Guyon, I., Elisseeff, A.: An introduction to variable and feature selection. *Journal of Machine Learning Research* **3** (2003) 1157–1182
5. Liu, H., Setiono, R.: A probabilistic approach to feature selection: a filter solution. In: *13th Inter. Conf. on Machine Learning*, Morgan Kaufmann (1996) 319–327
6. Dash, M., Liu, H., Motoda, H.: Consistency based feature selection. In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. (2000) 98–109
7. Almuallim, H., Dietterich, T.: Learning boolean concepts in the presence of many irrelevant features. *Artificial Intelligence* **69** (1994) 279–305
8. Hall, M.: Correlation-based feature selection for discrete and numeric class machine learning. In: *17th International Conf. on Machine Learning*, Morgan Kaufmann, San Francisco, CA (2000) 359–366
9. Yu, L., Liu, H.: Efficient feature selection via analysis of relevance and redundancy. *Journal of machine learning research* **5** (2004) 1205–24
10. Inza, I., Larrañaga, P., Blanco, R., Cerrolaza, A.: Filter versus wrapper gene selection approaches in dna microarray domains. *Artificial Intelligence in Medicine* **31** (2004) 91–103
11. Xiong, M., Fang, X., Zhao, J.: Biomarker identification by feature wrappers. *Genome Res* **11** (2001) 1878–87
12. Xing, E., Jordan, M., Karp, R.: Feature selection for high-dimensional genomic microarray data. In: *Proc. 18th International Conf. on Machine Learning*, Morgan Kaufmann, San Francisco, CA (2001) 601–608
13. Guyon, I., Weston, J., Barnhill, S., Vapnik, V.: Gene selection for cancer classification using support vector machine. *Machine Learning* **46** (2002) 389–422
14. Witten, I., Frank, E.: *Data Mining: Practical machine learning tools with Java implementations*. Morgan Kaufmann, San Francisco (2000)

# Intrinsic Dimensionality Maps with the PCASOM

Ezequiel López-Rubio, Juan Miguel Ortiz-de-Lazcano-Lobato,  
María del Carmen Vargas-González, and José Miguel López-Rubio

School of Computer Engineering, University of Málaga, Campus de Teatinos,  
s/n 29071 Málaga, Spain  
{ezeqlr, jmortiz}@lcc.uma.es  
<http://www.lcc.uma.es>

**Abstract.** The PCASOM is a novel self-organizing neural model that performs Principal Components Analysis (PCA). It is also related to the ASSOM network, but its training equations are simpler. The PCASOM has the ability to learn self-organizing maps of the means and correlations of complex input distributions. Here we propose a method to extend this capability to build intrinsic dimensionality maps. These maps model the underlying structure of the input. Experimental results are reported, which show the self-organizing map formation performed by the proposed network.

## 1 Introduction

The *Principal Components Analysis Self-Organizing Map* (PCASOM) [6] is an extension of the self-organizing feature map (SOFM) proposed by Kohonen [4]. The SOFM builds self-organizing maps where each neuron stores the centroid of its *receptive field*, i.e., the input subset which it represents. Each PCASOM neuron extends this by learning an estimation of the *covariance matrix* of its receptive field. This is related to the ASSOM model [5], where each neuron learns a vector subspace. These three models are adequate to create topographic maps, which are faithful representations of the input space. However, their representation ability is different, because the SOFM models the receptive field by a mean vector, while the other two build more elaborate representations. Nevertheless, both the ASSOM and the PCASOM are unable to learn the *intrinsic dimensionality* of the input data, which is a standard measure of the underlying complexity of a dataset ([2], [11]). This limitation is shared by other PCA-related procedures, such as the local PCA networks, where the input distribution is partitioned into meaningful clusters ([3], [9], [10]).

The Principal Components Analysis Competitive Learning (PCACL) [7] is a novel local PCA network, which uses the *explained variance method* to estimate the intrinsic dimensionality. In this paper we combine this method with the original PCASOM model to obtain topographic maps of the intrinsic dimensionality of an input distribution.

Sections 2 and 3 are devoted to the PCASOM model and the application of the explained variance method to PCASOM, respectively. In Section 4 we prove some important properties of our proposal, and a brief discussion is considered in Section 5. Finally, sections 6 and 7 deal with experimental results and conclusions.

## 2 The PCASOM Model

In the time instant  $t$  each unit  $j$  of the PCASOM stores the matrix  $\mathbf{R}_j(t)$  as a covariance matrix estimation and the vector  $\mathbf{e}_j(t)$  as a mean vector estimation. When an input sample is presented to a self-organizing map, a competition is hold among the neurons. Every unit (say,  $j$ ) of our network has an associated orthonormal vector basis  $B^j(t)$  at every time instant  $t$ . It is formed by the  $K_j$  eigenvectors corresponding to the  $K_j$  largest eigenvalues of  $\mathbf{R}_j(t)$ . The neuron  $c$  that has the minimum sum of projection errors for the input vectors  $\mathbf{x}^i(t)$ ,  $i=1, \dots, N$ , is declared the winner:

$$c = \arg \min_j \left( \sum_{i=1}^N \left\| \mathbf{x}^i(t) - \mathbf{e}_j(t) - \text{Orth}(\mathbf{x}^i(t) - \mathbf{e}_j(t), B^j(t)) \right\|^2 \right) \quad (1)$$

where  $\text{Orth}(\mathbf{x}, B)$  is the orthogonal projection of vector  $\mathbf{x}$  on basis  $B$ . So, we are looking for the neuron which best represents the inputs  $\mathbf{x}^i(t)$ .

We consider a topology that defines which neurons are neighbors. When a neuron  $c$  wins the competition it is updated. Its neighbors are also updated, according to the *degree of neighborhood*  $\pi_{j,c}$  between winning neuron  $c$  and its neighbor  $j$ :

$$\pi_{j,c}(t) = \exp \left( - \frac{d_{j,c}^2}{2\sigma(t)^2} \right) \quad (2)$$

In equation (2),  $d_{j,i}$  is the distance between winning neuron  $i$  and neuron  $j$ , and the value  $\sigma(t)$  controls the *neighborhood size*. The degree of neighborhood and the learning rate are combined to control the updating of the neurons. For every neuron  $i$  we have

$$\mathbf{e}_j(t+1) = \mathbf{e}_j(t) + \eta_e(t) \pi_{j,c}(t) \left[ \frac{1}{N} \left( \sum_{i=1}^N \mathbf{x}_i \right) - \mathbf{e}_j(t) \right] \quad (3)$$

$$\mathbf{A}_j(t+1) = \frac{1}{N} \sum_{i=1}^N (\mathbf{x}_i - \mathbf{e}_j(t+1)) (\mathbf{x}_i - \mathbf{e}_j(t+1))^T \quad (4)$$

$$\mathbf{R}_j(t+1) = \mathbf{R}_j(t) + \eta_R(t) \pi_{j,c}(t) (\mathbf{A}_j(t+1) - \mathbf{R}_j(t)) \quad (5)$$

where  $\eta_R(t)$  is *learning rate for the covariance matrix* and  $\eta_e(t)$  is the *learning rate for the mean*. These expressions are very similar to the weight update equations of the competitive learning and the self-organizing feature map (SOFM). Please see [6] for a detailed justification of the model equations.

The learning process is divided into two phases, like the standard self-organizing map algorithms: the *ordering phase* and the *convergence phase*. It is during the ordering phase when the topological ordering of the neurons takes place. The learning rates typically present a linear or exponential decay in this phase, and so does the neighborhood size  $\sigma(t)$ . The convergence phase is required principally for the

fine-tuning of the computational map. The learning rates are maintained at a small, constant value during this phase.

### 3 The Explained Variance Method for the PCASOM

PCA methods form a vector base  $B=\{\mathbf{b}_h \mid h=1,\dots,K\}$  with the eigenvectors  $\mathbf{b}_h$  corresponding to the  $K$  largest eigenvalues of  $\mathbf{R}$ , where  $K$  is a parameter which is specified before the start of the learning process. In the case of local PCA neural models and PCASOM, each neuron  $j$  has a covariance matrix  $\mathbf{R}_j$  and a mean vector  $\mathbf{e}_j$ .

The explained variance method considers a variable number of basis vectors  $K_j(t)$ , which is computed independently for each neuron  $j$ . This number reflects the intrinsic dimensionality of the data in the cluster of data represented by neuron  $j$ . Many methods to estimate intrinsic dimensionality are based in the analysis of the eigenvalues  $\lambda_j^p(t)$  of the covariance matrix  $\mathbf{R}_j(t)$ ,  $p=1,\dots,D$ .

If we use a vector base  $B_Z=\{\mathbf{b}_1,\dots,\mathbf{b}_Z\}$ , where  $Z$  is the number of basis vectors, the mean squared error associated with the cluster of data of the neuron is given by:

$$MSE_Z = E[\|\mathbf{x} - \mathbf{e}_j - Orth(\mathbf{x} - \mathbf{e}_j, B_Z)\|^2] \tag{6}$$

Note that if  $Z=0$  we get the maximum possible mean squared error. The goal here is to select a number of basis vectors  $K_j$  which ensures that at least a fraction  $\alpha$  of the maximum mean squared error is removed,

$$K_j = \min\{Z \in \{0,1,\dots,D\} \mid MSE_0 - MSE_Z \geq \alpha MSE_0\} \tag{7}$$

where  $\alpha \in [0,1]$ . This is achieved if we take (see [0]):

$$K_j(t) = \min\left\{Z \in \{0,1,\dots,D\} \mid \sum_{p=1}^Z \lambda_j^p(t) \geq \alpha trace(\mathbf{R}_j(t))\right\} \tag{8}$$

The parameter  $\alpha$  is the amount of variance which we want the neurons to explain. Hence, we select  $K_j(t)$  so that the amount of variance explained by the directions associated to the  $K_j(t)$  largest eigenvalues is at least  $\alpha$ .

We can summarize the algorithm for PCASOM with the explained variance method as follows:

1. For every unit  $j$ , obtain the initial covariance matrix  $\mathbf{R}_j(0)$  by generating a random symmetric nonnegative matrix, with the elements of the main diagonal near to one, and all the others near to zero.
2. For every unit  $j$ , build the vector  $\mathbf{e}_j(0)$  by using small random values, either negative or positive, for its components, and use (8) to compute  $K_j$ .
3. At time instant  $t$ , select the input vectors  $\mathbf{x}^i(t)$ ,  $i=1,\dots,N$ , with  $N \geq I$ , from the input distribution. Compute the winning neuron  $c$  according to (1).
4. For every unit  $j$ , update the vector  $\mathbf{e}_j$  and the matrix  $\mathbf{R}_j$  by using (3), (4) and (5).
5. For every unit  $j$ , use (8) to recompute  $K_j$ .
6. If convergence has been reached, stop the simulation. Otherwise, go to step 3.

## 4 Properties

**Proposition 1.** *Let  $z \in \{0, 1, \dots, D\}$ . If  $\alpha \in [0, z/D]$ , then for every neuron  $j$  and every time instant  $t$ , it holds that  $0 \leq K_j(t) \leq z$ .*

**Proof:** As the eigenvalues are sorted in decreasing order, it holds that:

$$\forall p \in \{1, \dots, z\} \forall q \in \{z + 1, \dots, D\}, \lambda_j^p(t) \geq \lambda_j^q(t) \tag{9}$$

$$\Rightarrow (D - z) \sum_{p=1}^z \lambda_j^p(t) \geq z \sum_{q=z+1}^D \lambda_j^q(t) \tag{10}$$

Next we add  $z \sum_{p=1}^z \lambda_j^p(t)$  to both sides, and then we divide by  $D$  to yield

$$\sum_{p=1}^z \lambda_j^p(t) \geq \frac{z}{D} \sum_{q=1}^D \lambda_j^q(t) \tag{11}$$

By the hypotheses, we have that  $\alpha \in [0, z/D]$ , so:

$$\sum_{p=1}^z \lambda_j^p(t) \geq \alpha \sum_{q=1}^D \lambda_j^q(t) \tag{12}$$

Then by (8) and (12) we have  $0 \leq K_j(t) \leq z$ , as desired.

**Proposition 2.** *If we take  $\alpha=0$ , the PCASOM model with the explained variance method reduces to Kohonen’s SOFM.*

**Proof:** We use  $z=0$  in the previous proposition, and we get  $\alpha=0 \Rightarrow K_j(t)=0$  for every neuron  $j$ . Then the covariance matrices are not used in the competitive phase, so it reduces to that of the original SOFM. The estimated mean vectors  $e_j(t)$  play the role of the weight vectors.

## 5 Discussion

The proposed model has the following advantages:

- a) It learns the number of basis vectors which are needed to represent the input distribution with a specified accuracy. That is, the basis vectors are selectively removed or added to the neurons as needed.
- b) The learned number of basis vectors can be used to obtain intrinsic dimensionality information. Low values of  $K_j$  mean that the region represented by neuron  $j$  has a low dimensionality, and vice versa. Hence, a *intrinsic dimensionality map* is obtained when we arrange the values  $K_j$  according to the network topology. This is a topographic map, which models how the intrinsic dimensionality varies in the input distribution.
- c) It is a generalization of Kohonen’s SOFM, as proven in the previous section.



## 6 Experimental Results

We have designed a set of experiments to test the map formation ability of our method. For this unsupervised task we have used two freely-accessible data sets.

The first data set comes from the VizieR service [12, which is an information system for astronomical data. In particular, we have selected the Table 6 of the Complete near-infrared and optical photometric CDFS Catalog from Las Campanas Infrared Survey [1]. We have extracted 22 numerical features from 10,000 stars. Hence, we have 10,000 sample vectors. These data have been normalized in order to cope with the variability and the very heterogeneous scaling of the original data.

The second data set is from the NASA Earth Observatory [8]. This data set is made up of climatic images taken every month by satellites. We have selected January 1988, and 14 parameters for this month. So we have 14 images, each with 360x180 pixels. These 14 images are combined to form a single multisensor image, where every pixel is a vector with 14 components. The values of the components are real numbers in the interval [0,1]. So we have 360x180=64800 sample vectors.

For both datasets, we have used  $\alpha=0.99$ , 200,000 epochs and linear decay rates for the learning rate and the neighbourhood size. The startup of the ordering phase has been  $\eta_e(0)=\eta_R(0)=0.5$ ,  $\sigma(0)=1$ , with null final values of the three parameters. The convergence phase has used constant values  $\eta_e=\eta_R=0.04$ ,  $\sigma=0.04$ .

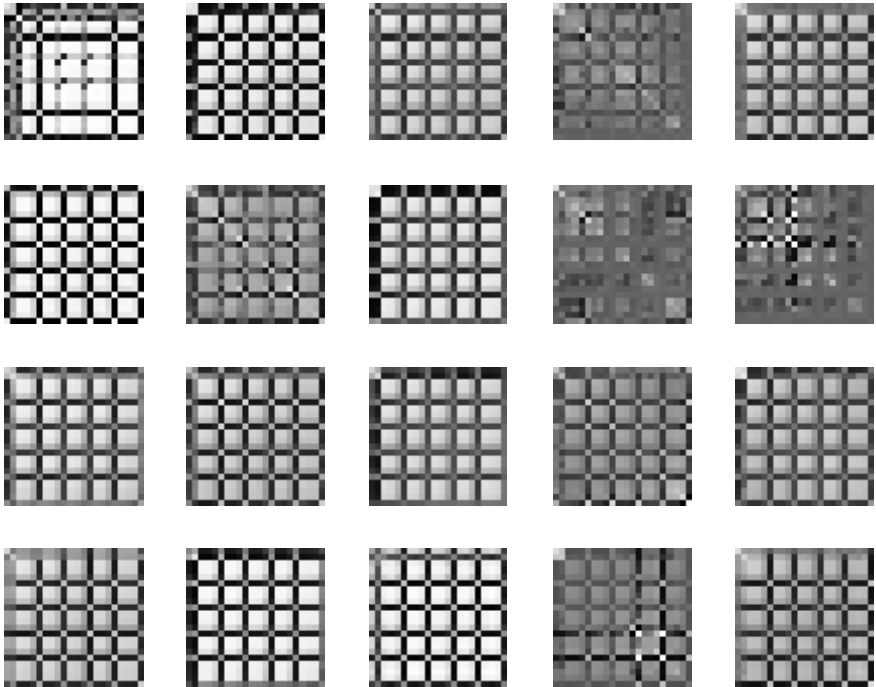
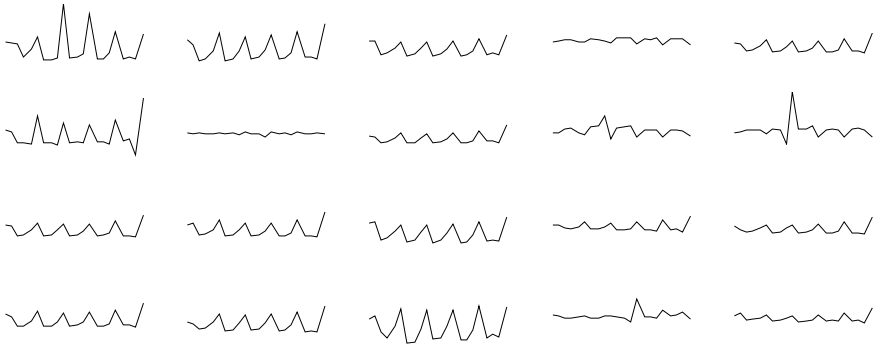


Fig. 1. Covariance matrices for the first data set, 4x5 rectangular lattice



**Fig. 2.** Mean vectors for the first data set, 4x5 rectangular lattice

Figures 1 to 3a show the results with the first dataset. We have tested a 4x5 rectangular topology (Figs. 1–3a), and a 20x20 rectangular topology (Fig. 4). Figure 1 shows the final covariance matrices, Fig. 2 is the final mean vectors, and Figs. 3a and 4 are the intrinsic dimensionality maps. These figures illustrate the formation of topographic maps, where neighbour neurons have similar values (darker tones mean higher values). In particular, Figs. 3a and 4 depict the intrinsic dimensionality modelling of the input distribution. Figure 3b shows the intrinsic dimensionality map obtained from a 4x5 rectangular topology with the second data set. It must be highlighted that two regions can be distinguished in this map (one with lower



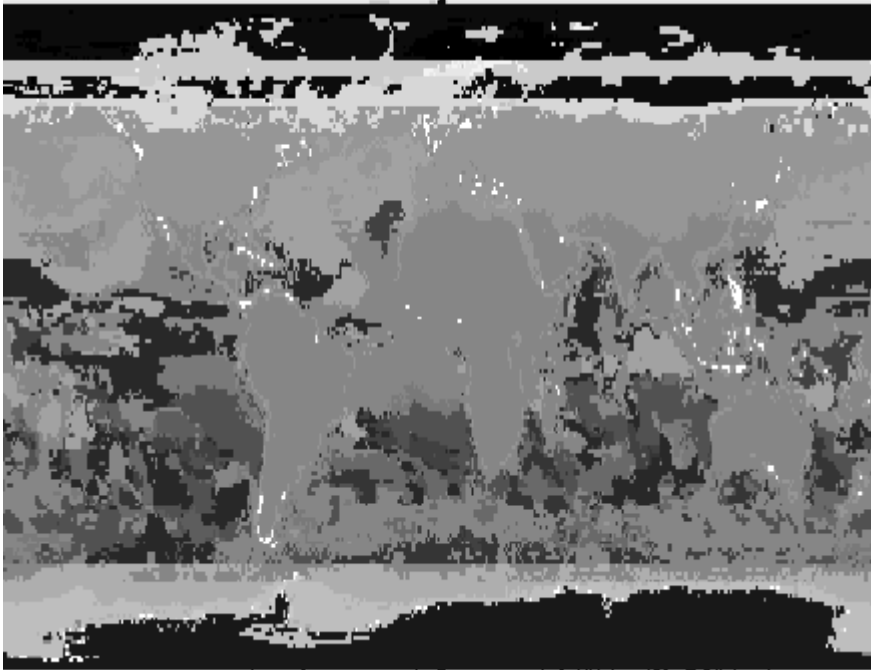
**Fig. 3.** Intrinsic dimensionality maps, 4x5 lattice: (a) data set 1 (left). (b) data set 2 (right)



**Fig. 4.** Intrinsic dimensionality map for the first data set, with a 20x20 rectangular lattice

dimensionality to the left and the other with higher dimensionality). There also a coherence between the neighbourhood in the map and the similarity in the value (indicated as gray levels).

Figure 5 shows the receptive field of each neuron in a different gray level, with the samples arranged according to their geographic location. We can see in this figure that the receptive fields correspond to geographical zones with similar climatic conditions. This is achieved despite the fact that the network is not feeded any information about geographic location (only climatic information is provided).



**Fig. 5.** Receptive fields of the neurons for the second data set

## 7 Conclusions

We have adapted the explained variance method to the PCASOM model in order to build self-organizing maps with intrinsic dimensionality information. This allows to obtain intrinsic dimensionality maps, which can be used to analyze input distributions more deeply than the ASSOM model and the original PCASOM. These maps represent how the variability of the data changes in the input distribution. Two important properties of our proposal have been proved. One of them explains the role of the parameter introduced by the explained variance method, and the other states that the proposed model is an extension of Kohonen's SOFM. Finally, experimental results have been presented to show its map formation capability.

## References

1. Chen, H.-W., et al., Early-type galaxy progenitors beyond  $z=1$ . *Astrophysical Journal*, 560, L131 (2001).
2. K. Fukunaga and D.R. Olsen, 'An Algorithm for Finding Intrinsic Dimensionality of Data', *IEEE Trans. Computers*, 20 (2), 176–183, (1971).
3. Kambhatla, N. & Leen, T.K. (1997). Dimension Reduction by Local Principal Component Analysis. *Neural Computation*, 9 (7), 1493-1516.
4. Kohonen, T. (1990). The Self-Organizing Map, *Proc. IEEE*, 78, 1464-1480.
5. Kohonen, T. (1995). The Adaptive-Subspace SOM (ASSOM) and its Use for the Implementation of Invariant Feature detection. F. Fogelman-Soulié & P. Galnari, Eds., *Proc. ICANN'95, International Conference on Artificial Neural Networks*, 1, 3-10.
6. López-Rubio, E., Muñoz-Pérez, J., Gómez-Ruiz, J.A.: A Principal Components Analysis Self-Organizing Map. *Neural Networks* 17(2) (2004) 261–270.
7. López-Rubio, E., Ortiz-de-Lazcano-Lobato, J.M., Muñoz-Pérez, J., Gómez-Ruiz, J.A.: Principal Components Analysis Competitive Learning. *Neural Computation* 16(11) (2004), 2459–2481.
8. NASA Earth Observatory, Data & Images [online]. Available at: <http://earthobservatory.nasa.gov/Observatory/datasets.html> (November 28, 2002).
9. Roweis, S., & Ghahramani, Z. (1999). A Unifying Review of Linear Gaussian Models, *Neural Computation*, 11, 305-345.
10. Tipping, M.E., & Bishop, C.M. (1999). Mixtures of Probabilistic Principal Components Analyzers, *Neural Computation*, 11, 443-482.
11. Verveer, P.J. & Duin, R.P.W. (1995). An Evaluation of Intrinsic Dimensionality Estimators, *IEEE Trans. Pattern Analysis and Machine Intelligence*, 17 (1), 81-86.
12. Vizier service [online]. Available at: <http://vizier.cfa.harvard.edu/viz-bin/VizieR> (March 29, 2004).

# The Curse of Dimensionality in Data Mining and Time Series Prediction

Michel Verleysen<sup>1</sup> and Damien François<sup>2,\*</sup>

Universit e catholique de Louvain, Machine Learning Group,

<sup>1</sup>Place du Levant, 3, 1380 Louvain-la-Neuve, Belgium

<sup>2</sup>Avenue G. Lemaitre, 4, 1380 Louvain-la-Neuve, Belgium

verleysen@dice.ucl.ac.be, francois@auto.ucl.ac.be

www.ucl.ac.be/mlg

**Abstract.** Modern data analysis tools have to work on high-dimensional data, whose components are not independently distributed. High-dimensional spaces show surprising, counter-intuitive geometrical properties that have a large influence on the performances of data analysis tools. Among these properties, the concentration of the norm phenomenon results in the fact that Euclidean norms and Gaussian kernels, both commonly used in models, become inappropriate in high-dimensional spaces. This papers presents alternative distance measures and kernels, together with geometrical methods to decrease the dimension of the space. The methodology is applied to a typical time series prediction example.

## 1 Introduction

Modern data analysis has to cope with tremendous amounts of data. Data are indeed more and more easily acquired and stored, due to huge progresses in sensors and ways to collect data on one side, and in storage devices on the other side. Nowadays, there is no hesitation in many domains in acquiring very large amounts of data without knowing in advance if they will be analyzed and how.

The spectacular increase in the amount of data is not only found in the number of samples collected for example over time, but also in the number of attributes, or characteristics, that are simultaneously measured on a process. The same arguments lead indeed to a kind of precaution principle: as there is no problem in measuring and storing many data, why not to collect many measures, even if some (many) of them prove afterward to be useless or irrelevant? For example, one could increase the number of sensors in a plant that has to be monitored, or increase the resolution of measuring instruments like spectrometers, or record many financial time series

---

\* Michel Verleysen is a Senior Associate of the Belgian National Fund for the Scientific Research. The work of D. Francois is funded by a grant from the Belgian FRIA. Part of the work of D. Francois is supported by the Interuniversity Attraction Pole (IAP), initiated by the Belgian Federal State, Ministry of Sciences, Technologies and Culture. The scientific responsibility rests with the authors.

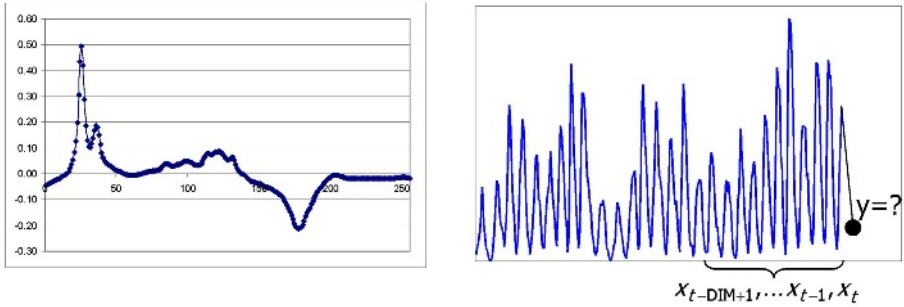
simultaneously in order to study their mutual influences, etc. In all these situations, data are gathered into vectors whose dimension correspond to the number of simultaneous measurements on the process of phenomenon. When the dimension grows, one speaks about high dimensional data, as each sample can be represented as a point or vector in a high-dimensional space.

The difficulty in analyzing high-dimensional data results from the conjunction of two effects. First, high-dimensional spaces have geometrical properties that are counter-intuitive, and far from the properties that can be observed in two-or three-dimensional spaces. Secondly, data analysis tools are most often designed having in mind intuitive properties and examples in low-dimensional spaces; usually, data analysis tools are best illustrated in 2-or 3-dimensional spaces, for obvious reasons. The problem is that those tools are also used when data are high-dimensional and more complex. In this kind of situations, we loose the intuition of the tools behavior, and might draw wrong conclusions about their results. Such loss of control is already encountered with basic linear tools, such as PCA (Principal Component Analysis): it is very different to apply PCA on a 2-dimensional example with hundreds of samples (as illustrated in many textbooks), or to apply it on a few tens of samples represented in a 100-dimensional space! Known problems such as collinearity and numerical instability easily occur. The problem is even worse when using nonlinear models: most nonlinear tools involve (much) more parameters than inputs (i.e. than the dimension of the data space), which results in lack of model identifiability, instability, overfitting and numerical instabilities.

For all these reasons, the specificities of high-dimensional spaces and data must then be taken into account in the design of data analysis tools. While this statement is valid in general, its importance is even higher when using nonlinear tools such as artificial neural networks. This paper will show some of the surprising behaviors of high-dimensional data spaces, what are the consequences for data analysis tools, and paths to remedies. In Section 2, examples oh high dimensional data are given, along with some details about the problems encountered when analyzing them. Section 3 details surprising facts in high-dimensional spaces and some ideas that could be incorporated in the tools to lower the impact of these phenomena. In Section 4, the current research about nonlinear dimension reduction tools is briefly presented, as another way to face the problems encountered in high-dimensional spaces. Finally, Section 5 gives an example of a time series prediction task where the dimensionality of the regressors has to be taken into account.

## 2 High-Dimensional Data

Working with high-dimensional data means working with data that are embedded in high-dimensional spaces. When speaking about non-temporal data, this means that each sample contains many attributes or characteristics. Spectra are typical examples of such data: depending on the resolution of the spectrometer, spectra contain several hundreds of measurements (see Figure 1 left). Fortunately for the sake of analysis, the hundreds of coordinates in spectra are not independent: it is precisely their



**Fig. 1.** Examples of high-dimensional data. Left: spectrum; right: regressor for a time series prediction problem

dependencies that are analyzed in order to extract relevant information from a set of spectra [1]. More generally, redundancy in the coordinates is a necessary condition to analyse a low number of samples in a high-dimensional space. Indeed let us imagine on the contrary that all coordinates are independent; a simple linear regression model will contain as many parameters as the number of coordinates in the space. If the number of samples available for learning is less than the dimension of the space, the problem is undefined (in other words the model is unidentifiable). This problem is known as collinearity, and has no other solution than exploiting the dependencies between coordinates in order to reduce the number of model parameters; using smoothing splines is an example of dependency exploitation [2]. While collinearity is the expression of this phenomenon when linear models are used, a similar problem appears when nonlinear models are used; it results in overfitting, i.e. in a too efficient modelling of learning samples without model generalization ability.

An example of high-dimensional data with temporal dependencies is shown in Figure 1 right. Knowing a time series up to time  $t$ , the problem consists in forecasting the next value(s) of the series. Without additional information from exogeneous variables, the forecasting problem is solved by building a regression model with a number of (often consecutive) values from the time series, and with output the next value. The model is built on the known part of the series, and used to predict unknown values. When no indication is available on the optimal regressor size, large regressors are usually preferred, in order to avoid losing relevant information necessary for the prediction. However large regressors mean high-dimensional input data to the model, a large number of parameters, and the same difficulties as the ones encountered with the first example.

In both situations, the goal will be threefold:

- to take into account in the model the dependencies between characteristics, in order to avoid a large number of effective model parameters;
- to adapt the design of the model to the specificities of high-dimensional spaces.
- to reduce, whenever possible, the dimensionality of the data through selection and projection techniques;

The first goal is highly problem-dependent and beyond the scope of this paper. The second and third goal will be respectively discussed in Sections 3 and 4.

### 3 Surprising Facts in High-Dimensional Spaces and Remedies

This section describes some properties of high-dimensional spaces, that are counter intuitive compared to similar properties in low-dimensional spaces. Consequences on data analysis are discussed, with possible ideas to be incorporated in data analysis tools in order to meet the specific requirements of high-dimensional spaces.

#### 3.1 The Curse of Dimensionality

Data analysis tools based on learning principles infer knowledge, or information, from available learning samples. Obviously, the models built through learning are only valid in the range or volume of the space where learning data are available. Whatever is the model or class of models, generalization on data that are much different from all learning points is impossible. In other words, relevant generalization is possible from interpolation but not from extrapolation.

One of the key ingredients in a successful development of learning algorithms is therefore to have enough data for learning so that they fill the space or part of the space where the model must be valid. It is easy to see that, every other constraint being kept unchanged, the number of learning data should grow exponentially with the dimension (if 10 data seem reasonable to learn a smooth 1-dimensional model), 100 are necessary to learn a 2-dimensional model with the same smoothness, 1000 for a 3-dimensional model, etc.). This exponential increase is the first consequence of what is called the curse of dimensionality [3]. It is among others illustrated by Silverman on the problem of the number of kernels necessary to approximate a dimension-dependent distribution up to a defined precision [4].

More generally, the curse of dimensionality is the expression of all phenomena that appear with high-dimensional data, and that have most often unfortunate consequences on the behavior and performances of learning algorithms.

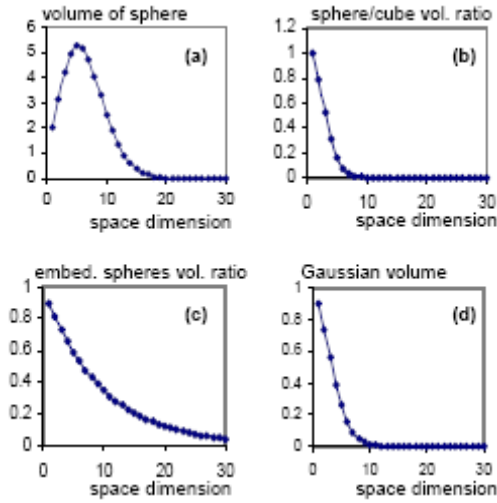
#### 3.2 Some Geometrical Properties of High-Dimensional Spaces

Even without speaking about data analysis, high-dimensional spaces have surprising geometrical properties that are counter-intuitive. Figure 2 illustrates four such phenomena.

Figure 2 a) shows the volume of a unit-radius sphere with respect to the dimension of the space. It is seen that while this volume increases from dimension 1 (a segment) to 5 (a 5-dimensional hypersphere), it then decreases and reaches almost 0 as soon as the space dimension exceeds 20. The volume of a 20-dimensional hypersphere with radius equal to 1 is thus almost 0!

Figure 2 b) shows the ratio between the volume of a unit-radius sphere and the volume of a cube with edge lengths equal to 2 (the sphere is thus tangent to the cube). In dimension 2, the ratio is obviously  $\pi/4$ , which means that most of the volume (here surface) of the cube is also contained in the sphere. When the dimension increases, this ratio rapidly decreases toward 0, to reach a negligible value as soon as the dimension reaches 10. In terms of density of data in a space, this means that if samples are drawn randomly and uniformly in a cube, the probability that they fall near the corners of the cube is almost one! As it will be detailed below, this also means that their norm is far from being random (it is concentrated near the maximum value, i.e. the square root of the dimension).

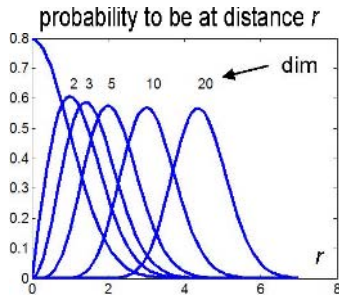




**Fig. 2.** Four phenomena in high-dimensional spaces

Figure 2 c) shows the ratio between the volumes of two embedded spheres, with radii equal to 1 and 0.9 respectively. Unsurprisingly the ratio decreases exponentially with the dimension. What is more surprising is that, even if the two radii only differ by 10%, the ratio between both volumes is almost 0 in dimension 10. If data are randomly and uniformly distributed in the volume of the larger sphere, this means that almost all of them will fall in its skull, and will therefore have a norm equal to 1!

Finally, one can consider a multi-dimensional Gaussian distribution scaled to have its integral equal to 1. Figure 2 d) shows the percentage of the volume of the Gaussian function that falls inside a radius equal to 1.65. It is well known that this percentage is equal to 90% in dimension 1. Figure 2 d) shows that this percentage rapidly decreases, up to almost 0 in dimension as low as 10! In other words, in dimension 10, almost all the volume of a Gaussian function is contained in its tails and not near its center, a definition that contracts with the commonly accepted view of locality!



**Fig. 3.** Probability of a point from a Normal distribution to be at distance  $r$  of the center, for several space dimensions

More than geometrical properties, these four examples show that data even uniformly distributed concentrate in unexpected parts of the space, that norms of vectors do not follow intuitive distributions, and that functions considered as local are not local anymore. Such properties should definitely be taken into account in the design of data analysis algorithms.

### 3.3 The Concentration of Norms

More directly related to data analysis is the so-called concentration of the norm phenomenon. Figure 3 illustrates this phenomenon in the case of Gaussian distributions (with standard deviations equal to 1). For several dimensions of the space (1, 2, 3, 5, 10 and 20), the figure shows the probability density functions (pdf) to find a point drawn according to a Gaussian distribution, at distance  $r$  from the center of that distribution. In dimension 1, this pdf is a monotonically decreasing function. In dimension 2, it has a bell shape (not exactly a Gaussian one; it actually is a Rayleigh distribution) with a peak around 1, that illustrates the fact that there are more points at distance 1 from the center than at distance 0.2 or 2. When the dimension increases, the bell shape remains, but is shifted to the right. In dimension 20 for example, the percentage of data lying at distance less than 2 from the center is so low that it cannot be seen at the scale of the figure (despite the fact that the standard deviation of the Gaussian distribution is 1). This means that the distances between all points and the center of the distribution are concentrated in a small interval. Relative differences between these distances vanish; these distances become less and less discriminative (therefore relevant for analysis) when the dimension increases.

The concentration of norm phenomenon is more precisely described in several mathematical results. In 1994, Demartines [5] has shown that for random vectors with independent and identically distributed components, the mean of their Euclidean norm increases as the square root as the dimension of the space, while the variance of their norm does not increase. He concludes that if the dimension is high, all vectors are normalized, as the error resulting from taking the mean of their norm instead of their actual norm becomes negligible.

Independently from Demartines' results, Beyer [6] proved that when the dimension increases, the relative difference between the largest and smallest norm in a dataset converge to zero in probability. The relative difference is the difference between the largest and smallest norms, divided by the smallest one; it is called the relative contrast. The result is valid for arbitrary distance measures, under mild conditions on the distributions. Beyer concludes that, in a nearest neighbor search context, all points converges to approximately the same distance from the query point. The notion of nearest neighbor becomes less intuitive in high-dimensional spaces.

Other norms than the Euclidean distance may be used in data analysis tools. For example, Minkowski norms with order  $p$  are defined as the  $p^{\text{th}}$  root of the sum of the vector components to the power  $p$ :

$$\|(x_1, \dots, x_d)\|_p = \left( \sum_i |x_i|^p \right)^{\frac{1}{p}}.$$

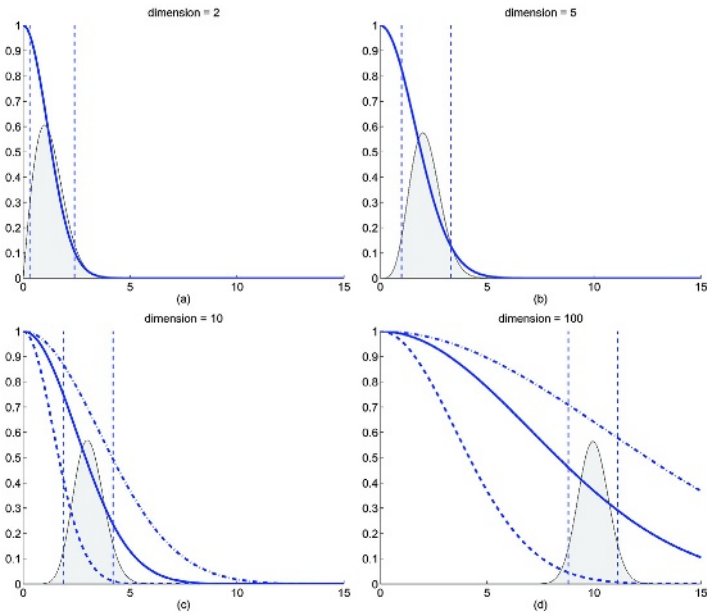
Hinneburg [7] has shown that the absolute difference between the largest and smallest norm in a set (called the contrast) is somewhat constant with dimension for the Euclidean distance. However it increases with dimension for the  $L_1$  norm and rapidly decreases toward 0 for norms with an order greater than 2. Hinneburg concludes that nearest neighbor search with  $L_p$  norms when  $p > 2$  are meaningless in high-dimensional spaces.

Aggarwal [8] has extended Hinneburg’s results to fractional norms (the value of  $p$  is no more restricted to be an integer). He proved that the relative contrast tends toward zero with the dimension, faster for large values of  $p$ . He then pleads in favor of small values of  $p$ .

Francois [9] has shown that in the context of nearest neighbor search, the optimal distance to be used also depends on the type of noise on the data. He has shown that fractional norms are preferable in the case of colored noise, but if Gaussian noise is assumed, then Euclidean metrics is more robust than fractional ones. This might give a lower limit to the value of  $p$  that Aggarwal prones to be small.

### 3.4 Gaussian Kernels

In many data analysis methods, Gaussian kernels are used. This is the case in RBFN (Radial-Basis Function Networks), in most SVM (Support Vector Machines) and LS-SVM (Least-Squares Support Vector Machines), and many others. Most often, the use of Gaussian kernels is justified by two properties:



**Fig. 4.** Kernel values as a function of the distance to their centers for several space dimensions, along with the distribution of distances for normally distributed data. Vertical lines correspond to 5 and 95 percentile resp

- Gaussian kernels are deemed to be local. Embedded in a model, using local functions avoids illegitimate generalization in regions of the space that are empty from learning points (it avoids the already mentioned extrapolation problem).
- Gaussian kernels are used (up to a scale factor) as a smooth, thus differentiable, measure of similarity between two points. A Gaussian function evaluated on the norm of the vector difference between two points (as usually done in RBFN and SVM) results in a high value if the points are close, and in a low one if the points are far one from the other.

The first property has already been discussed above. The second one makes the hypothesis that Gaussian kernels can discriminate between close and far points in a distribution. As illustrated in Figure 4 (from [10]), this is not the case in high-dimensional spaces either. The bell-shaped curves (thin lines) show, in dimensions 2, 5, 10 and 100, the distribution of distances between each sample and the center of a multi-dimensional Gaussian distribution; the vertical lines correspond to the 5% and 95% percentiles, respectively. Gaussian kernels are superimposed on the graphs (thick lines). As it can be seen in dimensions 2 and 5, the values taken by the Gaussian kernels are very different for small and large distances found in the distribution (see for example the dotted vertical lines). However in dimension 100 this does not remain true anymore. Even by adjusting the standard deviation of the Gaussian kernels (see the dotted kernels), they remain flat in the range of effective distances in the distribution. It becomes clear that Gaussian kernels are not appropriate for high-dimensional data analysis tools.

In order to overcome this problem, Francois [10] suggests to use so-called p-Gaussian kernels, defined as

$$K(x, y) = \exp(-d(x, y)^p / \sigma^p),$$

where  $p$  and  $\sigma$  are two parameters allowing to adjust not only the slope of the function but also a shift to larger distances. Through the knowledge of a robust measure (for example percentiles) of effective range of distances in the distribution, it is possible to adjust the  $p$  and  $\sigma$  values in order to be optimal for a specific dataset.

More flexible kernels such as p-Gaussians should certainly be used in high-dimensional contexts, to overcome the limitations of Gaussian ones.

## 4 Reducing Dimension

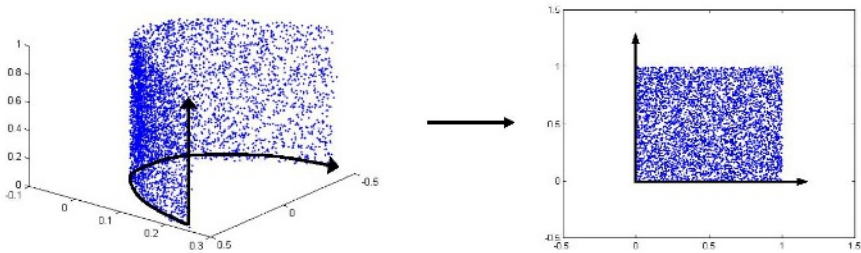
When faced to difficulties resulting from the high dimension of the space, a possibility is to try to decrease this dimension, of course without losing relevant information in the data. Dimension reduction is used as preprocessing, before applying data analysis models on data with a lower dimension.

PCA (Principal Component Analysis) is the most traditional tool used for dimension reduction. PCA projects data on a lower-dimensional space, choosing axes keeping the maximum of the data initial variance. Unfortunately, PCA is a linear tool. Nonlinear relations between the components of the initial data may be lost in the preprocessing. If the goal is to further use nonlinear data analysis tools on the reduced data, one easily sees that the use of a linear preprocessing is not appropriate.

There is nowadays a huge research effort in developing nonlinear projection tools that do not suffer from the above limitation. Nonlinear projection means to find a lower-dimensional space in which the data as well are described as in the original space. This supposes that data lie on a sub-manifold in the original space. Ideally, there should be a bijection between this sub-manifold and the lower-dimensional space; the existence of a bijection is a proof that no information is lost in the transformation. Figure 5 shows an artificial example of nonlinear dimension reduction (from dimension 3 to 2, for illustration purposes). If curved axes as the ones shown on the left part of the figure could be found and defined in the initial data space, one could unfold the axes to find the lower dimensional representation as shown in the right figure.

There are several ways to design nonlinear projection methods. A first one consists in using PCA, but locally in restricted parts of the space [11]. Joining local linear models leads to a global nonlinear one; it has however the disadvantage of being not continuous, therefore of limited interest.

Kernel PCA [12] consists in first transforming the data into a higher-dimensional space, and then applying PCA on the transformed data. Kernel PCA benefits from the strong theoretical background of kernel methods, and reveals to be interesting in specific situations. However, the method suffers from a difficult choice of the initial transformation, and from the apparent contradiction to increase the dimension of the data before reducing it.

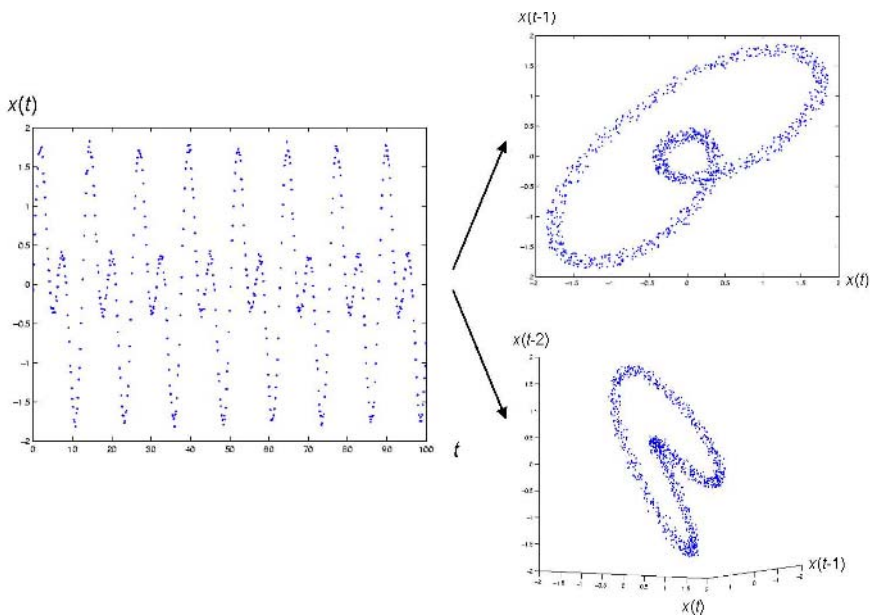


**Fig. 5.** 2-dimensional nonlinear projection of 3-dimensional horseshoe distribution

Distance preservation methods form a class of nonlinear projection tools that have interesting geometrical properties. The principle is to find a lower dimensional representation of data where the pairwise distances are respected as much as possible with respect to the original data space. Sammon's nonlinear mapping [13] belongs to this class of methods. Short distances in the original space are favored, to allow unfolding of large, nonlinear surfaces and volumes. Demartines and Herault's CCA (Curvilinear Component Analysis) [14] greatly improves the previous method by giving more weight to short distances in the projection space instead of the original one. This seemingly minor modification allows to cut varieties with loops, which are more than common in high dimensional spaces. Another important improvement in distance preservation methods consists in measuring the distances in the original space along the manifold, instead of taking the Euclidean distance between pairs of points; unfolding is then much facilitated. The Curvilinear Distance Analysis (CDA)

[15] and Isomap [16] methods, independently developed, belong to this category; contrarily to Isomap, CDA combines the advantages of the curvilinear measure and the larger weights on short distances, leading to efficient unfolding in a larger class of situations.

Other nonlinear projections tools must be mentioned too. Self-Organizing Maps (SOM) or Kohonen's maps [17] may be viewed as neighbor-preservation nonlinear projection tools. SOM are classically used in representation tasks, where the dimension of the projection space is limited to 3. However, there is no technical difficulty to extend the use of SOM to higher-dimensional projection spaces. SOM are used when a combination of vector quantization, clustering and projection is looked for. However, the quality of the bijection with the original space (no loss of information in the transformation) is limited compared to distance-preservation methods.



**Fig. 6.** Time series state space. Left: time series. Top right: 2-dimensional state space. Bottom right: 3-dimensional state space

Finally, the classical bottleneck MLP [18] also performs a nonlinear dimension reduction that is bijective by design. Despite its interesting concept, the bottleneck MLP suffers from its limitation to simple problems, because of the numerical difficulties to adjust the parameters of a MLP with many layers.

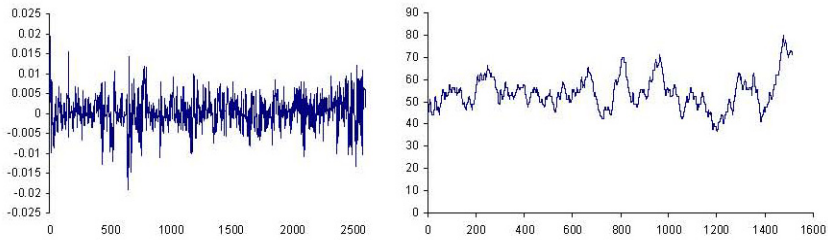
## 5 Time Series Prediction

Time series forecasting consists in predicting unknown values of a series, based on past, known values. Grouping past values into vectors (called regressors) makes time

series forecasting an almost standard function approximation problem (see Figure 1 right). Naturally, because of the non random character of the time series, dependencies exist between the coordinates of the regressors. The situation is thus a typical where dimension reduction should be possible, leading to improved prediction performances. Takens' theorem [19] provides a strong theoretical background for such dimension reduction.

### 5.1 Takens' Theorem

Let us first define a regressors state space as illustrated in Figure 6. The left part of the figure shows an artificial time series (that is obviously easy to predict). The right part shows state spaces of regressors formed by two (top) and three (bottom) consecutive values of the series. In these regressors spaces, it is possible to see that the data occupy a low-dimensional part of the space; this dimensionality is called the intrinsic dimension of the regressors (the intrinsic dimension is 1 in the illustrated example, as regressors follow a line), and may be estimated for example by using Grassberger-Procaccia's method [20].



**Fig. 7.** BEL20 financial index. Left: daily returns over approximately 7 years; right: 90-days averaged prediction over test set of the daily return sign

Takens' theorem expresses two properties regarding the regressors space and its intrinsic dimension:

- First, if  $q$  is the intrinsic dimension of the regressors (estimated in a sufficiently large-dimensional state space), then the size of the regressors to be used to predict the series is between  $q$  and  $2q+1$ . In other words, more than  $2q+1$  values in the regressors do not carry supplementary information useful to predict the series.
- Secondly, the regressors in the  $2q + 1$ -dimensional space may be projected without loss of information in a  $q$ -dimensional space.

As in most time series prediction problems the optimal size of regressors is difficult to know a priori, Takens' theorem provides a way to estimate it. The prediction model will then be developed on a minimal but sufficient number of variables.

## 5.2 BEL20 Prediction Example

Figure 7 shows an example of the application of the above methodology on the problem of predicting the Belgium BEL20 financial stock market index [21]. The left part of the figure shows the daily returns (relative variations) of the BEL20 index. According to standard procedures in stock market index forecasting [22], 42 indicators are built from the series (returns, averages of returns, moving averages, etc.). By design, many of these indicators are dependent or even correlated.

A linear PCA is applied to first reduce the the dimensionality of the regressors. Keeping 99% of the variance leads to a reduced set of 25 compound indicators. Grassberger-Procaccia's procedure is used to estimate the intrinsic dimensionality of the regressors, which is found to be approximately 9. Then, the 25-dimensional regressors resulting from the PCA are further projected in a 9-dimensional space, using the Curvilinear Component Analysis algorithm. Finally, a Radial-Basis Function Network is built on the 9-dimensional vectors to predict the next value of the BEL20 daily return.

Unsurprisingly, it is extremely difficult to obtain very good predictions in such problem! Nevertheless, if the goal is restricted to predict the sign of the next returns (which means to predict if the index will increase or decrease), the results are not so bad. Figure 7 (right) shows percentage of good predictions of the sign, averaged over 90 days. Numerically, the percentage of success in the correct approximation of the sign is 57%, i.e. 7% more than a pure random guess.

## 6 Conclusion

High-dimensional spaces show surprising geometrical properties that are counter intuitive with respect to the behavior of low-dimensional data. Among these properties, the concentration of norm phenomenon has probably the most impact on the design on data analysis tools. Its consequences are among others that standard Euclidean norms may become unselective in high-dimensional spaces, and that the Gaussian kernels, commonly used in many tools, become inappropriate too. Suggestions to overcome these consequences are presented. Another direction to follow is to reduce the dimensionality of the data space, through appropriate nonlinear data projection methods. The methodology is illustrated in the context of time series forecasting, on the BEL20 stock market index prediction problem.

## References

1. Ramsay, J.O., Silverman, B.W.: *Functional Data Analysis*. Springer (1997)
2. Borggaard, C., Thodberg, H.: Optimal minimal neural interpretation of spectra. *Analytical Chemistry* 64 (1992) 545–551
3. Bellmann, R.: *Adaptive Control Processes: A Guided Tour*. Princeton University Press (1961)
4. Silverman, B.: *Density estimation for statistics and data analysis*. Chapman and Hall (1986)
5. Demartines, P.: *Analyse de donnée par réseaux de neurones auto-organisées*. Ph.D. dissertation (in French), Institut National Polytechnique de Grenoble (France) (1994)



6. Beyer, K.S., Goldstein, J., Ramakrishnan, R., Shaft, U.: When is “nearest neighbor” meaningful? In Beeri, C., Buneman, P., eds.: Database Theory -ICDT '99, 7th International Conference, Jerusalem, Israel, January 10-12, 1999, Proceedings. Volume 1540 of Lecture Notes in Computer Science., Springer (1999) 217–235
7. Hinneburg, A., Aggarwal, C., Keim, D.: What is the nearest neighbor in high dimensional spaces? In: The VLDB Journal. (2000) 506–515
8. Aggarwal, C.C., Hinneburg, A., Keim, D.A.: On the surprising behavior of distance metrics in high dimensional spaces. In den Bussche, J.V., Vianu, V., eds.: Database Theory -ICDT 2001, 8th International Conference, London, UK, January 4-6, M. Verleysen, D. François 2001, Proceedings. Volume 1973 of Lecture Notes in Computer Science., Springer (2001) 420–434
9. François, D., Wertz, V., Verleysen, M.: Non-euclidean metrics for similarity search in noisy datasets. In: ESANN2005, European Symposium on Artificial Neural Networks (Accepted), Bruges (Belgium) (2005)
10. François, D., Wertz, V., Verleysen, M.: On the locality of kernels in highdimensional spaces. In: ASMDA2005, Applied Stochastic Models and Data Analysis (Submitted on invitation), Brest (France) (2005)
11. Kambhatla, N., Leen, T.: Dimension reduction by local principal component analysis. *Neural Computation* 9 (1997) 1493–1516
12. Schüller, K.R.: Nonlinear component analysis as a kernelolkopf, B., Smola, A., M' eigenvalue problem. *Neural Computation* 10 (1998) 1299–1319
13. Sammon, J.: A nonlinear mapping for data structure analysis. *IEEE Transactions on Computers* 18 (1969) 401–409
14. Demartines, P., Herault, J.: Curvilinear component analysis: A self-organizing neural network for nonlinear mapping of data sets. *IEEE Transaction on Neural Networks* 8 (1997)
15. Lee, J.A., Lendasse, A., Donckers, N., Verleysen, M.: Curvilinear distance analysis versus isomap. In: ESANN 2002 European Symposium on Artificial Neural Networks, Bruges (Belgium) (2002) 185–192
16. Tenenbaum, J.B., de Silva, V., Langford, J.C.: A global geometric framework for nonlinear dimensionality reduction. *Science* 290 (2000) 2319–2313
17. Kohonen, T.: *Self-Organizing Maps*. Springer, Berlin, Heidelberg (1995)
18. Kramer, M.: Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal* 37 (1991) 233
19. Takens, F.: On the numerical determination of the dimension of an attractor. *Lectures Notes in Mathematics* 1125 (1985) 99–106
20. Grassberger, P., Procaccia, I.: Measuring the strangeness of strange attractors. *Physica D* 9 (1983) 189–208
21. Lendasse, A., Lee, J., Bodt, E.D., Wertz, V., Verleysen, M.: Dimension reduction of technical indicators for the prediction of financial time series -application to the bel 20 market index. *European Journal of Economic and Social Systems* 15 (2001)
22. Refenes, A., Burgess, A., Bentz, Y.: Neural networks in financial engineering: A study in methodology. *IEEE Transactions on Neural Networks* 8 (1997) 1222–1267

# Obtaining a Complex Linguistic Data Summaries from Database Based on a New Linguistic Aggregation Operator

Zheng Pei<sup>1</sup>, Yajun Du<sup>1</sup>, Liangzhong Yi<sup>1</sup>, and Yang Xu<sup>2</sup>

<sup>1</sup> School of Computers & Mathematical-Physical Science,  
Xihua University, Chengdu, Sichuan, 610039, China  
pqyz@263.net

<sup>2</sup> Department of Mathematics, Southwest Jiaotong University,  
Chengdu, Sichuan, 610031, China

**Abstract.** In real-world database, generally, attribute values of objects are numerical, from real-world practice point of view, numeral is too detail to obtaining good information or decision. Hence, a linguistic data summary of a set of data, which is expressed by a sentence or a small number of sentences in a natural language, would be very desirable and human consistent. In this paper, from the structure and valuation of fuzzy statement point of view, extracting linguistic data summarize is discussed. To extract complex linguistic data summaries, a new aggregation operator for aggregating linguistic terms is proposed, a numerical example of Personnel Database is also provided.

## 1 Introduction

How to process perception based information that is described in natural languages is a problem. The methodology of computing with words (*CWW*) proposed by Zadeh may be viewed as an attempt to harness the highly expressive power of natural languages by developing ways of *CWW* or propositions drawn from a natural language[1]-[3]. Based on difference background, many researchers have study *CWW* [1], [4]-[9]. On the other hand, for a database, the raw data are often not useful and do not provide “knowledge.” Knowledge discovery from a database is not a trivial act that requires some intelligent techniques, many important methods and results have been proposed to extract linguistic data summaries from numerical database [10]-[19].

## 2 Linguistic Data Summary

In a database of workers, let  $V$  be a quality (attribute) of interest, *e.g.*, age, salary, etc;  $Y = \{y_1, y_2, \dots, y_n\}$  is a set of objects that manifest quality  $V$ ;  $V(y_i)$  ( $i = 1, \dots, n$ ) are values of quality  $V$  for each object  $y_i$ ;  $D = \{V(y_i) | i = 1, \dots, n\}$  is a set of data. A linguistic data summary of a set of data consists of

[10]: (1) summarizer  $S$ , e.g., young; (2) a quantity in agreement  $Q$ , e.g., most; (3) truth  $T$ . And a linguistic data summary can be formalized by  $Qy's\ are\ S\ is\ T$ , in which  $Q$  is a fuzzy linguistic quantifier,  $Y = \{y_i | i = 1, \dots, n\}$  is a set of objects,  $S$  is a summarizer (property), and  $T$  is a truth degree, e.g.,

$$\text{most } (Q) \text{ of employees } (y's) \text{ are young } (S) \text{ is } T. \tag{1}$$

Many researchers used  $T(are(Qy's, S)) = a, a \in [0, 1]$  to express the valuation [16]-[18]. In this paper, we always assume that  $Q, S$  and  $T$  consist of linguistic terms, respectively, i.e.,  $Q = \{q_1, \dots, q_m\}, S = \{s_1, \dots, s_l\}$  and  $T = \{t_1, \dots, t_k\}$ , for each  $q_{m'} \in Q, s_{l'} \in S$ , and  $t_{k'} \in T$  are linguistic terms, respectively, e.g., *most, young, and very true*. Due to each summarizer  $s_{l'} \in S$  is the property of objects (records)  $Y, s_{l'}$  can be represented by a fuzzy set on  $D = \{V(y_i) | i = 1, \dots, n\}$ . e.g., if  $s_{l'}$  is *young* and  $D_{s_{l'}} = \{1, \dots, 90\}$ , then  $s_{l'}$  is represented by  $\mu_{s_{l'}} : D_{s_{l'}} \rightarrow [0, 1]$ . In this paper, suppose each summarizer  $s_{l'} \in S$  has a fuzzy set on  $D_{s_{l'}}$ . Fuzzy linguistic quantifier  $q_{m'} \in Q$  and fuzzy linguistic truth degree  $t_{k'} \in T$  are defined as follows [15]:

- Let  $P(Y) = \{A | A \subseteq Y\}$  be the power set of  $Y$ . Define a binary relation on  $P(Y)$ :  $A \sim B$  if and only if  $|A| = |B|$ , where  $|A|$  is the cardinal number of  $A$ . Obviously, “ $\sim$ ” is an equivalence relation on  $P(Y)$ . The factor set of  $P(Y)$  by  $\sim$  is denoted by  $\bar{P}(Y) = P(Y) / \sim$ .
- For each fuzzy linguistic quantifier  $q_{m'} \in Q$ , its fuzzy set is defined by

$$\mu_{q_{m'}} : \bar{P}(Y) \rightarrow [0, 1]. \tag{2}$$

- For each fuzzy linguistic truth degree  $t_{k'} \in T$ , its fuzzy set is defined by

$$\mu_{t_{k'}} : [0, 1] \rightarrow [0, 1]. \tag{3}$$

*Example 1.* The membership functions of several fuzzy linguistic quantifiers are

$$\begin{aligned} \mu_{\exists}(A) &= \begin{cases} 1, & \text{if } |A| \geq 1, \\ 0, & \text{if } A = \emptyset. \end{cases} & \mu_{\forall}(A) &= \begin{cases} 1, & \text{if } A = Y, \\ 0, & \text{otherwise.} \end{cases} \\ \mu_{\text{most}}(A) &= \begin{cases} 1, & \text{if } |A| \geq \alpha, \\ \frac{|A| - \beta}{\alpha - \beta}, & \text{if } \beta \leq |A| \leq \alpha, \\ 0, & \text{if } |A| \leq \beta. \end{cases} & \mu_{\text{about}3}(A) &= \begin{cases} \frac{|A| - 1}{2}, & \text{if } 1 \leq |A| \leq 3, \\ \frac{5 - |A|}{2}, & \text{if } 3 \leq |A| \leq 5, \\ 0, & \text{otherwise.} \end{cases} \end{aligned}$$

In which  $\alpha, \beta$  such that  $\alpha, \beta \in (0, |Y|)$ , it is decided by experts or deciders.

### 3 Extracting a Simple Linguistic Data Summary

Let a simple linguistic data summary “ $Qy's\ are\ S\ is\ T$ ,” then  $Q, S$  and  $T$  can be obtained automatically by the following steps [15],

- Fixing a summarizer  $s_{l'} \in S$  (it can be one or several) and a level (threshold)  $\theta$ , this can be done by experts or deciders. Let

$$D_{s_{l'}}^\theta = \mu_{s_{l'}}^{-1}(V(y_i)) = \{V(y_i) | \mu_{s_{l'}}(V(y_i)) \geq \theta\}. \tag{4}$$

- Selecting a fuzzy linguistic quantifier  $q_{m'} \in Q$ . According to Eq.(4),  $q_{m'}$  can be selected such that

$$\mu_{q_{m'}}(A) = \max\{\mu_{q_1}(A), \mu_{q_2}(A), \dots, \mu_{q_m}(A)\}, \tag{5}$$

in which  $A = \{y_i | V(y_i) \in D_{S_{i'}}^\theta\}$ . In the real world practice, maybe  $\mu_{q_{m'}}(A)$  is not only one, in this case, one  $q_{m'}$  can be selected by deciders or the indexes of linguistic terms, *e.g.*,  $q_{m'}$  has a maximal index.

- Selecting a fuzzy linguistic truth degree  $t_{k'} \in T$ . From the viewpoint of logic, the more objects satisfying statement with the quantifier, the higher the truth degree. On the other hand, the bigger  $\mu_{q_{m'}}(A)$  is, the more objects is satisfying the statement with the quantifier. Hence,  $t_{k'}$  can be selected as:

$$\mu_{t_{k'}}(\mu_{q_{m'}}(A)) = \max\{\mu_{t_1}(\mu_{q_{m'}}(A)), \mu_{t_2}(\mu_{q_{m'}}(A)), \dots, \mu_{t_k}(\mu_{q_{m'}}(A))\}. \tag{6}$$

*Example 2.* Table 1 is a personnel database. In which, the employee  $y_1$  is 25 years old and earns 1.8 thousand dollar a year.

**Table 1.** Personnel Database

| $V \setminus Y$ | $y_1$    | $y_2$    | $y_3$    | $y_4$    | $y_5$    | $y_6$    | $y_7$    | $y_8$    | $y_9$    | $y_{10}$ | $y_{11}$ | $y_{12}$ |
|-----------------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|
| <b>age</b>      | 25       | 48       | 31       | 35       | 28       | 51       | 37       | 43       | 34       | 27       | 53       | 45       |
| <b>salary</b>   | 1.8      | 2.0      | 2.8      | 3.0      | 2.8      | 3.0      | 2.3      | 2.5      | 3.5      | 2.9      | 3.0      | 3.1      |
| $\vdots$        | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ |

Let  $S_{\text{age}} = \{\text{young}(y), \text{middle age}(ma)\}$ ,  $S_{\text{salary}} = \{\text{low}(l), \text{high}(h)\}$ . Select  $Q = \{\text{several}(s), \text{about half}(ah), \text{most}(m)\}$ ,  $T = \{\text{approximately true}(at), \text{true}(t), \text{very true}(vt)\}$ . Their membership functions are as follows:

$$\begin{aligned} \mu_y(x) &= \begin{cases} \frac{1}{10}x - 2, & \text{if } x \in [20, 30], \\ 4 - \frac{1}{10}x, & \text{if } x \in (30, 40], \\ 0, & \text{if } x > 40. \end{cases} & \mu_{ma}(x) &= \begin{cases} 5.5 - \frac{1}{10}x, & \text{if } x \in (45, 55], \\ \frac{1}{10}x - 3.5, & \text{if } x \in (35, 45], \\ 0, & \text{if } x \leq 35. \end{cases} \\ \mu_l(x) &= \begin{cases} 2(x - 1.5), & \text{if } x \in [1.5, 2], \\ 2(2.5 - x), & \text{if } x \in (2, 2.5], \\ 0, & \text{if } x \geq 2.5. \end{cases} & \mu_h(A) &= \begin{cases} 4.3 - x, & \text{if } x \in [3.3, 4.3], \\ x - 2.3, & \text{if } x \in [2.3, 3.3], \\ 0, & x < 2.3. \end{cases} \\ \mu_s(A) &= \begin{cases} \frac{1}{2}(|A| - 1), & \text{if } |A| \in [1, 3], \\ 2 - \frac{1}{3}|A|, & \text{if } |A| \in (3, 6], \\ 0, & \text{if } |A| > 6. \end{cases} & \mu_{ah}(A) &= \begin{cases} 0, & \text{if } |A| < 4 \\ \frac{1}{2}|A| - 2, & \text{if } |A| \in [4, 6], \\ 2.5 - \frac{1}{4}|A|, & \text{if } |A| \in (6, 10], \\ 0, & \text{if } |A| > 10. \end{cases} \\ \mu_m(A) &= \begin{cases} \frac{1}{4}(|A| - 10), & \text{if } |A| \in [10, 14], \\ 0, & \text{if } |A| < 10. \end{cases} & \mu_{at}(x) &= \begin{cases} 2(1 - x), & \text{if } x \in [0.5, 1], \\ 2x, & \text{if } x \in [0, 0.5]. \end{cases} \\ \mu_t(x) &= \begin{cases} 5(1 - x), & \text{if } x \in [0.8, 1], \\ \frac{10}{3}(x - 0.5), & \text{if } x \in [0.5, 0.8], \\ 0, & \text{if } x \in [0, 0.5]. \end{cases} & \mu_{vt}(x) &= \begin{cases} 5x - 4, & \text{if } x \in [0.8, 1], \\ 0, & \text{if } x \in [0, 0.8]. \end{cases} \end{aligned}$$

(1) Fixing a summarizer  $s' = young \in S_{age}$  and  $s'' = high \in S_{salary}$ . Let threshold  $\theta = 0.5$ , then according to  $\mu_y$  and  $\mu_h$ , we obtain

$$D_{s'}^{0.5} = \{V(y_i) | \mu_{s'}(V(y_i)) \geq 0.5\} = \{25, 31, 35, 28, 34, 27\}, \tag{7}$$

$$D_{s''}^{0.5} = \{V(y_i) | \mu_{s''}(V(y_i)) \geq 0.5\} = \{2.8, 3.0, 3.5, 2.9, 3.1\}, \tag{8}$$

$$A_{s'} = \{y_i | V(y_i) \in D_{s'}^{0.5}\} = \{y_1, y_3, y_4, y_5, y_9, y_{10}\}, \tag{9}$$

$$A_{s''} = \{y_i | V(y_i) \in D_{s''}^{0.5}\} = \{y_3, y_4, y_5, y_6, y_9, y_{10}, y_{11}, y_{12}\}, \tag{10}$$

(2) According to  $\mu_s, \mu_{ah}, \mu_m$  and Eq.(9), we obtain

$$\begin{aligned} \mu_s(A_{s'}) &= 0, \mu_{ah}(A_{s'}) = 1, \mu_m(A_{s'}) = 0, \\ \max\{\mu_s(A_{s'}), \mu_{ah}(A_{s'}), \mu_m(A_{s'})\} &= \mu_{ah}(A_{s'}), \end{aligned} \tag{11}$$

according to  $\mu_{at}, \mu_t, \mu_{vt}$  and Eq.(11), we obtain

$$\mu_{at}(\mu_{ah}(A_{s'})) = \mu_t(\mu_{ah}(A_{s'})) = 0, \mu_{vt}(\mu_{ah}(A_{s'})) = 1, \tag{12}$$

Hence the fuzzy linguistic truth degree is  $\mu_{vt}(\mu_{ah}(A_{s'}))$ , according to Eq.(11) and Eq.(12), we obtain the following linguistic data summary:

$$\text{about half of employees are young is very true.} \tag{13}$$

(3) Similar (2), we also obtain the following linguistic data summary about salary:

$$\text{most of employees have high salary is approximately true.} \tag{14}$$

## 4 A New Linguistic Aggregation Operator

### 4.1 Fuzziness of the Index of Linguistic Label

From information granule (IG) point of view[3], each linguistic label is a fuzzy IG which is a clump of physical or mental objects (points) drawn together by indistinguishability, similarity, proximity or functionality. If a granule is crisp, then using the natural number index of the granule is equal to using the granule itself, *e.g.*, 5th month of a year (May). However, if a granule is fuzzy, then there exists problem in using the natural number index of the granule, *e.g.*, let  $G$  be a peoples set, 1 denotes “young people” of  $G$ , 2 denotes “old people” of  $G$ , in this case, a people  $g \in G$  is in 1 (or 2) can not be decided due to their boundaries are not sharply. From algebra viewpoint, let  $X \subseteq R$  be a universal set. All granules on  $X$  are denoted by  $P(X)$ ,  $R^+ = \{x|x \geq 0\}$ , define a linear function  $F : X \longrightarrow R^+$ .

Let  $\mu_A$  be a membership function of granule  $A \in P(X)$ .  $\tilde{F}(X)$  the collection of membership functions on  $X$  such that  $\forall \mu_A(x) \in \tilde{F}(X)$ ,  $\mu_A(x)$  a membership function of a granule  $A \in P(X)$ . Define

$$G : \tilde{F}(X) \longrightarrow P(X) \tag{15}$$

Then an equivalence relation “ $\simeq$ ” on  $\tilde{F}(X)$  is obtained,  $\mu_A(x) \simeq \mu_B(x) \leftrightarrow G(\mu_A(x)) = G(\mu_B(x))$ , i.e.,  $\mu_A(x)$  and  $\mu_B(x)$  are membership functions of the same granule  $A$ , denoted  $[\mu_A(x)] \in \tilde{F}(X)/\simeq$ . Select representative element  $\mu_A(x)$  of  $[\mu_A(x)]$ , by using  $F$  and extension principle, one-to-one mapping is obtained:  $E : \tilde{F}(X)/\simeq \rightarrow D = \{\chi | \chi : R^+ \rightarrow [0, 1], E([\mu_A(x)]) = E(\mu_A(x)) = \chi\}$  Due to  $G$  is a surjection from  $\tilde{F}(X)$  to  $P(x)$ , there exists one-to-one mapping  $\tau$  between  $\tilde{F}(X)/\simeq$  and  $P(x)$ , and one-to-one mapping  $\kappa$  between  $P(x)$  and  $D$ . Therefor,  $\chi$  can be selected as the index of granule  $A$ .

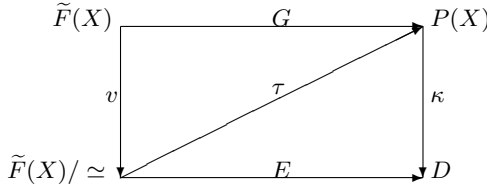


Fig. 1. Commutativity of a diagram of maps between  $P(x)$  and  $D$

In Fig.1,  $v$  is the natural map of  $\tilde{F}(X)$  to the quotient set  $\tilde{F}(X)/\simeq$ ,  $E$  is extension principle,  $\kappa = E\tau^{-1}$ . If granule  $A$  is crisp, then  $\mu_A(x)$  is the characteristic function of classical set  $A$ , and  $\chi$  is also a characteristic function. Using the natural numbers theory, the indexes of granules is the natural numbers, i.e.,  $E(A \in X/\simeq) = i \in R^+$ . If granule  $A$  is fuzzy, according to extension principle,  $E([\mu_A(x)]) = \chi$  is a fuzzy number on  $R^+$ . Let  $\mu_A$  be triangular membership function of fuzzy granule  $A$ , then  $\chi$  is a triangular fuzzy number  $\tilde{i}$ , where  $\mu_A(F^{-1}(i)) = 1$ . Therefor, indexes of granules are fuzzy numbers instead of natural numbers.

*Example 3.* Let linguistic labels  $young = (30, 10, 10)$ ,  $middle-age = (45, 10, 10)$ , are triangular fuzzy sets on  $[20, 55]$ , in which components are the center, left width and right width, respectively. Linear function  $F : [20, 55] \rightarrow [0, 1]$  is  $F(x) = \frac{1}{35}x - \frac{4}{7}$ , correspondingly, indexes of linguistic labels are  $young_{\frac{2}{7}} = young_{(\frac{2}{7}, \frac{2}{7}, \frac{2}{7})}$  and  $middle-age_{\frac{5}{7}} = middle-age_{(\frac{5}{7}, \frac{2}{7}, \frac{2}{7})}$ .

### 4.2 $F_{lowa}$ Operator

**Definition 1.** Let  $S = \{s_i\}$  and  $|S| = T$  (the cardinal number of  $S$ ),  $A = \{a_{\tilde{j}_1}, a_{\tilde{j}_2}, \dots, a_{\tilde{j}_n}\} \subseteq S$  be a set of labels to be aggregated ( $n \leq T$ ).  $W = \{w_1, w_2, \dots, w_n\}$  is a weighting vector such that  $\forall s \in \{1, \dots, n\}, w_s \in [0, 1]$  and  $\sum_{s=1}^n w_s = 1$ . Let  $B = (j_1, j_2, \dots, j_n)$ , where  $j_s$  is the center of  $\tilde{j}_s$ ,  $C = \sigma(B) = (j_{\sigma(1)}, \dots, j_{\sigma(n)})$  such that  $j_{\sigma(s')} \geq j_{\sigma(s)}, \forall s' \leq s$ , denote

$$w = f_{owa}(B) = WC^T = \sum_{s=1}^n w_s j_{\sigma(s)},$$

then the new linguistic aggregation operator  $F_{lowa}$  is defined as

$$F_{lowa}((a_{\tilde{j}_1}, a_{\tilde{j}_2}, \dots, a_{\tilde{j}_n})) = a_{\tilde{j}_k},$$

where  $a_{\tilde{j}_k}$  such that  $\tilde{j}_k(w) = \max\{\tilde{j}_1(w), \dots, \tilde{j}_T(w)\} \mid S \models T$ .

*Remark 1.* Due to  $a_{\tilde{j}_s} \iff \tilde{j}_s$ , therefore, aggregating linguistic labels  $\{a_{\tilde{j}_s} \mid s = 1, \dots, n\}$  is transformed by aggregating the indexes of the linguistic labels.

*Example 4.* Let  $Q = \{\text{several}(s), \text{about half}(ah), \text{most}(m)\}$ ,  $T = \{\text{approximately true}(at), \text{true}(t), \text{very true}(vt)\}$ , their membership function and the index are in Table 2.

**Table 2.** Linguistic labels, membership functions and the indexes

|   | number | Meaning | Membership function | The index                                                               |
|---|--------|---------|---------------------|-------------------------------------------------------------------------|
| Q | 1      | s       | (3, 2, 3)           | $s_{\frac{4}{13}} = s_{(\frac{4}{13}, \frac{4}{13}, \frac{6}{13})}$     |
|   | 2      | ah      | (6, 2, 4)           | $ah_{\frac{10}{13}} = ah_{(\frac{10}{13}, \frac{4}{13}, \frac{8}{13})}$ |
|   | 3      | m       | (14, 4, 0)          | $m_{\bar{2}} = m_{(2, \frac{8}{13}, 0)}$                                |
| T | 1      | at      | (0.5, 0.5, 0.5)     | $at_{\bar{1}} = at_{(1,1,1)}$                                           |
|   | 2      | t       | (0.8, 0.3, 0.2)     | $t_{1.6} = t_{(1.6, 0.6, 0.4)}$                                         |
|   | 3      | vt      | (1, 0.2, 0)         | $vt_{\bar{2}} = vt_{(2, 0.4, 0)}$                                       |

For Q, select  $y = F(x) = \frac{2}{13}x - \frac{2}{13}$ . For T, select  $y = F(x) = 2x$ . Suppose the weighting vector  $W = (0.3, 0.4, 0.3)$ , to aggregate Q by using  $F_{lowa}$  operator, we obtain  $w_1 = 0.3 \times \frac{4}{13} + 0.4 \times \frac{10}{13} + 0.3 \times 2 = 1$ ,  $ah_{\frac{10}{13}}(1) = \max\{s_{\frac{4}{13}}(1), ah_{\frac{10}{13}}(1), m_{\bar{2}}(1)\}$ ,  $F_{lowa}((s, ah, m)) = ah_{\frac{10}{13}} = \text{about half}$ .

**Theorem 1.** (Commutativity) Let  $F_{lowa}$ , then

$$F_{lowa}((a_{\tilde{j}_1}, a_{\tilde{j}_2}, \dots, a_{\tilde{j}_n})) = F_{lowa}((a_{\tilde{j}_{\sigma(1)}}, a_{\tilde{j}_{\sigma(2)}}, \dots, a_{\tilde{j}_{\sigma(n)}})).$$

Where  $(\sigma(1), \dots, \sigma(n))$  is a permutation of  $(1, \dots, n)$ .

**Theorem 2.** (Boundedness) Let  $W_{\star} = (0, \dots, 0, 1)$ ,  $W^{\star} = (1, 0, \dots, 0)$ ,  $T$  ( $t$ -norm) and  $S$  ( $co$ - $t$ -norm), then

- $\min\{j_1, \dots, j_n\} = W_{\star}C^T \leq f_{owa}((j_1, \dots, j_n)) \leq W^{\star}C^T = \max\{j_1, \dots, j_n\}$
- $\min\{a_{\tilde{j}_1}, a_{\tilde{j}_2}, \dots, a_{\tilde{j}_n}\} \leq F_{lowa}((a_{\tilde{j}_1}, a_{\tilde{j}_2}, \dots, a_{\tilde{j}_n})) \leq \max\{a_{\tilde{j}_1}, a_{\tilde{j}_2}, \dots, a_{\tilde{j}_n}\}$
- $T(a_{\tilde{j}_1}, \dots, a_{\tilde{j}_n}) \leq F_{lowa}(a_{\tilde{j}_1}, \dots, a_{\tilde{j}_n}) \leq S(a_{\tilde{j}_1}, \dots, a_{\tilde{j}_n})$

*Remark 2.* Based on OWA operator, these conclusion are proved easily. In definition 1, let  $f_{owa}^{\star}((\tilde{j}_1(w), \dots, \tilde{j}_T(w))) = W^{\star}C^T$ , then  $\tilde{j}_k(w) = \max\{\tilde{j}_1(w), \dots, \tilde{j}_T(w)\} = f_{owa}^{\star}((\tilde{j}_1(w), \dots, \tilde{j}_T(w)))$ . From this point of view, the new linguistic aggregation operator  $F_{lowa}$  can be understood as a OWA operator, differs from OWA operator, two times OWA operator is used in  $F_{lowa}$ .

### 4.3 Obtaining a Complex Linguistic Data Summary Based on $F_{lowa}$

A complex linguistic data summary can be obtained based on  $F_{lowa}$ , *i.e.*,

1. Extracting simple linguistic data summaries by using method of Section 3;
2. Fuzziness of indexes of linguistic labels by using method of Subsection 4.1;
3. Selecting weighting vector  $W$  by using the following fuzzy linguistic quantifier [11],  $a, r, b \in [0, 1]$ ,

$$Q(r, a, b) = \begin{cases} 0, & \text{if } r < a, \\ \frac{r-a}{b-a}, & \text{if } a \leq r < b, \\ 1, & \text{if } r \geq b. \end{cases} \tag{16}$$

4. Aggregating simple linguistic data summaries by using  $F_{lowa}$ .

*Example 5.* Continue Example 2, aggregate *about half of employees are young is very true* and *most of employees have high salary is approximately true*. According to Table 2 and Eq.(16) ( $(a, b) = (0.2, 0.8)$ ),  $W = (0.6, 0.4)$  can be obtained and

$$w_1 = 0.6 \times \frac{10}{13} + 0.4 \times 2 = \frac{16.4}{13}, w_2 = 0.6 \times 1 + 0.4 \times 2 = 1.4, \\ ah_{\frac{10}{13}}(\frac{16.4}{13}) = max\{s_{\frac{4}{13}}(\frac{16.4}{13}), ah_{\frac{10}{13}}(\frac{16.4}{13}), m_{\frac{2}{13}}(\frac{16.4}{13})\}, \\ F_{lowa}(\text{about half, most}) = \text{about half.} \tag{17}$$

$$t_{\frac{1}{6}}(1.4) = max\{at_{\frac{1}{6}}(1.4), t_{\frac{1}{6}}(1.4), vt_{\frac{2}{6}}(1.4)\}, \\ F_{lowa}(\text{approximately true, very true}) = \text{true.} \tag{18}$$

According to Eq.(17) and Eq.(18), the complex linguistic data summary (*about half of employees are young and have high salary*) *is true* can be obtained.

## 5 Conclusion

In this paper, based on the structure and semantics of predicate of fuzzy logic, automatic extracting simple linguistic data summaries from personnel database is discussed. To obtain complex linguistic data summaries, a new linguistic aggregation operator is presented.

This work is supported by the National Natural Science Foundation of China (Grant No. 60474022).

## References

1. P. P. Wang, “ Computing with words”, John Wiley and Sons, 2001.
2. L. A. Zadeh, “ Fuzzy logic = computing with words”, *IEEE Trans. Fuzzy Systems*, Vol. 4, pp. 103–111, 1996.



3. L. A. Zadeh, "Toward a theory of fuzzy information granulation and its centrality in human reasoning and fuzzy logic", *Fuzzy Sets and Systems*, Vol. 90, pp. 103–111, 1997.
4. F. Herrera and E. Herrera-Viedma, "Aggregation operators for linguistic weighted information," *IEEE Trans. System, Man, Cybernet.-Part A: Systems Humans*, Vol. 27, pp. 646–656, 1997.
5. Zheng Pei and Yang Xu, "Lattice implication algebra model of linguistic variable Truth and its inference", in edited by Da Ruan et al, *Applied Computational Intelligence*, World Scientific, pp. 93–98, 2004.
6. Zheng Pei, Keyun Qin, "Obtaining Decision Rules and Combining Evidence based on modal logic", *Progress in Natural Science (in chinese)*, Vol. 14(5), pp. 501-508, 2004.
7. F. Herrera, E. Lopez and M. A. Rodriguez, "A linguistic decision model for promotion mix management solved with genetic algorithms," *Fuzzy Sets and Systems*, Vol. 131, pp. 47–61, 2002.
8. Christer Carlsson and Robert Fullér, "Benchmarking in linguistic importance weighted aggregations", *Fuzzy Sets and Systems*. **114** (2000) 35–41.
9. N. C. Ho, T. D. Khang, V. N. Huynh and H. C. Nguyen, "Hedge algebras, linguistic valued logic and their application to fuzzy reasoning", *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, Vol. 7, pp. 347–361, 1999.
10. R.R. Yager, "A new approach to the summarization of data", *Information Sciences*. Vol. 28, pp. 69-86, 1982.
11. R.R. Yager, "On ordered weighted averaging aggregation operators in multicriteria decision making", *IEEE Trans. Systems, Man Cybernet.* **18** (1988) 183–190.
12. R.R. Yager, "Database discovery using fuzzy sets", *International Journal of Intelligent Systems*, Vol. 11, pp.691-712, 1996.
13. R.R. Yager, J. Kacprzyk, "Linguistic data summaries: A perspective", in *Proceedings of IFSA '99 Congress (Taipei, Taiwan R.O.C.)*, vol. 1, pp. 44-48, 1999.
14. G. Raschia, N. Mouaddib, "SAINTETIQ: a fuzzy set-based approach to database summarization", *Fuzzy Sets and Systems*, Vol. 129, pp. 137-162, 2002.
15. Zheng Pei, Yang Xu, Da Ruan, Keyun Qin, "Extracting a complex linguistic data summaries from personnel database via linguistic aggregation of simple ones", *Information Sciences on Linguistic Decision Making: Tools and Applications. (to appear)*
16. J. Kacprzyk, S. Zadrożny, "Computing with words in intelligent database querying: standalone and Internet-based applications", *Information Sciences*, Vol. 34, pp. 71-109, 2001.
17. J. Kacprzyk, R.R. Yager, "Linguistic summaries of data using fuzzy logic", *International Journal of General Systems*, Vol. 30, pp. 133-154, 2001.
18. J. Kacprzyk, S. Zadrożny, "Linguistic summarization of data sets using association rules", in *Proceedings of FUZZ-IEEE03 (St. Louis, USA)*, Vol. 2, pp. 702-707, 2003.
19. Y.-Ch. Hu, R.-Sh. Chen, G.-H. Tzeng, "Mining fuzzy association rules for classification problems", *Computers and Industrial Engineering*, Vol. 43, pp. 735-750, 2002.

# Bias and Variance of Rotation-Based Ensembles<sup>\*</sup>

Juan José Rodríguez<sup>1</sup>, Carlos J. Alonso<sup>2</sup>, and Oscar J. Prieto<sup>2</sup>

<sup>1</sup> Lenguajes y Sistemas Informáticos,  
Universidad de Burgos, Spain  
jjrodriguez@ubu.es

<sup>2</sup> Grupo de Sistemas Inteligentes,  
Departamento de Informática,  
Universidad de Valladolid, Spain  
calonso@infor.uva.es

**Abstract.** In Machine Learning, ensembles are combination of classifiers. Their objective is to improve the accuracy. In previous works, we have presented a method for the generation of ensembles, named rotation-based. It transforms the training data set; it groups, randomly, the attributes in different subgroups, and applies, for each group, an axis rotation. If the used method for the induction of the classifiers is not invariant to rotations in the data set, the generated classifiers can be very different. In this way, different classifiers can be obtained (and combined) using the same induction method.

The bias-variance decomposition of the error is used to get some insight into the behaviour of a classifier. It has been used to explain the success of ensemble learning techniques. In this work the bias and variance for the presented and other ensemble methods are calculated and used for comparison purposes.

## 1 Introduction

One of the research areas in Machine Learning is the generation of ensembles. The basic idea is to use more than one classifier, in the hope that the accuracy will be better. It is possible to combine heterogeneous classifiers, where each of the classifiers is obtained with a different method. Nevertheless, it is also possible to combine homogeneous classifiers. In this case all the classifiers have been obtained with the same method. In order to avoid identical classifiers, it is necessary to change something, at least the random seed.

There are methods that alter the data set. *Bagging* [3] obtains a new data set by resampling the original data set. An instance can be selected several times, so some instances will not be present in the new data set. The *Random Subspaces* [8] method obtains a new data set deleting some attributes. *Boosting* [14] is a family of methods. The most prominent member is AdaBoost. In this case the data

---

<sup>\*</sup> This work has been supported by the Spanish MCyT project DPI2001-01809.

set is modified depending on the classification errors of the previously generated base classifier. The bad classified examples are assigned a greater weight, so the next classifier will give more importance to those examples. Another possibility, instead of modifying the data set, is to modify the base learning method. For instance, in the *Random Forest* method [4], decision trees are constructed in a way such as the selected decision for a node is, to some extent, random. Comparatives among ensemble generation methods are presented in [1, 5, 9].

These methods share the idea that it is necessary to modify the data set in a way that some information is lost (e.g., instances, attributes), or the learning method does not use all the information (i.e., random forest). None of the modifications would be considered if it was desired to obtain a unique classifier. They are used because ensemble methods need diverse base classifiers.

Rotation-based ensembles [13] transform the data set, but in a way that all the present information is preserved (although it is transformed). The idea is to group the attributes, and for each group to apply an axis rotation. Hence, all the available information (instance and attributes) in the data set is still available. Although there are a lot of learning methods that are invariant to rotations, one of the most used with ensembles, decision trees, are very sensitive to this kind of variations, because the generated decision surfaces are formed by parallels to the axis.

When the method was presented [13] it was compared with other ensemble methods using only the error. It is well known that the error of a classifier can be divided in bias and variance. In this work we calculate these measures for different ensemble methods and use them for comparison purposes.

The rest of the work is organized as follows. The proposed method is described in Sect. 2. The bias-variance decomposition of the error is presented in Sect. 3. Section 4 includes the experimental validation. Finally, section 5 concludes.

## 2 Rotation-Based Ensembles

This method is introduced in [13]. It is based on transforming the data set, in a different way for each member of the ensemble. Then, the base learning method is used with the transformed data set. The results of the different classifiers are combined using majority vote.

The transformation process is based on Principal Component Analysis (PCA) [7]. These components are linear combinations of attributes. Normally, this technique is used for reducing the dimensionality of the data set, because the components are ordered according to their importance, so the last ones can be discarded. Nevertheless, we are going to consider all the components. In this case, the transformed data set has exactly the same information than the original one, with the only difference of axis rotation. This technique works with numeric attributes. If the data set has discrete attributes, they must be transformed to numeric.

One of the objectives in ensemble methods is that the combined classifiers were diverse, because nothing is gained if they are equal. Another objective,

somewhat contradictory to the previous one, is that the combined classifiers were accurate. PCA is an adequate transformation for ensemble generation because, first, no information is lost, so the accuracy of the base classifiers should not be worsened, as happens with other ensemble methods. And second, because the model generated by the base classifier can be rather different, given that the method was not invariant to rotations.

Nevertheless, for a given data set, the result of PCA is unique. For generating different classifiers it is necessary to obtain several transformed data sets. The possibilities are:

- To apply PCA to a subset of the instances. Unless the number of instances were rather small, the results will be rather similar.
- To apply PCA to a subset of the classes. In fact, this is a variant of the previous possibility, because when classes are selected, their instances are being selected. Nevertheless, the hope is that the results of the analysis will be more diverse.
- To apply PCA to a subset of the attributes. In this case only the selected attributes would be rotated. In order to modify all the attributes, it is possible to group the attributes in groups, and to apply PCA for each group.

The previous strategies are considered for the application of PCA, but once that the analysis has been obtained, then all the data set is reformulated using the components. If the previous strategies are combined, it is possible, except for very small data sets, to obtain a lot of different data sets.

The application of PCA to groups of attributes is another mechanism for diversification, but it also provides additional advantages. First, the execution time of the analysis depends mainly on the number of attributes, so is much quicker to do the analysis in groups than doing it with all the attributes.

An algorithmic description of the method is presented in [13].

### 3 Bias-Variance Decomposition of the Error

The bias-variance decomposition of the error is a useful tool for analyzing learning algorithms. Originally it was proposed for regression, but there are several variants for classification [10, 6]. It decomposes the error in three terms [16], derived with reference to the performance of a learner when trained with different training sets drawn from some distribution of training sets:

- Squared bias: a measure of the error of the central tendency of the learner.
- Variance: a measure of the degree to which the learner's predictions as it is applied to learn models from different training sets.
- Intrinsic noise: a measure of the degree to which the target quantity is inherently unpredictable.

Given that it is infeasible to estimate the intrinsic noise from sample data, this term is usually aggregated to the bias term.

One of the possible explanations for the success of bagging and boosting uses this decomposition [11]. Although there is no general theory on the effects of bagging and boosting on bias and variance, Bagging is assumed to reduce variance without changing the bias. Boosting in the early iterations primarily reduces bias and in the later iterations reduces mainly variance.

## 4 Experimental Validation

### 4.1 Data Sets

The used data sets appear in table 1. They are the ones available in the format of the WEKA library. All of them are from the UCI repository [2]. Some of the data sets were slightly modified. First, for the data sets “splice” and “zoo” an attribute was eliminated. They were instance identifiers. This kind of attributes are not useful for learning, and in the current implementation, they cause a considerably overhead, because continuous attributes are converted to numeric. Second, for the data set “vowel”, there was an attribute indicating if the instance was for training or for testing. This attribute was eliminated. Moreover, this data set includes information of the speakers and their sex. We consider two versions, “vowel-c” and “vowel-n” using and not using this context information, because there are works that use and not use this information.

### 4.2 Settings

The used method for constructing decision trees is one of the available in the WEKA library [17], named J48. It is a reimplementation of C4.5 [12]. The implementations of Bagging and AdaBoost.M1 are also from that library.

The parameters of the different methods were the default ones. The number of classifiers to combine was 10. For rotation-based ensembles, the number of attributes is each group was 3.

PCA is defined for numeric attributes. For the presented method, discrete attributes were converted to numeric ones, with as many attributes as possible values. This transformation was not applied for the methods used for comparison (bagging, boosting...) because they can deal directly with discrete attributes.

The bias and variance are calculated with the method proposed in [16]. It is available in WEKA. The default parameters were used. The exception was the number of times each instance is classified, that was set to 50 (the default value is 10). With these settings, it runs 50×2-fold cross validation.

### 4.3 Results

Table 2 shows the error, bias and variance results. It also includes, for each method, the mean for all the data sets, although it is a very gross measure of relative performance. Rotation-based ensembles have the minimum mean for the error and the variance. The minimum mean for the bias is obtained with boosting, although the result for rotation-based is very close.

**Table 1.** Characteristics of the used data sets

| data set      | classes | examples | discrete | continuous |
|---------------|---------|----------|----------|------------|
| anneal        | 6       | 898      | 32       | 6          |
| audiology     | 24      | 226      | 69       | 0          |
| autos         | 7       | 205      | 10       | 16         |
| balance-scale | 3       | 625      | 0        | 4          |
| breast-cancer | 2       | 286      | 10       | 0          |
| breast-w      | 2       | 699      | 0        | 9          |
| colic         | 2       | 368      | 16       | 7          |
| credit-a      | 2       | 690      | 9        | 6          |
| credit-g      | 2       | 1000     | 13       | 7          |
| diabetes      | 2       | 768      | 0        | 8          |
| glass         | 7       | 214      | 0        | 9          |
| heart-c       | 5       | 307      | 7        | 6          |
| heart-h       | 5       | 294      | 7        | 6          |
| heart-statlog | 2       | 270      | 0        | 13         |
| hepatitis     | 2       | 155      | 13       | 6          |
| hypothyroid   | 4       | 3772     | 22       | 7          |
| ionosphere    | 2       | 351      | 0        | 34         |
| iris          | 3       | 150      | 0        | 4          |
| labor         | 2       | 57       | 8        | 8          |
| letter        | 26      | 20000    | 0        | 16         |
| lymphography  | 4       | 148      | 15       | 3          |
| primary-tumor | 22      | 239      | 17       | 0          |
| segment       | 7       | 2310     | 0        | 19         |
| sonar         | 2       | 208      | 0        | 60         |
| soybean       | 19      | 683      | 35       | 0          |
| splice        | 3       | 3190     | 60       | 0          |
| vehicle       | 4       | 846      | 0        | 18         |
| vote          | 2       | 435      | 16       | 0          |
| vowel-c       | 11      | 990      | 2        | 10         |
| vowel-n       | 11      | 990      | 0        | 10         |
| waveform      | 3       | 5000     | 0        | 40         |
| zoo           | 7       | 101      | 16       | 2          |

The table also includes the geometric mean ratio [15]. For each data set, the ratio is the value for the alternative method divided by the value for rotation-based ensembles. Then, the geometric mean of the ratios is calculated. A geometric mean ratio greater than 1.0 represents an advantage of rotation-based and a lower value represents an advantage to the alternative algorithm. In the table, all the geometric rates are greater than 1.0.

Finally, the table also shows for each method and measure the number of times that the result is better and worst than the result for rotation-based. The only case that is not favourable to the presented method is the bias for boosting, that is better for 18 of 32 data sets. This is consistent with previous results, that indicate that boosting with few iterations reduces mainly the bias [11].

Table 2. Error, bias and variance results

|                 | J48    |               | Bagging J48 |               | Boosting J48 |               | Random Forest |               | Rotations J48 |               |         |        |        |         |        |
|-----------------|--------|---------------|-------------|---------------|--------------|---------------|---------------|---------------|---------------|---------------|---------|--------|--------|---------|--------|
|                 | Error  | Bias Variance | Error       | Bias Variance | Error        | Bias Variance | Error         | Bias Variance | Error         | Bias Variance |         |        |        |         |        |
| anneal          | 0.0202 | 0.0098        | 0.0104      | 0.0099        | 0.0096       | 0.0123        | 0.0051        | 0.0072        | 0.0141        | 0.0050        | 0.0091  | 0.0159 | 0.0082 | 0.0077  |        |
| audiology       | 0.2925 | 0.1573        | 0.1352      | 0.2674        | 0.1462       | 0.2372        | 0.1032        | 0.1340        | 0.3371        | 0.1219        | 0.2152  | 0.2665 | 0.1306 | 0.1359  |        |
| autos           | 0.3385 | 0.1423        | 0.1962      | 0.2894        | 0.1265       | 0.2617        | 0.0932        | 0.1685        | 0.2854        | 0.1030        | 0.1824  | 0.2822 | 0.1321 | 0.1501  |        |
| balance-scale   | 0.2200 | 0.0826        | 0.1374      | 0.1813        | 0.0710       | 0.2014        | 0.0862        | 0.1152        | 0.1927        | 0.0890        | 0.1037  | 0.1130 | 0.0513 | 0.0617  |        |
| breast-cancer   | 0.2933 | 0.2242        | 0.0691      | 0.2860        | 0.2290       | 0.3449        | 0.2126        | 0.1323        | 0.3264        | 0.2160        | 0.1104  | 0.2870 | 0.2133 | 0.0737  |        |
| breast-w        | 0.0572 | 0.0299        | 0.0273      | 0.0445        | 0.0310       | 0.0135        | 0.0418        | 0.0263        | 0.0155        | 0.0437        | 0.0286  | 0.0151 | 0.0312 | 0.0258  | 0.0054 |
| colic           | 0.1505 | 0.1367        | 0.0138      | 0.1516        | 0.1353       | 0.0163        | 0.1932        | 0.1226        | 0.0706        | 0.1670        | 0.1280  | 0.0390 | 0.1701 | 0.1261  | 0.0440 |
| credit-a        | 0.1527 | 0.1102        | 0.0425      | 0.1434        | 0.1120       | 0.0314        | 0.1645        | 0.0984        | 0.0661        | 0.1550        | 0.1076  | 0.0474 | 0.1415 | 0.1092  | 0.0323 |
| credit-g        | 0.2910 | 0.1969        | 0.0941      | 0.2713        | 0.1923       | 0.0790        | 0.2952        | 0.1698        | 0.1254        | 0.2731        | 0.1882  | 0.0849 | 0.2675 | 0.1838  | 0.0837 |
| diabetes        | 0.2730 | 0.1792        | 0.0938      | 0.2555        | 0.1852       | 0.0703        | 0.2771        | 0.1734        | 0.1037        | 0.2589        | 0.1890  | 0.0699 | 0.2434 | 0.1949  | 0.0485 |
| glass           | 0.3550 | 0.1641        | 0.1909      | 0.3157        | 0.1776       | 0.1381        | 0.3048        | 0.1553        | 0.1495        | 0.2894        | 0.1473  | 0.1421 | 0.2989 | 0.1711  | 0.1278 |
| heart-h         | 0.2390 | 0.1383        | 0.1007      | 0.2135        | 0.1372       | 0.0763        | 0.2139        | 0.1199        | 0.0940        | 0.2090        | 0.1355  | 0.0735 | 0.1906 | 0.1259  | 0.0647 |
| heart-c         | 0.2067 | 0.1522        | 0.0545      | 0.1988        | 0.1478       | 0.0510        | 0.2099        | 0.1428        | 0.0671        | 0.2076        | 0.1563  | 0.0513 | 0.1975 | 0.1443  | 0.0532 |
| heart-statlog   | 0.2369 | 0.1334        | 0.1035      | 0.2131        | 0.1300       | 0.0831        | 0.2181        | 0.1374        | 0.0807        | 0.2104        | 0.1331  | 0.0773 | 0.1927 | 0.1276  | 0.0651 |
| hepatitis       | 0.2074 | 0.1448        | 0.0626      | 0.1888        | 0.1415       | 0.0473        | 0.1914        | 0.1255        | 0.0659        | 0.1809        | 0.1243  | 0.0566 | 0.1764 | 0.1264  | 0.0500 |
| hypothyroid     | 0.0058 | 0.0042        | 0.0016      | 0.0056        | 0.0043       | 0.0013        | 0.0052        | 0.0025        | 0.0027        | 0.0124        | 0.0046  | 0.0078 | 0.0066 | 0.0030  | 0.0036 |
| ionosphere      | 0.1157 | 0.0583        | 0.0574      | 0.0929        | 0.0564       | 0.0365        | 0.0887        | 0.0500        | 0.0387        | 0.0783        | 0.0529  | 0.0254 | 0.0725 | 0.0450  | 0.0275 |
| iris            | 0.0647 | 0.0473        | 0.0174      | 0.0617        | 0.0527       | 0.0090        | 0.0668        | 0.0493        | 0.0175        | 0.0548        | 0.0447  | 0.0101 | 0.0527 | 0.0363  | 0.0164 |
| labor           | 0.2309 | 0.1274        | 0.1035      | 0.2193        | 0.1179       | 0.1014        | 0.1695        | 0.0667        | 0.1028        | 0.1821        | 0.0814  | 0.1007 | 0.1291 | 0.0484  | 0.0807 |
| letter          | 0.1551 | 0.0325        | 0.1226      | 0.1013        | 0.0373       | 0.0640        | 0.0662        | 0.0182        | 0.0480        | 0.0798        | 0.0220  | 0.0578 | 0.0665 | 0.0176  | 0.0489 |
| lymphography    | 0.2553 | 0.1363        | 0.1190      | 0.2249        | 0.1236       | 0.1013        | 0.2012        | 0.1055        | 0.0957        | 0.2091        | 0.1276  | 0.0815 | 0.1945 | 0.1078  | 0.0867 |
| primary-tumor   | 0.6271 | 0.3023        | 0.3248      | 0.6067        | 0.3827       | 0.2240        | 0.6267        | 0.3016        | 0.3251        | 0.6194        | 0.3099  | 0.3095 | 0.5838 | 0.3168  | 0.2670 |
| segment         | 0.0477 | 0.0163        | 0.0314      | 0.0401        | 0.0202       | 0.0199        | 0.0279        | 0.0111        | 0.0168        | 0.0313        | 0.0142  | 0.0171 | 0.0272 | 0.0138  | 0.0134 |
| sonar           | 0.2914 | 0.1215        | 0.1699      | 0.2625        | 0.1510       | 0.1115        | 0.2396        | 0.1056        | 0.1340        | 0.2352        | 0.1025  | 0.1327 | 0.2072 | 0.1044  | 0.1028 |
| soybean         | 0.1286 | 0.0558        | 0.0728      | 0.1042        | 0.0516       | 0.0526        | 0.0870        | 0.0475        | 0.0395        | 0.1164        | 0.0483  | 0.0681 | 0.0718 | 0.0417  | 0.0301 |
| splice          | 0.0740 | 0.0472        | 0.0268      | 0.0666        | 0.0466       | 0.0200        | 0.0603        | 0.0334        | 0.0269        | 0.1147        | 0.0223  | 0.0924 | 0.0506 | 0.0345  | 0.0161 |
| vehicle         | 0.3013 | 0.1551        | 0.1462      | 0.2760        | 0.1683       | 0.1077        | 0.2596        | 0.1487        | 0.1109        | 0.2689        | 0.1584  | 0.1105 | 0.2385 | 0.1428  | 0.0957 |
| vote            | 0.0474 | 0.0387        | 0.0087      | 0.0440        | 0.0376       | 0.0064        | 0.0527        | 0.0317        | 0.0210        | 0.0463        | 0.0300  | 0.0163 | 0.0445 | 0.0314  | 0.0131 |
| vowel-c         | 0.3036 | 0.0957        | 0.2479      | 0.1938        | 0.0555       | 0.1383        | 0.1585        | 0.0342        | 0.1243        | 0.1460        | 0.0101  | 0.1359 | 0.1057 | 0.0107  | 0.0950 |
| vowel-n         | 0.2984 | 0.0504        | 0.2480      | 0.1968        | 0.0560       | 0.1408        | 0.1564        | 0.0339        | 0.1225        | 0.1511        | 0.0288  | 0.1223 | 0.1097 | 0.0147  | 0.0950 |
| waveform        | 0.2534 | 0.1061        | 0.1473      | 0.1897        | 0.1187       | 0.0710        | 0.1900        | 0.0996        | 0.0904        | 0.1861        | 0.1055  | 0.0806 | 0.1625 | 0.0973  | 0.0652 |
| zoo             | 0.1046 | 0.0450        | 0.0596      | 0.1053        | 0.0410       | 0.0643        | 0.0745        | 0.0295        | 0.0450        | 0.0667        | 0.0204  | 0.0463 | 0.0964 | 0.0343  | 0.0621 |
| Mean            | 0.2075 | 0.1063        | 0.1012      | 0.1822        | 0.1092       | 0.0730        | 0.1781        | 0.0919        | 0.0862        | 0.1797        | 0.0955  | 0.0842 | 0.1592 | 0.0928  | 0.0663 |
| Geo. mean ratio | 1.3597 | 1.3065        | 1.4318      | 1.1904        | 1.3290       | 1.0577        | 1.1168        | 1.0441        | 1.3234        | 1.1708        | 1.0486  | 1.3061 | 1.4708 | 1.15    | 1.3061 |
| Win - Loss      | 2 - 30 | 3 - 29        | 6 - 26      | 4 - 28        | 2 - 30       | 11 - 21       | 6 - 26        | 18 - 14       | 5 - 27        | 4 - 28        | 13 - 15 | 6 - 26 | 4 - 28 | 13 - 15 | 6 - 26 |

## 5 Conclusions and Future Work

Rotation-based ensembles is a novel approach for the generation of ensembles of classifiers. The method compares favorably to Bagging, Random Forest and AdaBoost.M1, when using decision trees as base classifiers. The proposed method has smaller mean values, favourable geometric mean ratios, and is more times better than worst when compared with the other methods. The only exception is that the bias is better for boosting than for rotation-based ensembles.

It is somewhat surprising that the results of the method were better than the results for boosting, because it appears to be rather more elaborated, and with a most solid theoretical basis. Nevertheless, one of the features of boosting, not present in neither our method nor Bagging is the ability of obtaining *strong* classifiers from *weak* ones, such as decision stumps. On the other hand, Bagging and our method allow the construction of the base classifiers in parallel.

Currently, only decision trees have been considered as base classifiers. Other methods that are not rotation-invariant can be considered. On the other hand, rotation-invariant methods can also be used if the number of attributes in the transformed data set is different than the number for the original data set.

The experimental validation has been limited to classification problems. Apparently, the method can be also used for regression problems, if the base regression method is not invariant to rotations, as is the case for regression trees.

The presented method is compatible with another ensemble methods. First, the base classifier for an ensemble method can be another ensemble method. For instance, it could be possible to use the presented method using bagging as base classifier. In this way, it could be possible to combine 100 decision trees, but applying the PCA procedure only 10 times. This can be useful because the PCA procedure is slower than resampling, the used strategy for Bagging.

Second, it is possible to apply several transformations to the original data set. For instance, resample and then the presented method. In this way, the two ensemble methods are not used hierarchically, but simultaneously. Hence, it is necessary to study the possible usefulness of some of these combinations.

## References

1. Eric Bauer and Ron Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, 36(1–2):105–139, 1999.
2. C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998. <http://www.ics.uci.edu/~mllearn/MLRepository.html>.
3. Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.
4. Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.
5. Thomas G. Dietterich. Ensemble methods in machine learning. In *Multiple Classifier Systems 2000*, pages 1–15, 2000.
6. P. Domingos. A unified bias-variance decomposition and its applications. In *17th International Conference on Machine Learning*, pages 231–238, 2000.
7. Jiawei Han and Micheline Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2001.



8. Tin Kam Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844, 1998.
9. Tin Kam Ho. A data complexity analysis of comparative advantages of decision forest constructors. *Pattern Analysis and Applications*, 5:102–112, 2002.
10. R. Kohavi and D. Wolpert. Bias plus variance decomposition for zero-one loss functions. In *13th International Machine Learning Conference (ICML96)*, 1996.
11. Ludmila I. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms*. Wiley-Interscience, 2004.
12. J. Ross Quinlan. *C4.5: programs for machine learning*. Machine Learning. Morgan Kaufmann, San Mateo, California, 1993.
13. Juan José Rodríguez and Carlos J. Alonso. Rotation-based ensembles. In *urrent Topics in Artificial Intelligence: 10th Conference of the Spanish Association for Artificial Intelligence*, pages 498–506. Springer, 2004.
14. Robert E. Schapire. The boosting approach to machine learning: An overview. In *MSRI Workshop on Nonlinear Estimation and Classification*, 2002. <http://www.cs.princeton.edu/~schapire/papers/msri.ps.gz>.
15. Geoffrey I. Webb, Janice R. Boughton, and Zhihai Wang. Not so naive bayes. *Machine Learning*, 58:5–24, 2005.
16. Geoffrey I. Webb and Paul Conilione. Estimating bias and variance from data, 2004. <http://www.csse.monash.edu.au/~webb/Files/WebbConilione03.pdf>.
17. Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, 1999. <http://www.cs.waikato.ac.nz/ml/weka>.

# Comparative Assessment of the Robustness of Missing Data Imputation Through Generative Topographic Mapping

Iván Olier and Alfredo Vellido

Department of Computing Languages and Systems (LSI),  
Polytechnic University of Catalonia (UPC),  
C. Jordi Girona, 1-3. 08034 , Barcelona, Spain  
{iaolier, avellido}@lsi.upc.es

**Abstract.** The incompleteness of data is a most common source of uncertainty in real-world Data Mining applications. The management of this uncertainty is, therefore, a task of paramount importance for the data analyst. Many methods have been developed for missing data imputation, but few of them approach the problem of imputation as part of a general data density estimation scheme. Amongst the latter, a method for imputing and visualizing multivariate missing data using Generative Topographic Mapping was recently presented. This model and some of its extensions are tested under different experimental conditions. Its performance is compared to that of other missing data imputation techniques, thus assessing its relative capabilities and limitations.

## 1 Introduction

The occurrence of missing data is a pervasive problem in real application areas, especially acute in domains such as social sciences [1], bioinformatics [2], environmental sciences [3], or signal processing [4]. Methods that impute the missing values are therefore of paramount importance for the successful analysis of such data. Different methods are suitable for different types of data (continuous, discrete, categorical) and for different application fields.

Many data imputation methods have been proposed in the literature: their review is beyond the scope of this brief paper and can be found elsewhere [5]. A fair comparison between methods is difficult not only because of their dependency on data types and application domains, but also because they are not likely to perform homogeneously under all types of conditions. The aim of this study is not carrying out a full-blown comparison of methods. Instead, a fairly detailed evaluation of the robustness of performance of a proposed method is carried out, followed by a comparison of performance with two alternative methods.

The method for missing data imputation put forward in this study was only recently proposed and preliminarily evaluated [6,7]. It is based on a constrained mixture of Gaussians model, the Generative Topographic Mapping (GTM) [8],

trained by EM [9] and it encompasses missing data imputation, clustering, and visualization in a low dimensional latent space. The missing data imputation method is here extended to adaptively regularized variants of the GTM. Their performance is compared with that of two alternative methods, in order to assess the capabilities and limitations of the model.

## 2 The Generative Topographic Mapping

In this section we introduce the neural network-inspired Generative Topographic Mapping (GTM) [8], a latent variable model with sound foundations in probability theory that can simultaneously perform soft-clustering and visualization of the available data as well as imputation of missing data values. GTM is defined as a probabilistic alternative to Kohonen's SOM [10]: it generates a non-linear and topographically preserving mapping, from a low-dimensional latent space in  $\mathfrak{R}^L$  (with  $L$  being usually 1 or 2 for visualization purposes) onto the  $\mathfrak{R}^D$  space in which the original data reside, through a linear combination of basis functions defined as:

$$\mathbf{y} = \mathbf{W}\phi(\mathbf{u}) \tag{1}$$

where  $\mathbf{y}$  is a  $D$ -dimensional point in data space,  $\mathbf{u}$  is an  $L$ -dimensional point in latent space,  $\mathbf{W}$  is the matrix that generates the mapping, and  $\phi$  is a set of  $S$  basis functions  $\phi_s$  (Gaussians in this study). To achieve computational tractability, the prior distribution of  $\mathbf{u}$  in latent space is constrained to form a uniform discrete grid of  $M$  centres, analogous to the layout of the SOM units, in the form:

$$p(\mathbf{u}) = \frac{1}{M} \sum_{i=1}^M \delta(\mathbf{u} - \mathbf{u}_i) \tag{2}$$

This way defined, the GTM becomes a constrained mixture of Gaussians. A density model in data space is therefore generated for each component  $i$  of the mixture, leading to the definition of the following log-likelihood expression:

$$L(\mathbf{W}, \beta) = \sum_{n=1}^N \ln p(\mathbf{x}_n | \mathbf{W}, \beta) = \sum_{n=1}^N \ln \left( \int p(\mathbf{x}_n | \mathbf{u}, \mathbf{W}, \beta) p(\mathbf{u}) d\mathbf{u} \right) = \sum_{n=1}^N \ln \left\{ \frac{1}{M} \sum_{i=1}^M \left( \frac{\beta}{2\pi} \right)^{D/2} \exp \left\{ -\frac{\beta}{2} \|\mathbf{W}\phi(\mathbf{u}_i) - \mathbf{x}_n\|^2 \right\} \right\} \tag{3}$$

where  $\mathbf{x}_n$  is one of  $N$  points in data space,  $\mathbf{y}_i = \mathbf{W}\phi(\mathbf{u}_i)$  is the  $i^{\text{th}}$  component centre, and  $\beta$  is the estimated common inverse variance of the isotropic Gaussian distributions in data space whose centres are  $\mathbf{y}_i$ . Parameter optimisation can be accomplished using the EM algorithm [9]. Matrix  $\mathbf{W}$  is re-estimated as the solution of the following system of equations:

$$\Phi^T \mathbf{G} \Phi \mathbf{W}_{new}^T = \Phi^T \mathbf{R}_{old} \mathbf{X} \tag{4}$$

where  $\Phi$  is a  $M \times S$  matrix with elements  $\phi_{is}$ ,  $\mathbf{R}$  is a matrix with elements  $R_{in} = p(\mathbf{u}_i | \mathbf{x}_n)$ ,  $\mathbf{G}$  is a matrix with elements  $\sum_{n=1}^N R_{in}$  in the diagonal and zeros elsewhere, and  $\mathbf{X}$  is the data matrix. Parameter  $\beta$  is re-estimated as

$$\beta_{new}^{-1} = \frac{1}{ND} \sum_{n=1}^N \sum_{i=1}^M R_{in} \|\mathbf{y}_i - \mathbf{x}_n\|^2 \tag{5}$$

This algorithm is also at the core of the GTM ability to regenerate missing data.

### 2.1 Adaptive Regularization of the GTM

An advantage of the probabilistic setting of the GTM is the possibility of introducing regularization in the mapping. This procedure automatically regulates the level of map smoothing necessary to avoid data overfitting, resorting to either a single regularization term [11], or to multiple ones [Selective Map Smoothing : 12].

The first case entails the definition of a modified log-likelihood of the form:

$$L_{PEN}(\mathbf{W}, \alpha, \beta) = \sum_{n=1}^N \ln p(\mathbf{x}_n | \mathbf{W}, \beta) - \frac{1}{2} \alpha \|\mathbf{w}\|^2 \tag{6}$$

where  $\mathbf{w}$  is a vector shaped by concatenation of the different column vectors of  $\mathbf{W}$  and  $\alpha$  is a regularization coefficient. The re-estimation of  $\mathbf{W}$  now becomes:

$$\left( \Phi^T \mathbf{G} \Phi + \frac{\alpha}{\beta} \mathbf{I} \right) \mathbf{W}_{new}^T = \Phi^T \mathbf{R}_{old} \mathbf{X} \tag{7}$$

and the re-estimation of  $\beta$  remains unchanged. The regularization coefficient  $\alpha$  can be adaptively optimized using the evidence approximation within a Bayesian formalism [13] which aims to maximize  $p(\alpha, \beta | \mathbf{X})$ . Details of this method and its extensions can be found in [11, 12].

### 2.2 Missing Data Imputation Through Generative Topographic Mapping

A further advantage of the GTM’s probability theory foundations is its ability to cope with incomplete data and regenerate them [7,14] taking advantage of its optimization using the EM algorithm, which is ideally suited to deal with missing data [15].

The GTM is trained (i.e., the probability density of the data is approximated) using only the observed data, whereas the missing data are treated as latent (i.e. unobservable) variables. Missing values are reconstructed as part of the maximization step of EM, prior to the re-evaluation of  $\mathbf{W}$ , using the posterior probabilities  $p(\mathbf{u}_i | \mathbf{x}_n^o)$  of the latent space centres being responsible for each point in the original data space (*responsibilities*) that are calculated only on the observed data. The expectation of the missing data (identified by superscript  $m$ , as opposed to  $o$  for observed) is:

$$\left\langle \mathbf{x}_n^m \mid \mathbf{x}_n^o, \mathbf{W}, \beta \right\rangle = \sum_{i=1}^M p(\mathbf{u}_i \mid \mathbf{x}_n^o) (\mathbf{W} \phi(\mathbf{u}_i))^m \tag{8}$$

Parameter  $\mathbf{W}$  is re-estimated with a variation on Eq.4:

$$\Phi^T \mathbf{G}^o \Phi \mathbf{W}_{new}^T = \Phi^T \mathbf{R}^o \mathbf{X}^{reg} \tag{9}$$

where the superscript  $o$  indicates responsibilities calculated only on the observed data, and the superscript  $reg$  indicates that the fully regenerated (observed plus imputed) data are used. Parameter  $\beta$  is re-estimated as:

$$\beta_{new}^{-1} = \frac{1}{ND} \sum_{n=1}^N \sum_{i=1}^M R_{in}^o \left( \left\| \mathbf{y}_i^o - \mathbf{x}_n^o \right\|^2 + \left\langle z_{in} \left\| \mathbf{y}_i^m - \mathbf{x}_n^m \right\|^2 \right\rangle \right) \tag{10}$$

where  $z_{in}$  is an indicator variable that represents our ignorance of which latent centre  $i$  is responsible for the generation of the missing data in  $\mathbf{x}_n$ . Compare this to Eq.5.

### 2.3 Missing Data Imputation for the Adaptively Regularized GTM

The algorithmic implementation of GTM with adaptive regularization entails interleaving the EM algorithm with the Bayesian re-estimation of hyperparameters  $\alpha$  and  $\beta$ . This re-estimation must undergo some changes to adapt it to the GTM procedure for missing values imputation. Eq.7 then becomes:

$$\left( \Phi^T \mathbf{G}^o \Phi + \frac{\alpha}{\beta} \mathbf{I} \right) \mathbf{W}_{new}^T = \Phi^T \mathbf{R}_{old}^o \mathbf{X}^{reg} \tag{11}$$

Update formulae for hyperparameters  $\alpha$  and  $\beta$  are now changed, and can be calculated through maximization of the approximate expression:

$$L_{PEN}^{reg}(\mathbf{w}_*, \alpha, \beta) - \frac{1}{2} \ln |\mathbf{H}_*^{reg}| + \frac{W}{2} \ln \alpha + C \tag{12}$$

where  $L_{PEN}$  is that of Eq.6, evaluated at the most probable values of  $\mathbf{w}$  (symbolized by the asterisk), and where all the constant terms have been grouped as  $C$ ; finally,  $\mathbf{H}_*$  is the Hessian of  $-L_{PEN}(\mathbf{w}_*, \alpha, \beta)$ .

## 3 Alternative Missing Data Imputation Methods

We do not aim to provide a full review of missing data imputation methods here, which can be found elsewhere [5]. The target is somehow narrower, and only two methods are selected for comparison with the GTM. The first, introduced by Troyanskaya et al. [4] and called KNNimpute, is based on K-Nearest Neighbours (KNN). KNN shares with the GTM that both can be considered *prototype* methods that attempt to represent the overall data distribution with a reduced number of generated instances (prototypes). Unlike the GTM method, though, KNNimpute is

mostly heuristic, a feature that affects its results, as we shall see later. The algorithm can be summarized as follows: For each incomplete record in the data set, its missing values are replaced by a weighted-by-similarity average of the corresponding observed values of the  $K$  nearest neighbours. The second method is Unconditional Mean Imputation where, for each incomplete record in the data set, its missing values are replaced by the average of the observed values of the corresponding attribute for all records. This will be used as a baseline for the acceptability of the results.

Although not used in this comparative study, some methods, related to the GTM one that is presented in this paper, have been recently defined. They include those of Hunt and Jorgensen [16] for a mix of continuous and categorical data and Wang et al. [17] for general mixtures of multivariate Student distributions.

## 4 Experimental Settings

Five data sets of very different characteristics were selected for the experiments. An artificial data set with 150 registers was generated using a mixture of three 4-D generalized Gaussians. Additionally, the *Iris* and *Ecoli* data sets from UCI Repository<sup>1</sup>, the *Oil 3-Phase* data from the GTM homepage<sup>2</sup>, and the *kin8nm* data set of the *kin family* from the Delve project site<sup>3</sup>, were selected. Throughout the reported experiments, a *missing completely at random* (MCAR) [15] setting was assumed, where random masks were created to simulate different proportions of missing data. Following an otherwise common procedure [4], the performance of the models, in terms of the correctness of missing data imputation, was evaluated resorting to the

normalized root mean square error  $RMSE = \left( \frac{1}{N} \sum_n (\mathbf{x}_n - \mathbf{x}_n^{reg})^2 \right)^{1/2}$ , where  $\mathbf{x}_n^{reg}$  is a

data instance as regenerated by the model.

For the GTM, the adaptive matrix  $\mathbf{W}$  was initialized following a standard procedure [8]. The grid of GTM latent centres was fixed to a square layout of 10x10 nodes. The corresponding grid of basis functions  $\phi$  was fixed to a 5x5 layout. Alternative layouts were tested without significant differences being observed.

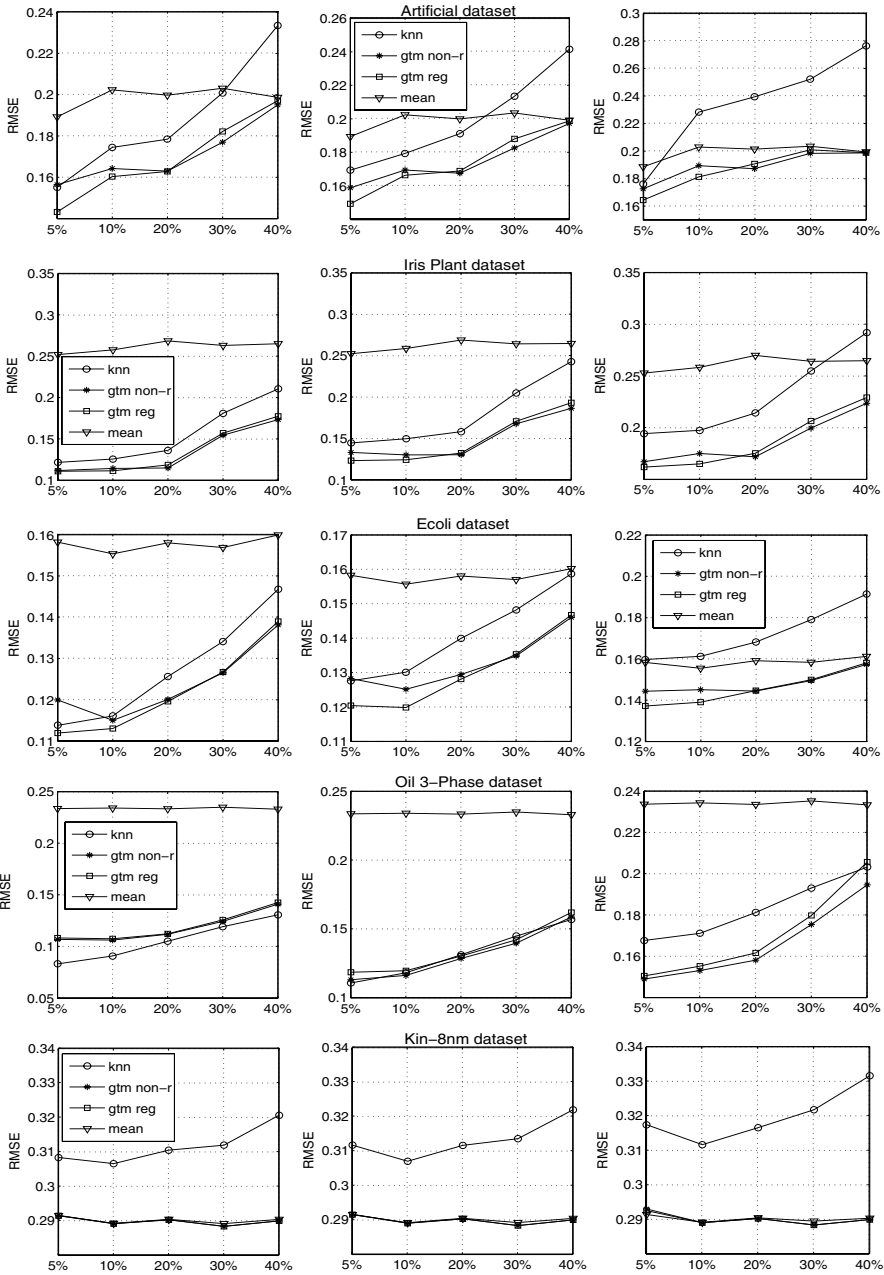
The experimental settings were varied in different ways to cover a broad spectrum of conditions:

- Several levels of data incompleteness were considered, ranging from 5% to 40%, in order to test the limits of the robustness of the data imputation methods.
- Two GTM models were tested: the completely unregularized GTM and its adaptively regularized variant with a single common regularization term [11].
- Three levels of Gaussian noise contamination were added (with standard deviations of 0.05, 0.1 and 0.2), in order to assess whether regularization improves the model performance in the presence of noise, and to what extent.

<sup>1</sup> [www.ics.uci.edu/~mllearn/MLRepository.html](http://www.ics.uci.edu/~mllearn/MLRepository.html)

<sup>2</sup> [www.ncrg.aston.ac.uk/GTM](http://www.ncrg.aston.ac.uk/GTM)

<sup>3</sup> [www.cs.toronto.edu/~delve/data/kin/desc.html](http://www.cs.toronto.edu/~delve/data/kin/desc.html)



**Fig. 1.** Performance curves for the GTM models, KNNImpute, and Unconditional Mean Imputation, on each of the datasets, following the layout described in the main text

For each of the experimental variants, the GTM was trained 10 times. Mean results are provided in Fig. 1.

## 5 Results and Discussion

The results of the set of experiments outlined in section 4 are presented in Fig. 1, according to the following layout: each of the rows corresponds to a data set. In turn, each of the rows consists of three displays for data with increasing level of added noise. Finally, each display contains results for the GTM models (*gtm non-r* and *gtm reg* in the figures), Unconditional Mean Imputation (*mean*), and KNNimpute (*knn*).

Overall, the imputation error is lower for the GTM models than for KNNimpute. The negative influence of added noise can be seen for all methods, although GTM models are more robust towards high noise levels than KNNimpute. All models degrade in a reasonably graceful way with the increase of the level of missing values. It is very interesting, though, to find out that, due to its probabilistic definition, the performance of the GTM-based method is bound by that of the Unconditional Mean Imputation baseline. This is not true for KNNimpute. An extreme example of this are the results for a very complex data set such as *kin8nm*: GTM does not provide any improvement over Unconditional Mean, but KNNimpute performs much worse than both. The regularization of the GTM consistently improves its performance, but only up to a level of data incompleteness usually around 20%, suggesting that higher levels of incompleteness render regularization useless in terms of missing data reconstruction.

## 6 Conclusions

In this study, we have tested a method for missing data imputation [6] based on the properties of the EM algorithm as applied on a constrained mixture of Gaussians: the GTM model [8]. We have extended this method to encompass adaptively regularized variants, providing details of the necessary modifications in the algorithm. This extension does entail only minor extra computational effort. The performance of the GTM models has been compared to that of two alternative methods: KNNimpute and Unconditional Mean Imputation.

The results show an overall satisfactory performance of the GTM method for missing data imputation, degrading gracefully even for very high levels of data incompleteness. The capabilities and limitations of the GTM-based method have been highlighted.

## References

1. Olinsky, A., Chen, S., Harlow, L.: The Comparative Efficacy of Imputation Methods for Missing Data in Structural Equation Modelling. *Eur. J. Oper. Res.* **151** (2003) 53–79
2. Troyanskaya, O., Cantor, M., Sherlock, G., Brown, P., Hastie, T., Tibshirani, R., Botstein, D., Altman, R.B.: Missing Value Estimation Methods for DNA Microarrays. *Bioinformatics* **17** (2001) 520-525



3. Vicente, D., Vellido, A., Martí, E., Comas, J., Rodríguez-Roda, I.: Exploration of the Ecological Status of Mediterranean Rivers: Clustering, Visualizing and Reconstructing Streams Data Using Generative Topographic Mapping. In Zanasi, A., Ebecken, N.F.F., Brebbia, C.A. (eds.) *Data Mining V. W.I.T. Transactions on Information and Communication Technologies*, Vol.30 (2004) 121-130
4. Cooke, M.P., Green, P.D., Josifovski, L., Vizinho, A.: Robust Automatic Speech Recognition with Missing and Unreliable Acoustic Data. *Speech Commun.* **34** (2001) 267–285
5. Schafer, J.L.: *Analysis of Incomplete Data*. Chapman & Hall, London (1997)
6. Sun, Y., Tiño, P., Nabney, I.: *GTM-Based Data Visualization with Incomplete Data*. Technical Report, NCRG, Aston University, UK (2001)
7. Olier, I., Vellido, A.: Assessing the Robustness of Missing Data Imputation Through Generative Topographic Mapping. *ACIA Newsletter* **31** (2004) 14-20
8. Bishop, C.M., Svensén, M., Williams, C.K.I.: *GTM: The Generative Topographic Mapping*. *Neural Comput.* **10** (1998) 215-234
9. Dempster, A.P., Laird, M.N., Rubin, D.B.: Maximum Likelihood from Incomplete Data Via the EM Algorithm. *J. Roy. Statist. Soc. Ser. B* **39** (1977) 1-38
10. Kohonen, T.: *Self-organizing Maps*. Springer-Verlag, Berlin (1995)
11. Bishop, C.M., Svensén, M., Williams, C.K.I.: Developments of the Generative Topographic Mapping. *Neurocomputing* **21** (1998) 203-224
12. Vellido, A., El-Deredy, W., Lisboa, P.J.G.: Selective Smoothing of the Generative Topographic Mapping. *IEEE Trans. Neural Networks* **14** (2003) 847-852
13. Mackay, D.J.C.: A Practical Bayesian Framework for Back-Propagation Networks. *Neural Comput.* **4** (1992) 448-472
14. Carreira-Perpiñán, M.A.: Reconstruction of Sequential Data with Probabilistic Models and Continuity Constraints. In: Solla, S.A., Leen, T.K., Müller, K.-R. (eds), *Advances in Neural Information Processing Systems*, Vol. 12. MIT Press (2000) 414-420
15. Ghahramani, Z., Jordan, M.I.: *Learning from Incomplete Data*. Technical Report, AI Laboratory, MIT (1994)
16. Hunt, L., Jorgensen, M.: Mixture Model Clustering for Mixed Data with Missing Information. *Comput. Statist. Data Anal.* **41** (2003) 429-440
17. Wang, H.X., Zhang, Q.B., Luo, B., Wei, S.: Robust Mixture Modelling Using Multivariate t-Distribution with Missing Information. *Pattern Recognit. Lett.*, **25** (2004) 701–710

# Induction of Decision Trees Using an Internal Control of Induction\*

Gonzalo Ramos-Jiménez, José del Campo-Ávila, and Rafael Morales-Bueno

Departamento de Lenguajes y Ciencias de la Computación,  
E.T.S. Ingeniería Informática. Universidad de Málaga,  
Málaga, 29071, Spain

**Abstract.** In this paper we present CIDIM (Control of Induction by sample DIvision Method), an algorithm that has been developed to induce small and accurate decision trees using a set of examples. It uses an internal control of induction to stop the induction and to avoid the overfitting. Other ideas like a dichotomic division or groups of consecutive values are used to improve the performance of the algorithm. CIDIM has been successfully compared with ID3 and C4.5. It induces trees that are significantly better than those induced by ID3 or C4.5 in almost every experiment.

## 1 Introduction

Learning from examples and inducing knowledge from them form a very wide research area in Machine Learning. Many knowledge representations have been proposed, but one of the most used is the decision tree. Decision trees have been extendedly used by Machine Learning community (CART [1], ID3 [2], C4.5 [3], ITI [4] ... ) and they have some positive characteristics. They have the ability of splitting the hyperspace into subspaces and fitting each space with different models. They have also a desirable feature like is the understandability.

Decision trees are usually evaluated considering their accuracy and their size [5]. An accurate tree is desirable, but, if the tree is also small, it is even more desirable. It is known that finding the smallest tree that classifies a set of examples without error is a *NP – complete* problem [6] and designing a good split criteria is fundamental.

In this paper we introduce CIDIM (Control of Induction by sample DIvision Method), an algorithm that induces decision trees using an internal control of induction. This control, combined with a dichotomic division of the training set of examples and with a grouping of consecutive values, lets CIDIM build small and accurate trees.

The rest of paper is organized as follows: Section 2 presents some preliminary concepts and sets out the basic notation. In Section 3, we introduce CIDIM. Some

---

\* This work has been partially supported by the MOISES project, number TIC2002-04019-C03-02, of the MCyT, Spain.

experimental results are shown in Section 4. Finally, in Section 5, we summarise our conclusions and suggest future lines of research.

## 2 Preliminary Concepts

A *problem* with  $a$  attributes can be defined as a pair  $(\mathbf{m}, k) \in \mathbb{N}^a \times \mathbb{N}$ , where  $\mathbf{m} = (m_1, \dots, m_a)$ . Each *attribute*  $X_i$  is defined in a domain  $D_i = \{1, 2, \dots, m_i\}$ , where  $i = 1, \dots, a$ . We call the set of attributes  $\mathbf{X}$ . Thus,  $\mathbf{X} = \{X_1, \dots, X_a\}$ .

There is a special attribute called *Class* that we denote  $C$  and its domain is  $D = \{1, 2, \dots, k\}$ , where  $k \geq 2$ .

For a given problem we can define the universe of examples ( $U_E$ ) as  $U_E = D_1 \times \dots \times D_a \times D$  and an *example* ( $e$ ) as an element of  $U_E$ .

In this way  $e = (X_1(e), X_2(e), \dots, X_a(e), C(e)) \in U_E$ , where  $X_i(e)$  represents the value of the attribute  $i$  in the example  $e$  ( $i = 1, \dots, a$ ) and  $C(e)$  is the class that matches this example.

We will work with finite sequences of examples ( $E$ ), thus  $E = \{e_1, e_2, \dots, e_N\}$ , where elements can be repeated.

## 3 CIDIM

CIDIM (Control of Induction by sample DIvision Method) is an algorithm that has been developed to induce accurate and small decision trees. It can be used with any problem with a finite number of attributes. These attributes must be nominal and can be ordered or not. The class attribute must have a finite number of unordered classes.

Three basic ideas support CIDIM: division of the training set, group of consecutive values and internal control of induction. In the following subsections we comment these characteristics with more detail.

### 3.1 Division of the Training Set

The top down decision induction tree (TDIDT) algorithms [2, 3], generally, divide the set of examples into two subsets: the training subset (used to induce the tree) and the test subset (used to test the results). CIDIM makes an additional division. It divides the training subset into two new subsets with the same class distribution and similar size: the construction subset (called *CNS*) and the control subset (called *CLS*). A formal description of *CNS* and *CLS* is shown in Figure 1.

*CNS* and *CLS* are used in the expansion process. Every node has its corresponding *CNS* and *CLS* subsets. When an expansion is made, *CNS* and *CLS* subsets of the parent node are divided into multiple *CNS* and *CLS* subsets, each one corresponding to the appropriate son node. Thus, the size of *CNS* and *CLS* decrease as the node is deeper in the tree. These subsets are used when groups of consecutive values are formed (see Subsection 3.2) and when the evaluation of the internal control of induction is tested (see Subsection 3.3).

$$\begin{array}{l}
 |CNS| \neq 0 \quad \wedge \quad |CLS| \neq 0 \\
 CNS \cap CLS = \emptyset \\
 CNS \cup CLS = E \\
 0 \leq \left| \{e \in CNS \mid C(e) = i\} \right| - \left| \{e \in CLS \mid C(e) = i\} \right| \leq 1 \quad \forall i \in D
 \end{array}$$

Fig. 1. CNS and CLS subsets

### 3.2 Group of Consecutive Values

Let us consider a nominal attribute with values  $\{O_1, O_2, \dots, O_n\}$ . Typically, TDIDT algorithms add one branch to the tree for each value, if the attribute is selected to be expanded. But, if it is known that an attribute is ordered, it will be desirable no to treat it like an unordered attribute.

CIDIM has different expansion behaviours depending on the kind of attribute selected to expand a node. When the attribute is unordered, the expansion is the classical expansion, but when the attribute is ordered, CIDIM uses a greedy algorithm to find groups of consecutive values. It is based on a recursive splitting of the values into groups. Initially there is an unique group with all the values of the attribute that is being considered. In each step, CIDIM evaluates if a split will produce an improvement. The process continues until each group has only one value or until there is no improvement.

The improvement of splitting is measured on CNS using a disorder measure ( $dm\_split$ ). Like BOAT algorithm [7], CIDIM does not fix the disorder measure and any one can be used. Thus, given a node to be expanded, CIDIM searches which is the best split for each unused attribute, so, as much pairs "attribute-split" as unused attributes will be calculated.

Another disorder measure ( $dm\_improve$ ) is used to decide which pair is the one that produces the best improvement considering CNS. This new disorder measure is also not fixed like the previous one. The same disorder measure can be used in both cases. We have empirically detected that entropy is a very good disorder measure to be used in splitting process and the disorder measure to decide which is the best one depends on the problem.

Once the best pair attribute-split (*Attribute&Split*) is selected, the internal control of induction (see Subsection 3.3) decides if the expansion is possible.

Obviously, a same attribute can be divided into different groups of values in different branches. It depends on the CNS and CLS subsets in each node. Remember that CNS and CLS subsets are partitioned during the expansion process and different leaves have different CNS and CLS subsets.

### 3.3 Internal Control of Induction

Usually, the expansion of the tree finishes when all examples associated with a node belong to the same class, yielding too large trees. In order to avoid this overfitting, external conditions are considered by different algorithms (C5, an updated version of C4.5, demands that at least two branches have at least a pre-configurable number of examples). CIDIM uses the following as an internal

condition: a node is expanded only if its expansion improves the accuracy calculated on *CLS*. Tree expansion supervision is local for every node and it is driven by two indexes: the absolute index  $I_A$  and the relative index  $I_R$  (see equations in (1)). For every step, a node is expanded only if one or both indexes are increased. If one index decrease, expansion is not made. The absolute and relative indexes are defined as

$$I_A = \frac{\sum_{i=1}^N CORRECT(e_i)}{N} \quad I_R = \frac{\sum_{i=1}^N P_{C(e_i)}(e_i)}{N} \quad (1)$$

where  $N$  is the number of examples in *CLS*,  $e$  a single example,  $C(e)$  the class of the  $e$  example,  $P_m(e)$  the probability of  $m$  class for the  $e$  example in the node using *CNS*, and  $CORRECT(e) = 1$  if  $P_{C(e_i)} = \max\{P_1(e), P_2(e), \dots, P_k(e)\}$  or 0 if another case.

### 3.4 The Algorithm

Now we have presented the basis of CIDIM, we can introduce the algorithm. A pseudocode description of CIDIM can be seen in the Figure 2.

CIDIM needs a set of examples ( $E$ , the training set) and two disorder measures ( $dm\_split$  for splitting and  $dm\_improve$  for selecting the best one) to induce a decision tree, whose root node ( $RootCIDIMTree$ ) is given as output.

The first step in the algorithm is a random dichotomic division of the training set ( $Random\_Dichotomic\_Division\_Keeping\_Class\_Frequency(E)$ ) resulting two subsets (*CNS* and *CLS*). This division keeps the class frequency of the original set ( $E$ ) and is made following the description given in Subsection 3.1. Once the training set is divided, the root node ( $RootCIDIMTree$ ) is created using *CNS* and *CLS*. Then, CIDIM begins an iterative process that finishes when all nodes are labelled. This process try to expand unlabelled nodes

**In:**  $E, dm\_split, dm\_improve$

1.  $\{CNS, CLS\} = Random\_Dichotomic\_Division\_Keeping\_Class\_Frequency(E)$
2.  $RootCIDIMTree = New\_Node(CNS, CLS)$
3.  $UnlabelledNodes = \{RootCIDIMTree\}$
4. **while**  $UnlabelledNodes \neq \emptyset$  **do:**
5.      $Node = Random\_Select(UnlabelledNodes)$
6.      $Attribute\&Split = Best\_Attribute\_Split(Node, dm\_split, dm\_improve)$
7.     **if**  $Improve\_Prediction(Node, Attribute\&Split)$  **then**
8.          $NewNodes = Expand(Node, Attribute\&Split)$
9.          $UnlabelledNodes = UnlabelledNodes - \{Node\} + NewNodes$
10.     **else**
11.          $UnlabelledNodes = UnlabelledNodes - \{Node\}$

**Out:**  $RootCIDIMTree$

**Fig. 2.** CIDIM algorithm

while they exists (*UnlabelledNodes*  $\neq \emptyset$ ). The process begins selecting an unlabelled node in a random way ( $Node = Random\_Select(UnlabelledNodes)$ ) and calculating the pair attribute-split that produces the best improvement ( $Attribute\&Split = Best\_Attribute\_Split(Node, dm\_split, dm\_improve)$ ). Both disorder measures and *CNS* subset of the node are used to calculate this pair like is described in 3.2. Once we have this pair, CIDIM tests if the best expansion that can be made using the selected pair (*Attribute&Split*) improves the prediction given by the unexpanded node. If the expansion must not be made, the node is labelled as a leaf node, but, if the expansion is possible, new nodes are created using the attribute selected and the split calculated (*Attribute&Split*).

We must note that a previous and simplified version of CIDIM [8] has been used to solve real problems, such as system modelling [9] or modelling of prognosis of breast cancer relapse [10].

## 4 Experimental Results

The experiments we have done and the results we have obtained are now exposed. Before we go on to deal with the particular experiments, we must explain some questions:

- The five datasets we have used are summarised in Table 1 that shows the number of examples, the number of attributes, and the number of values for the class. All these datasets have been taken from the *UCI Machine Learning Repository* [11] and are available online. All the datasets used have a common feature: attributes are continuous and have been discretized. We have used these because CIDIM is designed for dealing with nominal variables (ordered or unordered).
- CIDIM has been compared with other well-known method: C4.5 [3]. In addition, results for ID3 have been used to calculate a combined index that it is explained below. For the experiments, we have used the implementation of ID3 and C4.5 given in Weka [12]. CIDIM has been configured with entropy as the disorder measure for splitting (*dm\_split*) and the disorder measure for selecting the best split (*dm\_improve*). ID3 and C4.5 have been configured with their default configuration. ID3

**Table 1.** Summary table for the datasets used in experiments

| Name       | UCI Repository Name     | Examples | Attributes | Classes |
|------------|-------------------------|----------|------------|---------|
| Balance    | balance-scale           | 625      | 4          | 3       |
| Ecoli      | ecoli                   | 336      | 7          | 8       |
| Ionosphere | ionosphere              | 351      | 34         | 2       |
| Pima       | pima-indians-diabetes   | 768      | 8          | 2       |
| Wdbc       | breast-cancer-wisconsin | 569      | 30         | 2       |

- For studying the performance of CIDIM we have selected two criteria: the accuracy – the percentage of test examples that are correctly classified – and the size of the trees – number of leaves and number of nodes.
- For every experiment, the presented values have been obtained from a 10 x 10 fold cross-validation. Average and standard deviation values are given. To compare prediction accuracies, number of leaves and nodes, and evaluate the relevance of the differences, a statistical test must be made [13]. A t-test has been conducted using the results of the cited 10 x 10 fold cross-validation. Calculation of t-test values have been done using the statistical package R [14]. A difference is considered as significant if the significance level of the t-test is better than 0.05. In tables,  $\oplus$  indicates that the accuracy or the size of the tree induced are significantly better than those of CIDIM.  $\ominus$  signifies that the accuracy or the size of the tree induced are significantly worse than those of CIDIM. In addition to this comparisons, the best result for each experiment has been emphasized using numbers in boldface.

Once we have established the datasets we can continue talking about the experiments. In Table 2 we show results of ID3, C4.5 and CIDIM.

Having obtained the results shown in Table 2, we can observe two questions:

- Decision trees induced by CIDIM are usually significantly smaller than those induced by ID3 or C4.5 (there is only one exception in balance dataset).
- Accuracy reached by CIDIM is comparable to the accuracy reached by C4.5 and significantly better than accuracy reached by ID3.

**Table 2.** Comparison between ID3, C4.5 and CIDIM. Average values and standard deviations are given for number of leaves, number of nodes and accuracy. Significance tests are with respect to CIDIM

| Dataset    | Algorithm | Leaves                                                 | Nodes                                                  | Accuracy                                               |
|------------|-----------|--------------------------------------------------------|--------------------------------------------------------|--------------------------------------------------------|
| Balance    | ID3       | 387.92 $\pm$ 1.87 $\ominus$                            | 484.65 $\pm$ 2.33 $\ominus$                            | 38.19 $\pm$ 1.20 $\ominus$                             |
|            | C4.5      | <b>33.56 <math>\pm</math> 1.02 <math>\oplus</math></b> | <b>41.70 <math>\pm</math> 1.27 <math>\oplus</math></b> | 63.86 $\pm$ 1.40 $\ominus$                             |
|            | CIDIM     | 53.50 $\pm$ 3.10                                       | 78.54 $\pm$ 4.73                                       | <b>68.48 <math>\pm</math> 1.18</b>                     |
| Ecoli      | ID3       | 295.24 $\pm$ 2.45 $\ominus$                            | 344.28 $\pm$ 2.86 $\ominus$                            | 70.28 $\pm$ 0.36 $\ominus$                             |
|            | C4.5      | 49.00 $\pm$ 1.52 $\ominus$                             | 57.00 $\pm$ 1.78 $\ominus$                             | <b>77.33 <math>\pm</math> 0.90</b>                     |
|            | CIDIM     | <b>30.16 <math>\pm</math> 1.64</b>                     | <b>42.25 <math>\pm</math> 2.52</b>                     | 77.29 $\pm$ 0.80                                       |
| Ionosphere | ID3       | 107.80 $\pm$ 1.52 $\ominus$                            | 125.60 $\pm$ 1.78 $\ominus$                            | 85.61 $\pm$ 1.07 $\ominus$                             |
|            | C4.5      | 31.78 $\pm$ 1.13 $\ominus$                             | 36.91 $\pm$ 1.32 $\ominus$                             | 87.85 $\pm$ 0.51                                       |
|            | CIDIM     | <b>16.91 <math>\pm</math> 1.07</b>                     | <b>23.59 <math>\pm</math> 1.68</b>                     | <b>88.71 <math>\pm</math> 1.64</b>                     |
| Pima       | ID3       | 896.08 $\pm$ 6.32 $\ominus$                            | 1045.26 $\pm$ 7.38 $\ominus$                           | 61.30 $\pm$ 0.92 $\ominus$                             |
|            | C4.5      | 48.04 $\pm$ 5.34 $\ominus$                             | 55.88 $\pm$ 6.23 $\ominus$                             | <b>73.85 <math>\pm</math> 0.49 <math>\oplus</math></b> |
|            | CIDIM     | <b>21.81 <math>\pm</math> 2.33</b>                     | <b>29.37 <math>\pm</math> 3.43</b>                     | 73.29 $\pm$ 0.74                                       |
| Wdbc       | ID3       | 135.88 $\pm$ 2.21 $\ominus$                            | 158.36 $\pm$ 2.57 $\ominus$                            | 92.94 $\pm$ 0.71                                       |
|            | C4.5      | 56.98 $\pm$ 2.30 $\ominus$                             | 66.31 $\pm$ 2.68 $\ominus$                             | <b>93.01 <math>\pm</math> 0.55</b>                     |
|            | CIDIM     | <b>15.14 <math>\pm</math> 1.58</b>                     | <b>21.62 <math>\pm</math> 2.42</b>                     | 92.48 $\pm$ 1.05                                       |

$InitialError = 100$  - probability of majority class (fixed in every experiment)  
 $Error = 100$  - accuracy percentage  
 $Leaves$  = number of leaves in the induced tree  
 $Improvement = ((InitialError - Error)/InitialError) \times 100$   
 $Reduction = (ID3Leaves - Leaves)/ID3Leaves$   
                   where  $ID3Leaves$  is the number of leaves in tree induced by ID3  
 $Combined\ Index\ (C.I.) = Reduction \times Improvement/100$

**Fig. 3.** *CombinedIndex (C.I.)*

**Table 3.** Comparison between C4.5 and CIDIM using *C.I.* Significance tests are with respect to CIDIM

| Dataset    | Algorithm | Combined Index (C.I.)  |
|------------|-----------|------------------------|
| Balance    | C4.5      | 0.3012 ± 0.0239 ⊖      |
|            | CIDIM     | <b>0.3582 ± 0.0204</b> |
| Ecoli      | C4.5      | 0.5048 ± 0.0116 ⊖      |
|            | CIDIM     | <b>0.5429 ± 0.0114</b> |
| Ionosphere | C4.5      | 0.4665 ± 0.0092 ⊖      |
|            | CIDIM     | <b>0.5778 ± 0.0363</b> |
| Pima       | C4.5      | <b>0.2374 ± 0.0138</b> |
|            | CIDIM     | 0.2290 ± 0.0206        |
| Wdbc       | C4.5      | 0.4713 ± 0.0127 ⊖      |
|            | CIDIM     | <b>0.7091 ± 0.0253</b> |

Table 2 shows good performance of CIDIM in this experiments, but we want to establish a relation between the accuracy and the size of the trees. Both characteristics are important to determine the quality of the decision tree. Thus, we have defined a *Combined Index (C.I.)*. This index takes two references: the initial error in the dataset (*InitialError*) and the number of leaves in decision trees induced by ID3 (*ID3Leaves*). We use this values because they can be considered as the worst (or very bad) bounds for accuracy and complexity (or size) of an induced decision tree. Then we evaluate the improvement of both characteristics and combine them in an index. Thus, the definition of this index is shown in Figure 3. We have calculated the Table 3 using this *C.I.*

As we can see in Table 3 the results show that CIDIM is significantly better than C4.5 in almost every experiment and it is never significantly worse. This is possible because of the basic ideas we used to develop this algorithm.

## 5 Conclusions

This paper introduces CIDIM, an algorithm that induces small and accurate decision trees. We have compared results obtained with ID3, C4.5 and CIDIM using different datasets and we can note some questions. CIDIM induces small



trees (usually smaller than those induced by ID3 or C4.5). Accuracy achieved by CIDIM is comparable with the accuracies achieved by ID3 or C4.5, and some times it is better.

The complexity (or size) of a decision tree must be considered as a measurement almost as important as accuracy. Considering this we have proposed a combined index that relates both qualities and CIDIM performs better than C4.5 in almost every experiment. Thus, we can conclude that CIDIM has a good performance. However, it would be desirable to improve the method for dealing with continuous attributes. Our aim of improving CIDIM involves two issues:

- we are working to make that CIDIM could learn using continuous attributes. Thus, we will not have to discretize real attributes to nominal ordered attributes.
- we are also working to develop a multiple classifier system that uses CIDIM as the base classifier. The random dichotomic division of the training set makes CIDIM a randomized algorithm, what is very profitable to be used in a multiple classifier system.

## References

1. Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: Classification and Regression Trees. Wadsworth (1984)
2. Quinlan, J.R.: Induction of decision trees. *Machine Learning* **1** (1986) 81–106
3. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann (1993)
4. Utgoff, P.E., Berkman, N.C., Clouse, J.A.: Decision tree induction based on efficient tree restructuring. *Machine Learning* **29** (1997) 5–44
5. Folino, G., Pizzuti, C., Spezzano, G.: A cellular genetic programming approach to classification. In: Proceedings of the Genetic and Evolutionary Computation Conference. Volume 2., Morgan Kaufmann Publishers Inc. (1999) 1015–1020
6. Hyafil, L., Rivest, R.L.: Constructing optimal binary decision trees is np-complete. *Information Processing Letters* **5** (1976) 15–17
7. Gehrke, J., Ganti, V., Ramakrishnan, R., Loh, W.: Boat – optimistic decision tree construction. In: Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data, ACM Press (1999) 169–180
8. Ramos-Jiménez, G., Morales-Bueno, R., Villalba-Soria, A.: CIDIM. Control of induction by sample division methods. In: Proceedings of the International Conference on Artificial Intelligence (IC-AI'00), Las Vegas (2000) 1083–1087
9. Ruiz-Gómez, J., Ramos-Jiménez, G., Villalba-Soria, A.: Modelling based on rule induction learning. In: Computers and Computational Engineering in Control. World Scientific and Engineering Society Press, Greece (1999) 158–163
10. Jerez-Aragónés, J.M., Gómez-Ruiz, J.A., Ramos-Jiménez, G., noz Pérez, J.M., Alba-Conejo, E.: A combined neural network and decision trees model for prognosis of breast cancer relapse. *Artificial Intelligence in Medicine* **27** (2003) 45–63
11. Blake, C., Merz, C.J.: UCI repository of machine learning databases. University of California, Department of Information and Computer Science (2000)
12. Witten, I.H., Frank, E.: Data Mining: Practical machine learning tools with Java implementations. Morgan Kaufmann, San Francisco (2000)

13. Herrera, F., Hervás, C., Otero, J., Sánchez, L.: Un estudio empírico preliminar sobre los tests estadísticos más habituales en el aprendizaje automático. In: Tendencias de la Minería de Datos en España. Red Española de Minería de Datos (2004) 403–412
14. R Development Core Team: R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. (2004) 3-900051-07-0. <http://www.R-project.org>.

# An Approach to Reduce the Cost of Evaluation in Evolutionary Learning<sup>\*</sup>

Raúl Giráldez, Norberto Díaz-Díaz, and Isabel Nepomuceno,  
and Jesús S. Aguilar-Ruiz

Department of Computer Science, University of Seville,  
Avenida Reina Mercedes s/n, 41012 Sevilla, Spain  
{giralde, ndiaz, isabel, aguilar}@lsi.us.es

**Abstract.** The supervised learning methods applying evolutionary algorithms to generate knowledge model are extremely costly in time and space. Fundamentally, this high computational cost is fundamentally due to the evaluation process that needs to go through the whole datasets to assess their goodness of the genetic individuals. Often, this process carries out some redundant operations which can be avoided. In this paper, we present an example reduction method to reduce the computational cost of the evolutionary learning algorithms by means of extraction, storage and processing only the useful information in the evaluation process.

## 1 Introduction

Machine Learning is used when we want to build a knowledge model from a training dataset and predict the outcome of a new unseen instance. When the class of the training data is known, we work in the Supervised Learning field. There are several methods and algorithms in the specific literature that extract the inherent knowledge to a set of labelled data. A large number of these methods (HIDER [1, 11], CN2 [6], RISE [9], OC1 [15], GABIL [7], GASSIST [3], GIL [13], SIA [17], ECL [8], etc.) use probabilistic algorithms to search solutions that able to model the behaviour of data. When the learning process is carried out by applying techniques of evolutionary computation, particularly evolutionary algorithms, it is called evolutionary learning, which is the framework of our approach.

The evolutionary learning methods usually evaluate the rules directly from the database. That is to explore such database sequentially, taking each of the examples and testing the quality of the rule through the correct classification of those examples. We can see, therefore, that the learning process of these systems is very costly in terms of time and space. Some approaches is focused on improving the learning process in order to reduce its computational cost by applying methods of incremental learning and *windowing* [4] or techniques of

---

<sup>\*</sup> This research was supported by the Spanish Research Agency CICYT and European FEDER Funds, under grants TIN2004-00159 and TIN2004-06689-C03-03.

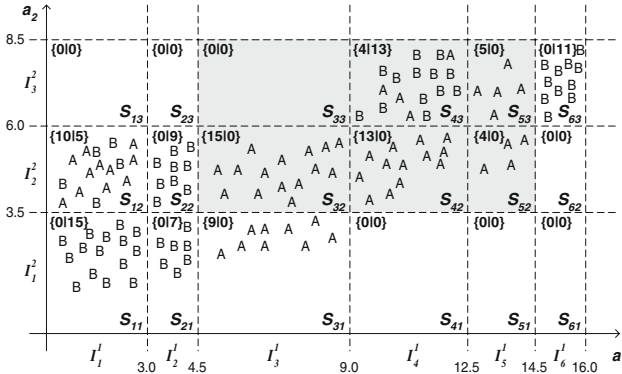


Fig. 1. Example

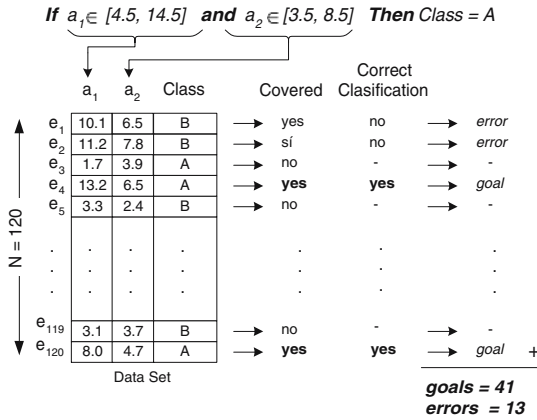
scalability [16]. Nevertheless, an appropriated organization of the information can also contribute to a reduction of the computing time. In this sense, the structure named EES [12] allows us to process only those examples, the values of which are covered by such rule will be processed, and not the totality of the database. However, although this solution reduces the computational cost, the use of the EES reflects a redundant process of the data for the rules sharing regions in the space.

The aim of this research is to avoid repetitive counting of examples during the evaluation process of the individuals of the population. In this paper, we propose a preprocessing method which extracts the useful information from the data to be used in the evaluation and stores it in a format that allows an efficient access of such data.

## 2 Motivation

One of the critical factors related to applying the evolutionary algorithms in supervised learning is the evaluation of the individuals of the population. Usually, each individual represents one or several rules as potential solutions to the problem. In this work, we assume that each individual codifies only one rule, however, our approach can be easily adapted to individual with variable length. The evaluation function measures the goodness of each individual of the population. This goodness is based on the number of goals (number of examples correctly classified) and errors (number of misclassified examples) that the encoded rule has during the classification of the examples from the dataset.

GABIL and GIL are named *concept learner*, since they handle with discrete attributes exclusively. Other advanced tools, like HIDER, GASSIST or ECL can treat continuous and discrete attributes thanks to a discretization algorithm that diminishes the cardinality of the set of values of the continuous attribute. Thus, the rules can only establish conditions using a finite set of intervals. We



**Fig. 2.** Linear Evaluation

name these *atomic intervals*, because once the intervals are obtained, they can not be split. The discretization process turns the initial search space, that it is theoretically infinite, into a finite space of solutions. Figure 1 shows an example of a dataset with 120 examples, two classes (A and B) and two continuous attributes. The discretization method has obtained 6 intervals ( $I_i^1$ ) for  $At_1$  and 3 intervals ( $I_j^2$ ) for  $At_2$ . Each pair of intervals ( $I_i^1, I_j^2$ ) defines a subspace or region ( $\mathcal{S}_{ij}$ ). The values in brackets  $\{\varepsilon_A | \varepsilon_B\}$  represent the number of examples of each class in the correspondent region. These subspaces can be linked in a rule, but they can not be split, since their decision bounds are given by atomic intervals. Consequently, these regions are called *atomic subspaces*.

Evaluation by means of a linear search processes each and every one of the examples in the database independently of the conditions established by the rule. The computational cost of a individual evaluation<sup>1</sup> is  $\Theta(Nm)$ , where  $N$  is the number of examples and  $m$  is the number of attributes in the database. For example, the rule

$$\text{If } a_1 \in [4.5, 14.5] \text{ and } a_2 \in [3.5, 8.5] \Rightarrow \text{Class} = A$$

is represented by the shaded area in Figure 1 and it means that if an example belongs to the subspace  $\{I_3^1 \cup I_4^1 \cup I_5^1\} \cap \{I_2^2 \cup I_3^2\}$ , such example must be classified as A. In order to count the goals and errors for this rule, each example of the dataset is analysed. If the example is covered, that is it fulfills the conditions of the rule, then the class is compared, as Figure 2 illustrates. In this case, the rule has 41 goals and 13 errors. This process is repeated for each individual, which means an unnecessary computational cost due to two aspects principally:

<sup>1</sup> Evaluation cost for only one individual.

1. Redundant count of examples for those space areas shared by some rules.
2. The whole space exploration to evaluate rules which only cover a part of such space.

In general, since the cost of the individual evaluation is very high, normally it is tried to reduce some of the two parameters which take part in this:  $N$  and  $m$ . The techniques that reduce the number of attributes  $m$  are commonly named *feature selection methods* [14], and their goal is to remove those attributes which are irrelevant and/or harmful for learning. In other way, the example pruning methods are included into the *instance editing techniques* [18], and they are focused on reduce the size of dataset ( $N$ ). This work is framed in these last techniques. We propose an instance reduction method that benefits from the methodology followed by the typical evolutionary algorithms for rule discovery.

### 3 Example Reduction Method

As we mentioned before, the atomic subspaces can not be divided. Therefore, although examples which belong to a same subspace can be different syntactically, from the point of view of the individual evaluation they are semantically similar. This fact makes possible to count how many examples of each class coexist in each atomic subspace and to store these values for its later utilization in evaluation process. This idea is put into practice in a novel editing method that removes all those examples that result redundant for evaluation.

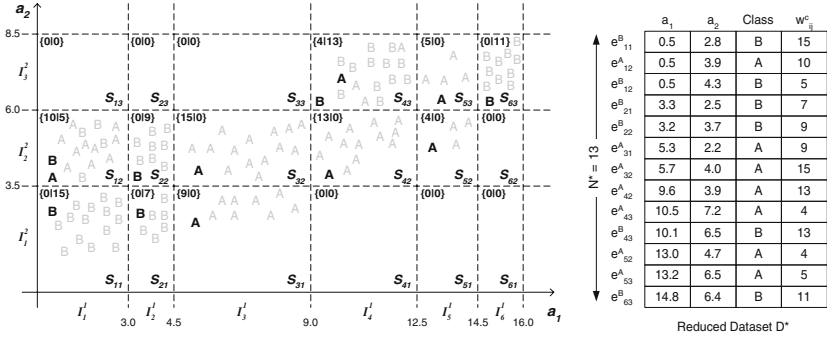
To explain our proposal in a clear way, we are going to use the example showed in Figures 1 and 2, to later generalize the solution to data collections with any kind of attributes and greater number of classes.

#### 3.1 Algorithm

The aim of the editing method is to reduce the number of examples of the dataset  $\mathcal{D}$  to obtain a subset  $\mathcal{D}^* \subseteq \mathcal{D}$  which contains the same knowledge that  $\mathcal{D}$ , but with a smaller number of examples  $N^*$ . Initially, we begin from a dataset  $\mathcal{D}$  with a number of examples  $N$ , where continuous attributes have already been discretized. From this discretization results a set of atomic intervals per attribute that define the atomic subspaces. Each example  $e = (a_1, a_2, \dots, a_m | c)$  is made up by a collection of attributes and a class (v.g,  $e_1 = (10.1, 6.5 | B)$ ). For each atomic subspace  $\mathcal{S}_{ij}$ , the instances of each class  $\{\varepsilon_A | \varepsilon_B\}$  are counted and they are chosen as many representative examples ( $e_{ij}^c$ ) as different class coexist in the subspace. These representative examples<sup>2</sup> are added to the reduced dataset  $\mathcal{D}^*$ . Each representative example have the same form that an original example, but we add a weight  $\omega_{ij}^c$  equal to the  $\varepsilon_c$  which counts the instances of the class  $c$  in the subspace  $\mathcal{S}_{ij}$ . The regions with some  $\varepsilon_c$  equal to 0 do not have representative in  $\mathcal{D}^*$  for the class  $c$ . For example, in Figure 1,  $e_{43}^A = (10.5, 7.2 | A, 4)$  and

---

<sup>2</sup> Although these representatives not have to coincide with some original example, for simplicity, we choose the first which is in the data set.



**Fig. 3.** Reduction for example of Figures 1 and 2

$e_{43}^A = (10.1, 6.5 |B, 13)$  represent the subspace  $\mathcal{S}_{43}$ , whereas  $e_{53}^A = (13.2, 6.5 |A, 5)$  is the only representative for the subspace  $\mathcal{S}_{53}$ . For the empty regions, like  $\mathcal{S}_{13}$ ,  $\mathcal{D}^*$  does not contain any example.

Figure 3 shows the result obtained by this editing method for the example explained in Figure 1. Representative examples of each atomic subspace are displayed in bold type, whereas removed examples appear with a lighter color. The reduced dataset ( $\mathcal{D}^*$ ) are shown to the right and it will be used to the evaluate process. As we can see, the original dataset, with  $N = 120$  examples, has been replaced by  $\mathcal{D}^*$ , with  $N^* = 13$  weighted examples.

Once editing process has finished, the evaluation of the individuals can be carried out in a linear way, as it was shown by Figure 2, but now, the number of examples smaller. Note that it is necessary to take into account the examples' weight when the goals and errors are counted, since each  $e_{ij}^c \in \mathcal{D}^*$  represents to  $\omega_{ij}^c$  examples in  $\mathcal{D}$ . This method solves only the first of the problems mentioned in Section 2, since it continues being needed a linear search through  $\mathcal{D}^*$ . However, our approach is very simple to apply and it achieves satisfactory experimental results, as Section 4 shows.

Notice that the generalization of the methods for  $k$  class and  $m$  attribute is trivial. Simply we would have a collection of class accountant  $\{\varepsilon_{c_1} | \varepsilon_{c_2} | \dots | \varepsilon_{c_k}\}$ ; and a indexes collection to denote a subspace ( $\mathcal{S}_{i_1 \dots i_m}$ ) or example ( $e_{i_1 \dots i_m}^c$ ).

### 3.2 Discrete Attributes

By applying our proposal for data set with discrete attributes is similar to the previous one for continue attributes, although we should emphasize some important peculiarities.

Continuous attributes, though they are discretized, often define a space much more complex than the discrete ones, principally due to two reasons: first, the number of intervals is normally larger in real applications, which multiplies the number of subspaces; and second, the regions usually include more than one example, that is, there are examples with the same semantical meaning. Thus, our proposal has a priori more justification when the dataset contains continuous attributes.

When the dataset contains only discrete attributes and there are not repeated examples, each atomic subspace contains at the most one example only, that is, the editing process would not produce reduction in the number of examples. However, although the multiplicity of examples can look like not much habitual, it is relatively common. For example, the application of some feature selection method can cause that some examples are identical if those attributes that distinguish them was eliminated. Another clear example is given when there is noise in the dataset. In this case we can remove the repeated examples in a same atomic subspace by setting  $w_{ij}^c$  to 1. However, this solution is not advisable because we could be eliminating useful information for the learning.

Therefore, the our approach is favourable whenever the dataset contains similar examples from the point of view of the learning process, otherwise it does not reduce the size of data. Anyway, our method does not cause a significant increment in the computational cost with regard to the evolutionary algorithm, and we advise its use when there is no previous information about the multiplicity of dates.

## 4 Empirical Results

In order to show the reduction of the computational cost of the evaluation process, we have designed the following experiments with some datasets from UCI Repository [5]. The the evolutionary tool used was HIDER [1, 11], that generates a set of hierarchical decision rules from a labeled dataset. This tool uses its own discretization method, named USD [10], before running the evolutionary algorithm that obtains the rules. Thus, the editing method must be applied after the discretization and before the learning process. This algorithm required some changes, though minimal, to adapt the evaluation of the individuals to the new dataset with weight. To check that the editing does not damage the accuracy of the rules, a 10-fold cross-validation was achieved with each dataset. In this sense, it is important to point out that the accuracy and complexity of the knowledge models resulted similar by using the editing method and without it.

The datasets used were: *Breast Cancer*, *Bupa*, *Cleveland*, *Glass*, *Hayes Roth*, *Heart*, *Hepatitis*, *Horse Colic*, *Iris*, *Led7*, *Pima Diabetes*, *Tic Tac Toe*, *Vote* and *Zoo*. HIDER was run for each database by using the original dataset ( $D$ ) and the reduced dataset ( $D^*$ ) in order to compare the computational cost in time and space. Thus, the number of examples was reduced for 8 cases and kept for the other 6 datasets. Logically, for those last ones, the cost shown a light increase by using  $D^*$ , although this never exceeded the 5% with respect to the inverted time by using  $D$ . Among to the 8 databases where the editing had a favourable effect, the reduction of the dataset was higher than 20% for 6 cases. Table 1 shows this results. The five first columns show the features of each database: name, number of examples, number of attributes, type of attributes (continuous or discrete) and number of class, respectively. The next column is the number of examples after the editing process. Finally, the three last columns give the relative cost concerned to the runtime, the evaluation time and the space used. This values is obtained by dividing, in each case, the cost with  $D^*$  by the cost with  $D$ . The last row shows each previous relative cost on average.



**Table 1.** Results

| Dataset          | Features |     |      |          | Editing | Relative Cost |                 |       |
|------------------|----------|-----|------|----------|---------|---------------|-----------------|-------|
|                  | $N$      | $m$ | Type | #Classes | $N^*$   | Runtime       | Evaluation Time | Space |
| Breast Cancer W. | 699      | 9   | C    | 2        | 263     | 0.44          | 0.44            | 0.38  |
| Hayes Roth       | 132      | 4   | C    | 3        | 28      | 0.33          | 0.23            | 0.21  |
| Iris             | 150      | 4   | C    | 3        | 70      | 0.59          | 0.57            | 0.47  |
| Led7             | 3200     | 7   | D    | 10       | 336     | 0.31          | 0.13            | 0.11  |
| Vote             | 435      | 16  | D    | 2        | 342     | 0.82          | 0.74            | 0.79  |
| Zoo              | 101      | 16  | D    | 7        | 59      | 0.49          | 0.38            | 0.58  |
| Average          |          |     |      |          |         | 0.49          | 0.41            | 0.42  |

By observing the average results, the size of dataset is reduced to 42%. This caused a decrease in the evaluation time to 41%. The rule-learning methods that EAs use invest approximately 85% of their time in evaluating the individuals (the mean of the executions of a 10-fold cross-validation with 20 UCI Repository databases [2]). Therefore, this proves the importance of the evaluation with regard to the efficiency of the algorithm. We can deduce that the editing method does not add significant computational cost to the algorithm, so that the runtime is very similar to the evaluation time. In short, for the dataset of table 1, we concludes that the our approach speeds up the learning process, by using less than a half of computational resources, on average.

## 5 Conclusions and Future Works

In this paper we present new editing method that reduces the evaluation cost of individuals in evolutionary algorithms for supervised learning. This method identifies those regions of attribute space that are indivisible during the learning process, and, furthermore, the method extracts the useful information from each of them. The method takes advantage of those examples which share the same region are identical from the point of view of the learning.

After the empirical experiments, we conclude that our proposal produces a reduction of the computational cost associated to the evaluation of individuals during the learning, in time and space. This reduction is proportional to the reduction in the number of the examples that are the result from the editing process. This no affects to the quality of the knowledge model obtained by the learning algorithm. If the method does reduce the number of examples, our method does not cause a significant increment in the computational cost with regard to the evolutionary algorithm.

## References

1. J. S. Aguilar–Ruiz, J. C. Riquelme and M. Toro. Evolutionary Learning of Hierarchical Decision Rules. *IEEE Transactions on Systems, Man and Cybernetics – Part B*, **33**(2)(2003), 324–331.

2. J. S. Aguilar. *Discovering Hierarchical Decision Rules with Evolutionary Algorithms in Supervised Learning*. PhD thesis, University de Seville, 2001.
3. J. Bacardit and J. M. Garrell. Evolving multiple discretizations with adaptive intervals for a Pittsburgh Rule-Based Learning Classifier System. *Genetic and Evolutionary Computation Conference - GECCO 2003*. Lecture Notes in Computer Science 2724, pp. 1818–1831, Springer-Verlag, 2003.
4. J. Bacardit y J. M. Garrell. Incremental Learning for Pittsburgh Approach Classifier Systems. *2nd. Spanish Conference on Metaheuristics and Evolutionary Algorithms (MAEB'03)*, pp. 303–311. Gijón, Spain, 2003.
5. C. L. Blake and C. J. Merz. UCI Repository of machine learning databases [<http://www.ics.uci.edu/mlearn/MLRepository.html>]. Irvine, CA: University of California, Department of Information and Computer Science, 1998.
6. P. Clark and R. Boswell. Rule induction with cn2: Some recent improvements. In *Machine Learning: Proceedings of the Fifth European Conference (EWSL-91)*, pages 151–163, 1991.
7. K. A. DeJong, W. M. Spears and D. F. Gordon. Using genetic algorithms for concept learning. *Machine Learning*, 1(13):161–188, 1993.
8. F. Divina and E. Marchiori, Evolutionary Concept Learning, *Genetic and Evolutionary Computation Conference - GECCO 2002*, W.B. Langdon et al. eds, Morgan Kaufmann, NY, USA, 2002, pp. 343–350.
9. P. Domingos. Rule induction and instance-based learning: A unified approach. In *Proceedings of International Joint Conference on Artificial Intelligence*, 1995.
10. R. Giráldez, J. S. Aguilar-Ruiz, J. C. Riquelme y D. Mateos Discretization Oriented to Decision Rule Generation, *In Proceedings of International Conference on Knowledge-Based Intelligent Information & Engineering Systems*, pp. 275–279, IOS Press, Crema, Italy, 2002.
11. R. Giráldez, J. S. Aguilar-Ruiz and J. C. Riquelme. Natural Coding: A More Efficient Representation for Evolutionary Learning. *Genetic and Evolutionary Computation Conference - GECCO 2003*, pp. 279–290. Chicago, USA, 2003.
12. R. Giráldez and J. S. Aguilar-Ruiz and J. C. Riquelme. Knowledge-based Fast Evaluation for Evolutionary Learning, *IEEE Transactions on Systems, Man & Cybernetics - Part C*, (in press), 2005.
13. C. Z. Janikow. A knowledge-intensive genetic algorithm for supervised learning. *Machine Learning*, 1(13):169–228, 1993.
14. H. Liu and H. Motoda *Feature Selection for Knowledge Discovery and Data Mining*. Kluwer Academic, 1998.
15. S. K. Murthy, S. Kasif, and S. Salzberg. A system for induction of oblique decision trees. *Journal of Artificial Intelligence Research*, 1994.
16. K. Shim. *SIGKDD Explorations*. December 2000. Volume 2, Issue 2.
17. G. Venturini. SIA: a supervised inductive algorithm with genetic search for learning attributes based concepts. In *Proceedings of European Conference on Machine Learning*, pages 281–296, 1993.
18. D. R. Wilson and T. R. Martinez, Reduction Techniques for Instance-Based Learning Algorithms. *Machine Learning*, 38(3):257–286, 2000.

# Efficient Design of Fixed Point Digital FIR Filters by Using Differential Evolution Algorithm

Nurhan Karaboğa and Bahadır Çetinkaya

Erciyes University, Department of Electronic Engineering,  
38039, Melikgazi, Kayseri, Turkey  
{nurhan\_k, cetinkaya}@erciyes.edu.tr

**Abstract.** Differential Evolution (DE) algorithm is a new heuristic approach which has been proposed particularly for numeric optimization problems. It is a population based algorithm like genetic algorithms using the similar operators; crossover, mutation and selection. In this work, DE algorithm has been applied to the design of fixed point digital Finite Impuls Response (FIR) filters and its performance has been compared to that of Genetic Algorithm (GA) and Least Squares Algorithm (LSQ).

## 1 Introduction

Heuristic optimization algorithms such as genetic algorithm, tabu search and simulated annealing algorithms have been widely used in the optimal design of digital filters. When considering global optimization methods for digital filter design, the GA seems to have attracted considerable attention. Filters designed by GA have the potential of obtaining near global optimum solution [1,2]. However, they are not so successful at determining local minimum in terms of convergence speed. In order to overcome this disadvantage of GA in numeric optimization problems, Differential Evolution algorithm has been introduced by Storn and Price [3]. Differential Evolution algorithm is a new heuristic approach mainly having three advantages; finding the true global minimum of a multi modal search space regardless of the initial parameter values, fast convergence, and using a few control parameters. DE algorithm is a population based algorithm like genetic algorithms using the similar operators; crossover, mutation and selection. The studies on the design of optimal digital filters by using DE algorithm are not as common as GA. In literature, there are only a few studies related to the application of DE algorithm to the digital filter design [4-6].

For the filter parameters the choice of the appropriate binary representation, i.e., fixed point or power of two representation, can improve the performance of the filter designed. The approximation of FIR digital filters with fixed point representation coefficients provides a larger coefficients space than that of power of two for the same wordlength so it yields a better performance. In this work, the performance comparison of the design methods based on Differential evolution and Genetic algorithms is presented for digital FIR filters with fixed point coefficients since DE algorithm is very similar to, but much simpler than GA. Also, Least Squares design of FIR filters is realized for fixed point coefficient coding scheme and the results are com-

pared. The paper is organized as follows. Section 2 briefly describes a basic genetic algorithm. Section 3 presents a basic differential evolution algorithm. Section 4 describes the application of algorithms to the design of digital FIR filters. Section 5 presents the simulation results and discussion.

## 2 Basic Genetic Algorithm

The genetic algorithm is an artificial genetic system based on the process of natural selection and genetic operators. It is also a heuristic algorithm which tries to find the optimal results by decreasing the value of objective function (error function) continuously [7]. A simplified GA cycle is shown in Fig.1.

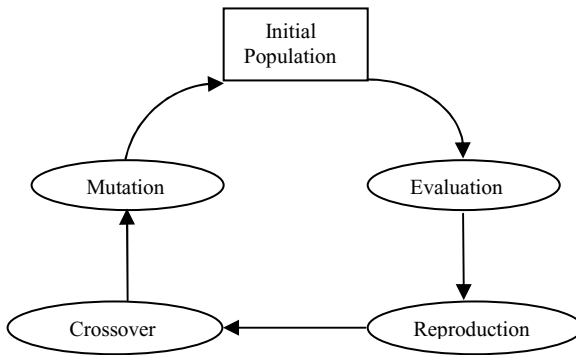


Fig. 1. A simplified GA cycle

Initial population consists of a collection of chromosomes [8]. In practice these chromosomes represent a set of solutions for the problem. The chromosome which produces the minimum error function value represents the best solution. The chromosomes which represent the better solutions are selected by the reproduction operator and then sent to the crossover operation. In this operation, two new chromosomes are produced from two chromosomes existing in the population. A common point in the selected chromosomes is randomly chosen and their corresponding digits are exchanged. Thus, new chromosomes which represent the new solutions are produced. The next operator is mutation. Generally, over a period of several generations, the genes tend to become more and more homogenous. Therefore, many chromosomes can not continue to evolve before they reach their optimal state. In the mutation process, some bits of the chromosomes mutate randomly. Namely, certain digits will be altered from either '0' to '1' or '1' to '0' in binary encoding [9].

In addition to the operators mentioned above GA also contains 'Elite' operator. By means of Elite operator, the best solution is always kept. In the evaluation process, the solutions in the population are evaluated and a fitness value associated with each solution is calculated. These fitness values are used by the selection operator. Roulette Wheel method is employed for the selection process.

### 3 Differential Evolution Algorithm

An optimization task consisting of  $D$  parameters can be represented by a  $D$  dimensional vector. In DE, a population of  $NP$  solution vectors is randomly created at the start where  $NP$  is a control parameter called population size. Then, the population is successfully improved by applying mutation, crossover and selection operators.

The main steps of a basic DE algorithm is given below:

*Initialization*  
*Evolution*  
*Repeat*  
     *Mutation*  
     *Recombination*  
     *Evaluation*  
     *Selection*  
*Until (termination criteria are met)*

#### 3.1 Mutation

For each target vector  $x_{i,G}$ , a mutant vector is produced by

$$v_{i,G+1} = x_{i,G} + K \cdot (x_{r_1,G} - x_{i,G}) + F \cdot (x_{r_2,G} - x_{r_3,G}) \quad (1)$$

where  $i, r_1, r_2, r_3 \in \{1, 2, \dots, NP\}$  that randomly chosen and must be different from each other. In Eq (1),  $F$  is the scaling factor  $\in [0, 2]$  affecting on difference vector  $(x_{r_2,G} - x_{r_3,G})$ ,  $K$  is the combination factor.

#### 3.2 Crossover

The parent vector is mixed with the mutated vector to produce trial vector  $u_{ji,G+1}$

$$u_{ji,G+1} = \begin{cases} v_{ji,G+1} & \text{if } (rnd_j \leq CR) \text{ or } j = rn_i \\ x_{ji,G} & \text{if } (rnd_j > CR) \text{ and } j \neq rn_i \end{cases} \quad (2)$$

where  $j = 1, 2, \dots, D$ ;  $rnd_j \in [0, 1]$  random number;  $CR$  stands for the crossover constant  $\in [0, 1]$  and  $rn_i \in (1, 2, \dots, D)$  randomly chosen index.

#### 3.3 Selection

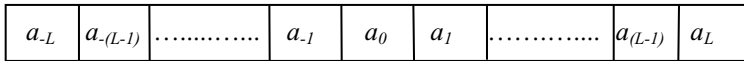
Performance of the trial vector and its parent is compared and the better one is selected. This method is usually named greedy selection. All solutions have the same chance of being selected as parents without dependence of their fitness value. The better one of the trial solution and its parent wins the competition providing significant advantage of converging performance over genetic algorithms.

### 4 Application of Algorithms to the Problem

The transfer function of a FIR filter is given by Equation (3),

$$H(z) = \sum_{n=0}^N a_n z^{-n} \tag{3}$$

$a_n$  represents the filter parameters to be determined in the design process and  $N$  represents the polynomial order of the function. In the design of FIR filter by using GA or DE, firstly the solutions must be represented in the string form of parameters. The representation scheme used in this work is shown in Fig. 2.



**Fig. 2.** Representation of solutions in string form

In order to evaluate the strings representing possible FIR filters, Least Mean Square (LMS) error is used. The strings which have higher evaluation values represent the better filters, i.e. the filters with better frequency response. The expression of LMS function is given below:

$$LMS = \left\{ \sum_f [ |H_I(f)| - |H_D(f)| ]^2 \right\}^{\frac{1}{2}} \tag{4}$$

where  $H_I(f)$  is the magnitude response of the ideal filter and  $H_D(f)$  is the magnitude response of the designed filter. The fitness function to be maximized is defined depending on LMS function as the following,

$$Fitness_i = \frac{1}{LMS_i} \tag{5}$$

where  $fitness_i$  is the fitness value of the solution  $i$  and  $LMS_i$  is the LMS error value calculated when the solution  $i$  is used for the filter.

#### 4.1 Fixed Point Coding Scheme

Among the possible quantizations, using the fixed point representation as a representation scheme provides a larger coefficient space for the same coefficient wordlengths and this yields a better performance in most cases. The coefficient space in this representation is defined in a ‘b’ bits wordlength as follows:

$$\text{Fixed point} : D = \left\{ a : a = s \cdot \sum_{k=1}^{b-1} c_k 2^{-k} \quad , c_k \in \{0,1\} \right\} \tag{6}$$

$$: \left\{ s \in \{-1,1\} \right\}$$

### 5 Simulation Results

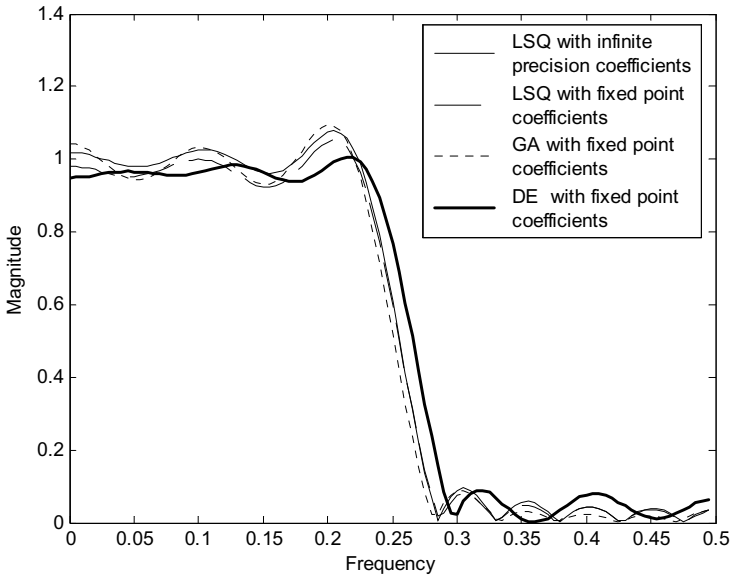
The simulations are realized for the symmetric FIR filter with the order of 20. In the simulations, the sampling frequency was chosen as  $f_s = 1Hz$ . Also, for all the simulations the sampling number was taken as 100. A lowpass filter with the following characteristic has been designed.

$$H(f) = \begin{cases} 1, & 0 \leq f \leq 0.25 \\ 0, & 0.3 \leq f \leq 0.5 \end{cases} \quad (7)$$

The control parameter values of DE and GA used in this work are given in Table 1.

**Table 1.** Control parameter values of DE and GA algorithms

| Differential Evolution Algorithm                                       | Genetic Algorithm              |
|------------------------------------------------------------------------|--------------------------------|
| <i>Population size = 50</i>                                            | <i>Population size = 50</i>    |
| <i>Crossover rate = 0.8</i>                                            | <i>Crossover rate = 0.8</i>    |
| <i>Scaling factor (F) = 0.8</i><br><i>Combination factor (K) = 0.8</i> | <i>Mutation rate = 0.01</i>    |
| <i>Generation number = 500</i>                                         | <i>Generation number = 500</i> |



**Fig. 3.** Magnitude responses of filters with fixed point coefficients

The magnitude responses of the designed digital FIR filters with the fixed point coefficients by using DE, GA, and LSQ algorithms for the filter of 20<sup>th</sup> order are demonstrated in Fig. 3.

The coefficients relevant to the designs demonstrated in Fig. 3., are given in Table 2.

**Table 2.** The coefficients obtained in the designs

|       | Least Square Design<br>(Infinite Precision<br>Coefficients) | Least Square<br>Design<br>(Fixed Point ) | Genetic<br>Algorithm<br>(Fixed Point) | Differential<br>Evolution Algorithm<br>(Fixed Point) |
|-------|-------------------------------------------------------------|------------------------------------------|---------------------------------------|------------------------------------------------------|
| a(1)  | -0.00962590531221                                           | -2                                       | 1                                     | -5                                                   |
| a(2)  | 0.03273214110702                                            | 8                                        | 9                                     | 4                                                    |
| a(3)  | 0.00983774777526                                            | 2                                        | 1                                     | 6                                                    |
| a(4)  | -0.04340235407583                                           | -11                                      | -11                                   | -7                                                   |
| a(5)  | -0.01000454673447                                           | -3                                       | -1                                    | -7                                                   |
| a(6)  | 0.06217196582458                                            | 15                                       | 17                                    | 12                                                   |
| a(7)  | 0.01012478623494                                            | 3                                        | 1                                     | 8                                                    |
| a(8)  | -0.10520482879053                                           | -26                                      | -27                                   | -25                                                  |
| a(9)  | -0.01019737132958                                           | -3                                       | -1                                    | -9                                                   |
| a(10) | 0.31800965173022                                            | 79                                       | 81                                    | 81                                                   |
| a(11) | 0.51022164017166                                            | 127                                      | 127                                   | 127                                                  |
| a(12) | 0.31800965173022                                            | 79                                       | 81                                    | 81                                                   |
| a(13) | -0.01019737132958                                           | -3                                       | -1                                    | -9                                                   |
| a(14) | -0.10520482879053                                           | -26                                      | -27                                   | -25                                                  |
| a(15) | 0.01012478623494                                            | 3                                        | 1                                     | 8                                                    |
| a(16) | 0.06217196582458                                            | 15                                       | 17                                    | 12                                                   |
| a(17) | -0.01000454673447                                           | -3                                       | -1                                    | -7                                                   |
| a(18) | -0.04340235407583                                           | -11                                      | -11                                   | -7                                                   |
| a(19) | 0.00983774777526                                            | 2                                        | 1                                     | 6                                                    |
| a(20) | 0.03273214110702                                            | 8                                        | 9                                     | 4                                                    |
| a(21) | -0.00962590531221                                           | -2                                       | 1                                     | -5                                                   |

In the case of LSQ, firstly the filter is designed with infinite precision coefficients and then the coefficients are converted to the fixed point form. In the case of GA and DE the coefficients are directly produced as fixed point coefficients.

The maximum deviation rate  $\delta$  represents the deviation from the desired response in the passband and stopband regions and is usually used for the evaluation of the designed filter's performance. In Table 3, the  $\delta$  values for the bands are given for all design methods.

**Table 3.** Maximum deviation rates

|                     | Least Squares Design<br>(Infinite Precision<br>Coefficients) | Least Squares<br>Design<br>(Fixed Point ) | Genetic<br>Algorithm<br>(Fixed Point) | Differential<br>Evolution Algorithm<br>(Fixed Point) |
|---------------------|--------------------------------------------------------------|-------------------------------------------|---------------------------------------|------------------------------------------------------|
| $\delta_{passband}$ | 0.39019760                                                   | 0.41524920                                | 0.48050                               | 0.23050                                              |
| $\delta_{stopband}$ | 0.09032543                                                   | 0.09087496                                | 0.08636                               | 0.08949                                              |



As seen from Table. 3., DE algorithm provides the minimum deviation from the desired response in the passband, namely, it provides the best response in terms of deviation. In the stopband region, all the algorithms show the similar performances but the performance of GA is slightly better than the others.

In Table 4, the LMS error values for the filters designed by GA and DE are given. From the table it is clear that the performance of DE is significantly better than that of GA in terms of LMS error.

**Table 4.** LMS error values for different filter orders

| Algorithms | LMS error value |
|------------|-----------------|
| DE         | 0.519923        |
| GA         | 1.005820        |

## 6 Conclusion

Differential evolution, genetic and least squares algorithms have been applied to the design of digital FIR filters with fixed point coefficients. Although DE algorithm has more simple structure than GA, for the same population size and generation number, DE algorithm demonstrates a better performance in terms of magnitude response and hence LMS error. Also, in the simulations, it is seen that DE is significantly faster than GA for finding the optimum filter. Consequently, DE algorithm has successfully designed fixed point digital FIR filter with desired magnitude response and also found optimal filter much quicker than GA. Therefore, DE can be successfully used in fixed point digital FIR filter design.

## References

1. Chen, S. IIR Model Identification Using Batch-Recursive Adaptive Simulated Annealing Algorithm. In Proceed. of 6th Annual Chinese Automation and Computer Science Conf., (2000) 151-155
2. Mastorakis, N. E., Gonos, I. F., Swamy, M.N.S.: Design of Two Dimensional Recursive Filters Using Genetic Algorithms. IEEE Transaction on Circ. and Syst. I-Fundamental Theory and Applications. 50 (2003) 634-639
3. Storn, R., Price, K.: Differential Evolution - A Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces. Technical Report TR - 95 - 012, ICSI. (1995) ftp.icsi.berkeley.edu.
4. Storn, R.: Differential evolution design of an IIR- filter with requirements for magnitude and group delay. IEEE Int. Conf. on Evolutionary Computation, Japan (1996) 268-273
5. Karaboga, N.: Digital IIR Filter Design using Differential Evolution Algorithm. EUROSIP J. on Applied Sig. Process., (to be appear) (2005)

6. Karaboğa, N., Çetinkaya, B.: Performance Comparison of Genetic and Differential Evolution Algorithms for Digital FIR Filter Design. Lecture Notes in Computer Science, Vol. 3261. Springer-Verlag, (2004) 482-489
7. Karaboğa, N., Çetinkaya, B.: Performance Comparison of Genetic Algorithm based Design Methods of Digital Filters with Optimal Magnitude Response and Minimum Phase. The 46th IEEE Midwest Sym. on Circ. and Syst. (2003) (Accepted, In Press).
8. Lee, A., Ahmadi, M., Jullien, G. A., Miller, W. C., Lashkari R. S.: Design of 1-D FIR Filters with Genetic Algorithms. IEEE Int. Symp. on Circ. and Syst. (1999) 295-298
9. Xiaomin, M., Yixian, Y. : Optimal Design of FIR Digital Filter using Genetic Algorithm. The J. China Univ. Posts Telecom. 5 (1998) 12-16

# Manifold Constrained Finite Gaussian Mixtures

Cédric Archambeau\* and Michel Verleysen\*\*

Machine Learning Group - Université catholique de Louvain,  
Place du Levant 3, B-1348 Louvain-la-Neuve, Belgium  
{archambeau, verleysen}@dice.ucl.ac.be

**Abstract.** In many practical applications, the data is organized along a manifold of lower dimension than the dimension of the embedding space. This additional information can be used when learning the model parameters of Gaussian mixtures. Based on a mismatch measure between the Euclidian and the geodesic distance, manifold constrained responsibilities are introduced. Experiments in density estimation show that manifold Gaussian mixtures outperform ordinary Gaussian mixtures.

## 1 Introduction

Probability density estimation is a fundamental concept in unsupervised learning and knowledge discovery. In general, density estimation is performed regardless of the intrinsic geometric structure of the data. However, they are concentrated on lower dimensional manifolds embedded in the higher dimensional input space in many data mining applications. As a result, the true density mass in the vicinity of a data point is oriented along the manifold, rather than along all the directions in the input space. Estimating the unknown density by conventional techniques such as the Parzen windows [1] is suboptimal, as it leads to giving too much probability to irrelevant directions of space (i.e. perpendicular to the local manifold orientation) and too little along the manifold. In [2] manifold Parzen windows are introduced to improve nonparametric density estimation in this situation. In this paper, a related approach for mixture models is proposed.

In practice, finite mixtures [3], and in particular Gaussian mixtures, can also be used for nonparametric-like density estimation [4]. That is, provided the number of components can be varied arbitrarily and provided the numerical difficulties encountered when learning the parameters by the expectation-maximization (EM) algorithm [5] can be avoided, they are suitable to estimate any unknown density. The aim of this work is to show how to incorporate the prior knowledge that the data are located on a lower dimensional manifold during the learning process by EM. This is achieved by acting on the responsibilities only. Based on the discrepancy between the Euclidian and the geodesic distance, a manifold constrained E-step is constructed resulting in better generalization capabilities.

---

\* C.A. is supported by the European Commission project IST-2000-25145.

\*\* M.V. is a Senior Research Associate of Belgian National Fund for Scientific Research.

Section 2 presents how to recover the data manifold and how to approximate the geodesic distance by the graph distance. In Section 3, the learning procedure of finite Gaussian mixtures (FGM) by EM is recalled. Section 4 introduces manifold constrained Gaussian mixtures (MFGM) and discusses the resulting E-step. Finally, in Section 5, the approach is validated experimentally and compared to Parzen windows using Gaussian isotropic kernels and ordinary FGM.

## 2 Constructing the Data Manifold

The basic principle of nonlinear data projection techniques, such as CDA [6] and ISOMAP [7] is to find the lower dimensional data manifold (if any) embedded in the input space and unfold it. An essential building block for constructing the manifold is the geodesic distance. This metric is measured along the manifold and not through the embedding space, akin the Euclidean distance. As a result, the geodesic distance less depends on the curvature of the manifold, thus taking the intrinsic geometrical structure of the data into account.

### 2.1 Geodesic Distances

Consider two data points  $\mathbf{x}_i$  and  $\mathbf{x}_j$  on the multidimensional manifold  $\mathcal{M}$  of lower dimensionality as the embedding space. Manifold  $\mathcal{M}$  is parameterized as follows:

$$\mathbf{m} : \mathbb{R}^p \rightarrow \mathcal{M} \subset \mathbb{R}^d : \mathbf{t} \mapsto \mathbf{x} = \mathbf{m}(\mathbf{t}) \text{ ,}$$

where  $d$  is the dimension of the embedding space and  $p (\leq d)$  is the dimension of  $\mathcal{M}$ . Different paths may go from point  $\mathbf{x}_i$  to point  $\mathbf{x}_j$ . Each of them is described by a one-dimensional submanifold  $\mathcal{P}_{i,j}$  of  $\mathcal{M}$  with parametric equations:

$$\mathbf{p} : \mathbb{R} \rightarrow \mathcal{P}_{i,j} \subset \mathbb{R}^p : z \mapsto \mathbf{t} = \mathbf{p}(z) \text{ .}$$

The geodesic distance between  $\mathbf{x}_i$  and  $\mathbf{x}_j$  is then defined as the minimal arc length connecting both data samples:

$$l(\mathbf{x}_i, \mathbf{x}_j) = \min_{\mathbf{p}(z)} \int_{z_i}^{z_j} \|\mathbf{J}_z \mathbf{m}(\mathbf{p}(z))\| dz \text{ ,}$$

where  $\mathbf{J}_z(\cdot)$  denotes the Jacobian with respect to  $z$ . In practice, such a minimization is untractable, since it is a functional minimization.

### 2.2 Graph Distances

Even though geodesic distances cannot be computed in practice, they can easily be approximated by graph distances [8]. The problem of minimizing the arc length between two data samples lying on  $\mathcal{M}$  reduces to the problem of minimizing the length of path (i.e. broken line) between these samples, while passing through a number of other data points of  $\mathcal{M}$ . In order to follow the manifold, only the smallest jumps between successive samples are permitted. This can be

achieved by using either the  $K$ -rule, or the  $\epsilon$ -rule. The former allows jumping to the  $K$  nearest neighbors. The latter allows jumping to samples lying inside a ball of radius  $\epsilon$  centered on them. In the remaining of the paper, we only consider the  $K$ -rule as the choice for  $\epsilon$  is more difficult in practice than for  $K$ .

The data and the set of allowed jumps constitutes a weighted graph, the vertices being the data, the edges the allowed jumps and the edge labels the Euclidean distance between the corresponding vertices. In order to be a distance, the path length (i.e. the sum of successive jumps) must satisfy the properties of non-negativity, symmetry and triangular inequality. The first and the third are satisfied by construction. Symmetry is ensured when the graph is undirected. For the  $K$ -rule, this is gained by adding edges as follows: if  $\mathbf{x}_j$  belongs to the  $K$  nearest neighbors of  $\mathbf{x}_i$ , but  $\mathbf{x}_i$  is not a neighbor of  $\mathbf{x}_j$  then the corresponding edge is added. Remark also that extra edges are added to the graph in order to avoid disconnected parts. For this purpose a minimum spanning tree [9] is used.

The only remaining problem for constructing the distance matrix of the weighted undirected graph is to compute the shortest path between all data samples. This is done by repeatedly applying Dijkstra’s algorithm [10], which computes the shortest path between a source vertex and all other vertices in a weighted graph provided the labels are non-negative (which is here the case).

### 3 Finite Gaussian Mixtures

A finite Gaussian mixture (FGM) [3] is a linear combination of  $M$  Gaussian distributions:

$$\hat{p}(\mathbf{x}) = \sum_{m=1}^M \pi_m \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_m, \boldsymbol{\Lambda}_m) \quad , \tag{1}$$

The mixing proportions  $\{\pi_m\}_{m=1}^M$  are non-negative and must sum to one. The multivariate Gaussian distribution with mean  $\boldsymbol{\mu}$  and precision or inverse covariance matrix  $\boldsymbol{\Lambda}$  is defined as:

$$\mathcal{N}(\mathbf{x} | \boldsymbol{\mu}, \boldsymbol{\Lambda}) = (2\pi)^{-\frac{d}{2}} |\boldsymbol{\Lambda}|^{\frac{1}{2}} \exp \left\{ -\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Lambda} (\mathbf{x} - \boldsymbol{\mu}) \right\} \quad , \tag{2}$$

where  $\mathbf{x} \in \mathbb{R}^d$  and  $|\boldsymbol{\Lambda}|$  is the determinant of  $\boldsymbol{\Lambda}$ .

Estimating the true density  $p(\mathbf{x})$  by the approximate density  $\hat{p}(\mathbf{x})$  then consists in computing the parameters  $\{\boldsymbol{\mu}_m\}_{m=1}^M$ ,  $\{\boldsymbol{\Lambda}_m\}_{m=1}^M$  and  $\{\pi_m\}_{m=1}^M$  based on the observed data  $\{\mathbf{x}_n\}_{n=1}^N$ . By applying the EM algorithm [5] their maximum likelihood estimates can be computed in an elegant way.

Given a particular density model and assuming the data samples are i.i.d., the joint distribution of the observed data or data likelihood is:

$$\mathcal{L} = \hat{p}(\mathbf{x}_1, \dots, \mathbf{x}_N | \pi_1, \dots, \pi_M, \boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_M, \boldsymbol{\Lambda}_1, \dots, \boldsymbol{\Lambda}_M) = \prod_{n=1}^N \hat{p}(\mathbf{x}_n) \quad .$$

Unfortunately for FGM, maximizing  $\mathcal{L}$  (or equivalently its log) subject to the constraint on the mixture proportions is untractable, unless one defines a component dependent auxiliary variable associated to each data sample:

$$\rho_m(\mathbf{x}_n) = \frac{\pi_m \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_m, \boldsymbol{\Lambda}_m)}{\sum_{m=1}^M \pi_m \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_m, \boldsymbol{\Lambda}_m)} . \quad (3)$$

Keeping the auxiliary variables fixed, the Lagrangian  $\log \mathcal{L} + \lambda(\sum_{m=1}^M \pi_m - 1)$ ,  $\lambda$  being the Lagrange multiplier, can be maximized by setting its derivatives with respect to the model parameters to zero. Rearranging leads to the following estimation formulas for the component means, precisions and weights:

$$\boldsymbol{\mu}_m = \frac{\sum_{n=1}^N \rho_m(\mathbf{x}_n) \mathbf{x}_n}{\sum_{n=1}^N \rho_m(\mathbf{x}_n)} , \quad (4)$$

$$\boldsymbol{\Lambda}_m = \left\{ \frac{\sum_{n=1}^N \rho_m(\mathbf{x}_n) (\mathbf{x}_n - \boldsymbol{\mu}_m) (\mathbf{x}_n - \boldsymbol{\mu}_m)^T}{\sum_{n=1}^N \rho_m(\mathbf{x}_n)} \right\}^{-1} , \quad (5)$$

$$\pi_m = \frac{1}{N} \sum_{n=1}^N \rho_m(\mathbf{x}_n) . \quad (6)$$

Observe that (4) and (5) are nothing else than weighted averages based on the auxiliary variables  $\rho_m(\mathbf{x}_n)$ .

EM [5, 3] operates iteratively in two stages. In the E-step, the auxiliary variables (3) are computed, while the current model parameters are kept fixed. Subsequently, during the M-step the model parameters are updated according to (4-6) using the auxiliary variables computed in the E-step. At each iteration step a monotonic increase of the likelihood function is guaranteed [11].

**Interpretation of the E-Step.** Each mixture proportion  $\pi_m$  is the prior probability of having the  $m^{\text{th}}$  component of the mixture. Recalling Bayes' rule, it can easily be seen from expression (3) that each auxiliary variable  $\rho_m(\mathbf{x}_n)$  is the posterior probability that data sample  $\mathbf{x}_n$  was generated by the mixture component  $m$ , provided density model (1). In other words, it corresponds to the probability of having component  $m$  if data sample  $\mathbf{x}_n$  is observed:

$$\hat{P}(m | \mathbf{x}_n) = \frac{P(m) \hat{p}(\mathbf{x}_n | m)}{\hat{p}(\mathbf{x}_n)} = \frac{\pi_m \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_m, \boldsymbol{\Lambda}_m)}{\sum_{m=1}^M \pi_m \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_m, \boldsymbol{\Lambda}_m)} = \rho_m(\mathbf{x}_n) .$$

The auxiliary variables are therefore often called responsibilities.

**Latent Variable Viewpoint of the E-Step.** More formally, finite mixture models can be viewed as latent variable models. The component label associated to each data sample is unobserved, that is we do not know by which component a data sample was generated. Consider the set of binary latent vectors  $\{\mathbf{z}_n\}_{n=1}^N$ , with latent variables  $z_{nm} \in \{0, 1\}$  indicating which component has

generated  $\mathbf{x}_n$  ( $z_{nm} = 1$  if  $\mathbf{x}_n$  was generated by component  $m$  and 0 otherwise, and  $\sum_{m=1}^M z_{nm} = 1$ ). The prior distribution of the latent variables and the conditional distribution of observed data are then respectively:

$$\hat{P}(\mathbf{z}_n) = \prod_{m=1}^M \pi_m^{z_{nm}} , \quad \hat{p}(\mathbf{x}_n | \mathbf{z}_n) = \prod_{m=1}^M \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_m, \boldsymbol{\Lambda}_m)^{z_{nm}} .$$

Marginalizing over the latent variables results indeed in (1). Given this latent variable model, it can be shown that EM maximizes iteratively the expected complete data log-likelihood with respect to the posterior distribution of the latent variables (subject to the constraint on the mixture proportions):

$$E_{z|\mathbf{x}} [\log \mathcal{L}] = \sum_{n=1}^N \sum_{m=1}^M \underbrace{E_{z|\mathbf{x}} [z_{nm}]}_{\rho_m(\mathbf{x}_n)} \{ \log \pi_m + \log \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_m, \boldsymbol{\Lambda}_m) \} ,$$

where  $E_{z|\mathbf{x}} [\cdot]$  is the expectation with respect to  $\hat{P}(z_{nm} | \mathbf{x}_n)$ . In other words, EM uses the expected value of the latent variables as indicator of the component that generated the data samples. This expected value is equal to the responsibility.

## 4 Manifold Finite Gaussian Mixtures

Assume the data is lying on a manifold of lower dimension than the dimension of the input space. It would be appealing to take this additional information into account when learning the model parameters. Below, we explain how to achieve this by adjusting the responsibilities according to some prior belief on the discrepancy between the Euclidian and the geodesic distance.

### 4.1 Manifold Constrained E-Step

Let us respectively denote the Euclidian and graph distance between sample  $\mathbf{x}_n$  and component mean  $\boldsymbol{\mu}_m$  by  $\delta^e(\mathbf{x}_n, \boldsymbol{\mu}_m)$  and  $\delta^g(\mathbf{x}_n, \boldsymbol{\mu}_m)$ . The graph distance  $\delta^g(\mathbf{x}_n, \boldsymbol{\mu}_m)$  approximates the corresponding geodesic distance  $l(\mathbf{x}_n, \boldsymbol{\mu}_m)$ .

Consider the exponential distribution with location parameter  $\gamma$  and scale parameter  $\beta$ :

$$\mathcal{E}(y | \gamma, \beta) = \frac{1}{\beta} \exp \left\{ -\frac{y - \gamma}{\beta} \right\} . \tag{7}$$

Setting  $\gamma$  to  $\delta^e(\mathbf{x}_n, \boldsymbol{\mu}_m)^2$  and  $y$  to  $\delta^g(\mathbf{x}_n, \boldsymbol{\mu}_m)^2$  provides an appropriate measure of the mismatch between both distances, since  $\delta^e(\mathbf{x}_n, \boldsymbol{\mu}_m) \leq \delta^g(\mathbf{x}_n, \boldsymbol{\mu}_m)$ . The adjusted responsibilities can be defined as follows:

$$\rho_m'(\mathbf{x}_n) = \frac{P(m) \hat{p}'(\mathbf{x}_n | m)}{\hat{p}'(\mathbf{x}_n)} = \frac{\pi_m \mathcal{N} \mathcal{E}(\mathbf{x}_n | \boldsymbol{\mu}_m, \boldsymbol{\Lambda}_m)}{\sum_{m=1}^M \pi_m \mathcal{N} \mathcal{E}(\mathbf{x}_n | \boldsymbol{\mu}_m, \boldsymbol{\Lambda}_m)} , \tag{8}$$

where  $\mathcal{N} \mathcal{E}(\mathbf{x}_n | \boldsymbol{\mu}_m, \boldsymbol{\Lambda}_m)$  is a Gaussian-Exponential distribution of the following particular form:

$$\mathcal{N} \mathcal{E}(\mathbf{x}_n | \boldsymbol{\mu}_m, \boldsymbol{\Lambda}_m) = \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_m, \boldsymbol{\Lambda}_m) \mathcal{E}(\delta^g(\mathbf{x}_n, \boldsymbol{\mu}_m)^2 | \delta^e(\mathbf{x}_n, \boldsymbol{\mu}_m)^2, 1) . \tag{9}$$

Choosing  $\beta$  equal to 1 leaves the responsibility unchanged if both distances are identical. However, when the discrepancy between the distances increases the conditional distribution  $\hat{p}'(\mathbf{x}_n|m)$  decreases. This means that it is less likely that data sample  $\mathbf{x}_n$  was generated by component  $m$  because the corresponding geodesic distance is large compared to the Euclidian distance. This results in a weaker responsibility. As a consequence, data samples lying far away from the component means on the manifold will contribute less to the update of the corresponding component means and precisions during the M-step.

Remark also that adapting the responsibilities in this way is consistent with the latent variable viewpoint. It can be shown that in this case, manifold constrained EM maximizes iteratively the expected complete data log-likelihood with respect to the resulting adjusted posterior  $\hat{P}'(z_{nm}|\mathbf{x}_n)$  instead of  $\hat{P}(z_{nm}|\mathbf{x}_n)$ :

$$E_{z|\mathbf{x}} [\log \mathcal{L}] = \sum_{n=1}^N \sum_{m=1}^M \underbrace{E_{z|\mathbf{x}} [z_{nm}]}_{\rho_{m'}(\mathbf{x}_n)} \{ \log \pi_m + \log \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_m, \boldsymbol{\Lambda}_m) \} .$$

In this equation  $E_{z|\mathbf{x}} [\cdot]$  is the expectation with respect to the posterior  $\hat{P}'(z_{nm}|\mathbf{x}_n)$ , which is adjusted according to the mismatch between both distances.

### 4.2 Learning Manifold Gaussian Mixtures

The learning procedure for manifold constrained finite Gaussian mixtures (MFGM) can be summarized as follows:

1. Construct the learning manifold by the  $K$ -rule and compute the associated distance matrix  $\delta^g(\mathbf{x}_i, \mathbf{x}_j)$  by Dijkstra's shortest path algorithm.
2. Repeat until convergence:

**Update the distance matrix of the component means.** Find for each  $\boldsymbol{\mu}_m$  the  $K$  nearest training samples  $\{\mathbf{x}_k\}_{k=1}^K$  and compute its graph distances to all training data by  $\delta^g(\mathbf{x}_n, \boldsymbol{\mu}_m) = \min_k \{ \delta^g(\mathbf{x}_n, \mathbf{x}_k) + \delta^e(\mathbf{x}_k, \boldsymbol{\mu}_m) \}$ .

**E-step.** Compute the manifold constrained responsibilities by (8).

**M-step.** Update the model parameters by (4-6).

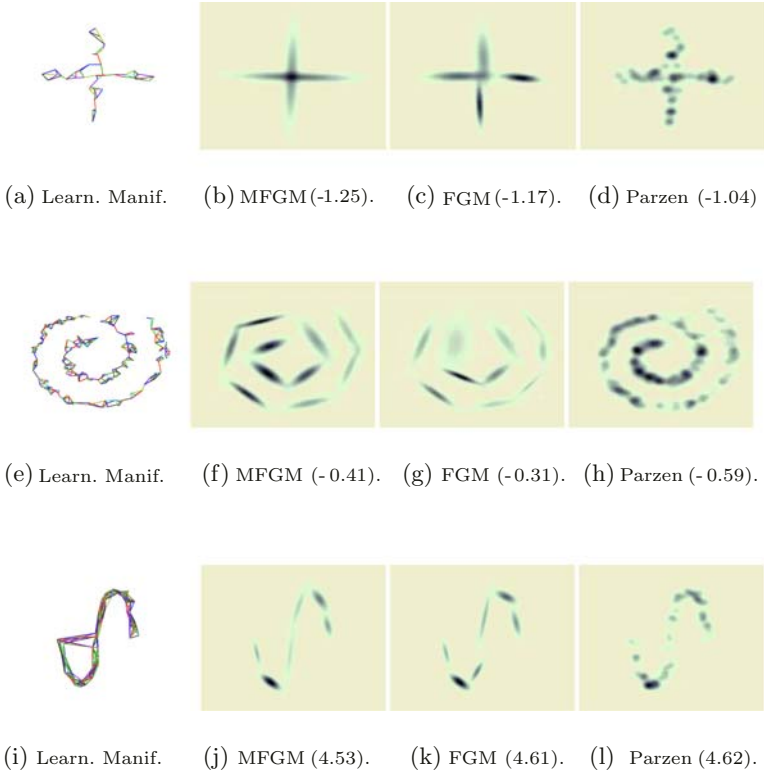
End.

Remark that the increase of the computational cost at each iteration step is limited with respect to conventional FGM. Indeed, updating the distance matrix of the component means does not require to recompute the data manifold, nor to re-apply Dijkstra's algorithm. The additional computational effort is due to the construction of the learning manifold and the computation of its distance matrix; both are performed only once (in step 1).

## 5 Experimental Results

In this section, the quality of MFGM density estimators are assessed on three 2D artificial data sets. MFGM is compared to ordinary FGM and Parzen windows





**Fig. 1.** Density estimators of a Cross, a Spiral and a S-shape. Each column shows successively the learning manifolds, the estimates of MFGM, ordinary FGM and Parzen windows. For each model, the ANLL of the test set is between parentheses

using Gaussian kernels [1]. The performance measure that we use is the average negative log-likelihood of the test set  $\{\mathbf{x}_q\}_{q=1}^{N_t}$ :  $ANLL = -\frac{1}{N_t} \sum_{q=1}^{N_t} \log \hat{p}(\mathbf{x}_q)$ .

The first distribution is a *Cross*. The data samples are generated from a uniform  $\mathcal{U}(-0.5, +0.5)$  in horizontal or vertical direction with probability  $\frac{1}{2}$ . Gaussian noise with zero mean and standard deviation  $\sigma_n = 0.03$  is added in the transversal direction. The training set and the validation set contain both 100 samples, and the test set 500 samples. For comparison purposes  $M$  is fixed a priori to 4 for both mixture models. The density estimators using the optimal kernel width for Parzen windows ( $\sigma_{opt} = 0.03$ ) and the optimal number of neighbors for MFGM ( $K_{opt} = 3$ ), as well as the ANLL are shown in Figure 1.

The second data set is located along a noisy *Spiral*. A training set of 300 points, a validation set of 300 points and a test of 1000 points were generated from the following distribution:  $\mathbf{x} = [0.04t \sin(t) + e_1, -0.04t \cos(t) + e_2]$ , where  $t \sim \mathcal{U}(3, 15)$  and  $e_1, e_2 \sim \mathcal{N}(0, 0.025^{-2})$ . The number of components in the

mixtures is fixed to 10, the optimal kernel width for Parzen is 0.025 and the optimal number of neighbors for constructing the learning manifold is 4. The results are shown in Figure 1.

The third distribution has a *S-shape*. A training set, validation set and test set of respectively 100, 100 and 1000 points are generated from one of the following distributions with probability  $\frac{1}{2}$ :  $\mathbf{x} = [3 \cos(t) - 3 + e_1, -10 \sin(t) + e_2]$  or  $\mathbf{x} = [3 \cos(t) + 3 + e_1, 10 \sin(t) + e_2]$ , with  $t \sim \mathcal{U}(0, \pi)$  and  $e_1, e_2 \sim \mathcal{N}(0, 0.5^{-2})$ . The results for  $M = 6$ ,  $\sigma_{\text{opt}} = 0.5$  and  $K_{\text{opt}} = 10$  are shown in Figure 1.

**Discussion.** Visually MFGM gives the best results for the three experiments, the discretization step being chosen sufficiently small to avoid visual artifacts. On the one hand, MFGM provides smoother estimates than Parzen windows. On the other hand, the geometric arrangement of the data is better respected with MFGM than with conventional FGM. In the case of the spiral, FGM completely fails to provide a good estimate as one component mixes two branches. Numerically, MFGM generalizes better than FGM in the three examples, as we observe a lower ANLL on the test set (see Fig. 1). Note also that the MFGM is not sensitive to few unhappy edges in the learning manifold, e.g. the S-shape.

## 6 Conclusion

In this paper, manifold finite Gaussian mixtures (MFGM) were introduced. It was shown that in situations where the data are located along a lower dimensional manifold, MFGM outperforms ordinary FGM. As with FGM, the parameters of MFGM are learnt by EM, except that the E-step is further constrained according to the mismatch between the Euclidean and the geodesic distance. As a result, training samples lying close to a component mean in Euclidean space, but far away on the manifold, will less contribute to the computation of the corresponding mean and covariance matrix in the M-step. In the near future, we plan to extend the approach to other mixtures models, e.g. Student-*t* mixtures. We also plan to study the effect of fine tuning hyperparameter  $\beta$ , which regulates how the mismatch between both distances penalizes the responsibilities.

## References

1. E. Parzen. On estimation of a probability density function and mode. *Ann. Math. Stat.*, 33:1065–1076., 1962.
2. P. Vincent and Y. Bengio. Manifold Parzen windows. In S. Thrun S. Becker and K. Obermayer, editors, *NIPS 15*, pages 825–832. MIT Press, 2003.
3. G. J. McLachlan and D. Peel. *Finite Mixture Models*. Wiley, New York, NY., 2000.
4. C. Archambeau and M. Verleysen. From semiparametric to nonparametric density estimation and the regularized Mahalanobis distance. *Submitted.*, 2005.
5. A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm (with discussion). *J. Roy. Stat. Soc., B*, 39:1–38., 1977.

6. J. A. Lee, A. Lendasse, and M. Verleysen. Nonlinear projection with curvilinear distances: Isomap versus Curvilinear Distance Analysis. *Neurocomputing*, 57:49–76., 2003.
7. J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323., 2000.
8. M. Bernstein, V. de Silva, J. Langford, and J. Tenenbaum. Graph approximations to geodesics on embedded manifolds. Techn. report Stanford University, CA., 2000.
9. D. B. West. *Introduction to Graph Theory*. Prentice Hall, Upper Saddle River, NJ., 1996.
10. E. W. Dijkstra. A note on two problems in connection with graphs. *Num. Math.*, 1:269–271., 1959.
11. L. Xu and M. I. Jordan. On convergence properties of the EM algorithm for Gaussian mixtures. *Neural Comput.*, 8:129–151., 1996.

# A Comparison of Gaussian Based ANNs for the Classification of Multidimensional Hyperspectral Signals

A. Prieto, F. Bellas, R.J. Duro, and F. Lopez-Peña

Grupo de Sistemas Autónomos, Universidade da Coruña, Spain  
{abprieto, fran, richard, flop}@udc.es

**Abstract.** This paper is concerned with the comparison of three types of Gaussian based Artificial Neural Networks in the very high dimensionality classification problems found in hyperspectral signal processing. In particular, they have been compared for the spectral unmixing problem given the fact that the requirements for this type of classification are very different from other realms in two aspects: there are usually very few training samples leading to networks that are very easily overtrained, and these samples are not usually representative in terms of sampling the whole input-output space. The networks selected for comparison go from the classical Radial Basis Function (RBF) network to the more complex Gaussian Synapse Based Network (GSBN) considering an intermediate type, the Radial Basis Function with Multiple Deviation (RBFMD). The comparisons were carried out when processing a benchmark set of synthetic hyperspectral images containing mixtures of spectra from materials found in the US Geological Service database.

## 1 Introduction

As remote sensing of the earth becomes a more important element in the toolbox of scientists, the mechanisms for processing the ever larger data streams coming from the instruments used for this purpose increase their relevance. High resolution hyperspectrometers are among the instruments that will be used in an ever increasing number [1][2][3]. In hyperspectral remote sensing images, each spatial resolution element data is acquired with high spectral resolution over the electromagnetic spectrum ranging the 400-2500 nm (visible to near infrared). It is commonplace to use 50 to 250 spectral bands of bandwidths in the 5 to 20 nm range. The large amount of information hyperspectral imaging provides permits a detailed description of the spectral signature thus greatly improving the ability to detect and identify individual materials or classes with respect to other remote sensing techniques.

Any analysis or classification method for hyperspectral image processing is aimed at the identification of what pixels contain different spectrally distinct materials (endmembers) and in what proportions. A number of approaches based on statistical theory or using filtering or correlations have been applied to the analysis of these data sets by different authors with the objective of improving the classification results [4]. Some of these methods are compared using classification performance in [5].

From this point of view, artificial neural networks (ANNs) appear as a very suitable and effective alternative to deal with spectral image analysis difficulties [6] [7]. In the

case of ANNs trained using supervised training algorithms, the main drawback to performing the analysis and classification of hyperspectral remote sensing data is the difficulty in obtaining labelled sample data that can be trusted to the point of using it for quantitative evaluation. The scarcity of ground truth data has been recognized and specific training strategies have been devised to cope with this handicap [8], [9]. In fact, in an ideal world, we would be able to obtain a training procedure that produces good classifiers from very small training sets. This problem becomes even more pronounced when pixels correspond to combinations of materials, that is, the “spectral unmixing” problem [6]. Ideally, hyperspectral images may allow the identification of single pixel materials. However as these pixels are frequently combinations of materials, it is often a plus to be able to decompose each pixel spectrum into its constituent material spectra. The cause of the mixture of several material spectra into a hyperspectral pixel spectrum may be that different land covers are included in the area whose radiance measurement results in an image pixel.

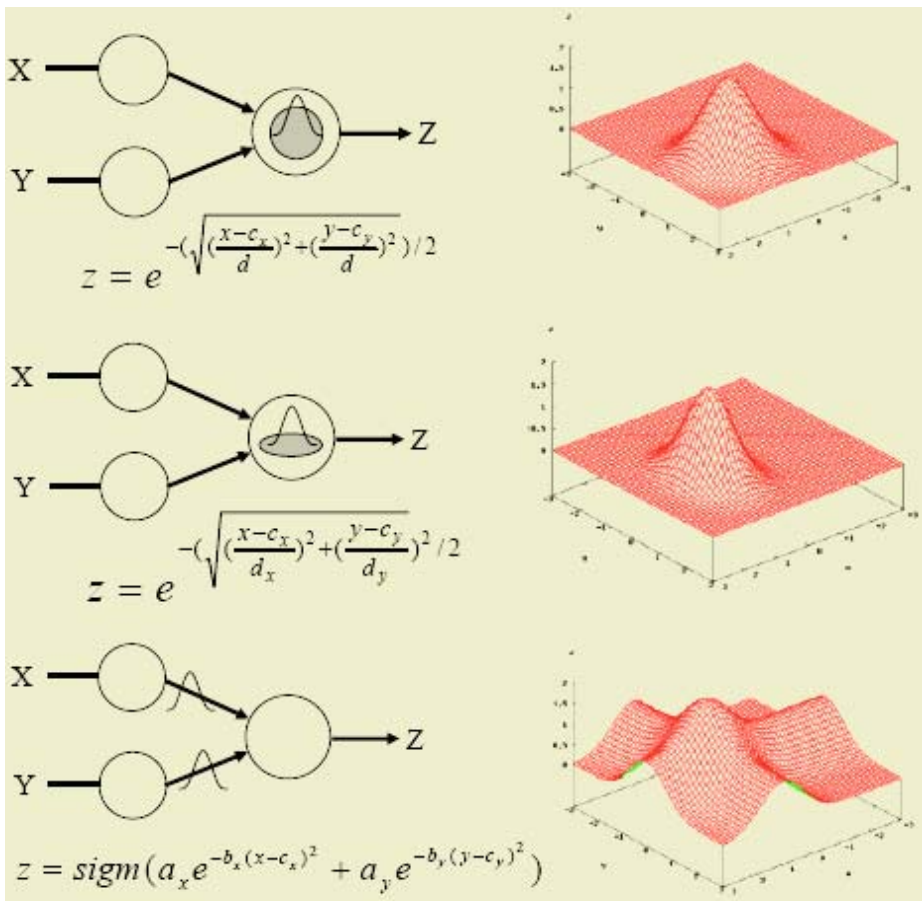
In this work we assume a linear mixture model, in which several basic materials (endmembers) are combined according to some abundance coefficients at each image pixel. Taking its spatial distribution, the abundance coefficients may be visualized as abundance images, which provide a description of the spatial distribution of the material. The computation of the abundance coefficients given a pixel spectrum and a set of endmembers is what is termed the unmixing procedure. If the endmembers are given, the unmixing procedure is equivalent to the parallel detection of the spectral features represented by the endmembers.

One basic approach is classical, if you concentrate only on what is relevant the classification becomes much more robust and efficient. This is the approach followed in the work leading to this paper. An Artificial Neural Network architecture and training algorithm that implement an automatic procedure to concentrate on what is relevant and ignore what is not straight from the training set is required in order to effectively perform the task. To do this, many authors have resorted to Gaussian based ANNs as a way to implicitly establish this filtering ability in the network. This is the case of the work of Dundar and Landgrebe [10] with RBFs, where the authors claim that Gaussian RBFs have proven to be in their case the most effective network for hyperspectral image processing, or that of Crespo et al. [11] with GSBNs where the Gaussian processing has been transferred to the synapses, thus providing more degrees of freedom.

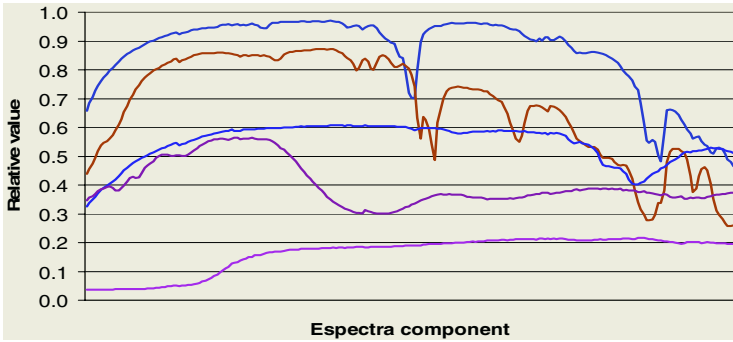
In this paper the objective is to evaluate the performance of three Gaussian based types of ANNs when dealing with multidimensional signals and very few training points are available. These three types of networks range from the classical Radial Basis Function Network as proposed above, to a modification of RBFs where they have been endowed with trainable deviations for each dimension, to the more versatile Gaussian Synapse Based networks. In the sections that follow we will describe this networks and through a similar backpropagation based training algorithm we will compare their performance when unmixing a benchmark set of images based on Graña et al’s repository.

## 2 Description of the ANNs

Gaussian based ANNs have been widely applied in image processing due to their capabilities in noise filtering. As commented in the previous section, we are going to compare the results provided by three different types of these networks. In the radial basis function (RBF) neural networks the input layer directly transmits the inputs to the neurons of the hidden layer. In these neurons the Gaussian function is applied over the inputs by using one parameter per synapse (center) and one parameter per hidden neuron (deviation). The output provided by the neurons in the hidden layer passes to the output neurons through a linear combination of weights as in a perceptron. The radial basis function with multiple deviation (RBFMD) neural networks are structurally similar to RBF networks but having one deviation parameter per synapse

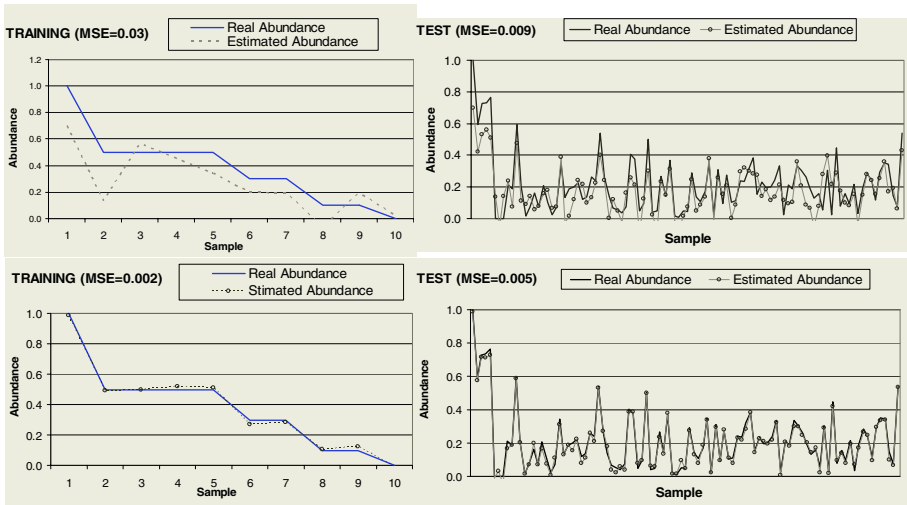


**Fig. 1.** Top graph represents an RBF neural network, middle graph and RBFMD neural network and bottom graph a GSNB



**Fig. 2.** Representation of the spectra corresponding to the 5 endmembers used to generate the training and test data sets

instead of per neuron in the hidden layer. Finally, the Gaussian synapse based networks (GSBN) have a multilayer perceptron structure, but replacing the simple weights used in the synapses by Gaussian functions in order to filter the inputs. The main difference between these three types of networks is the number of parameters to be trained, being the RBF the one requiring fewer parameters and the GSBN the one requiring most. As it is shown in the next section, this number of parameters determines the complexity of the learning process and the generalization capability.



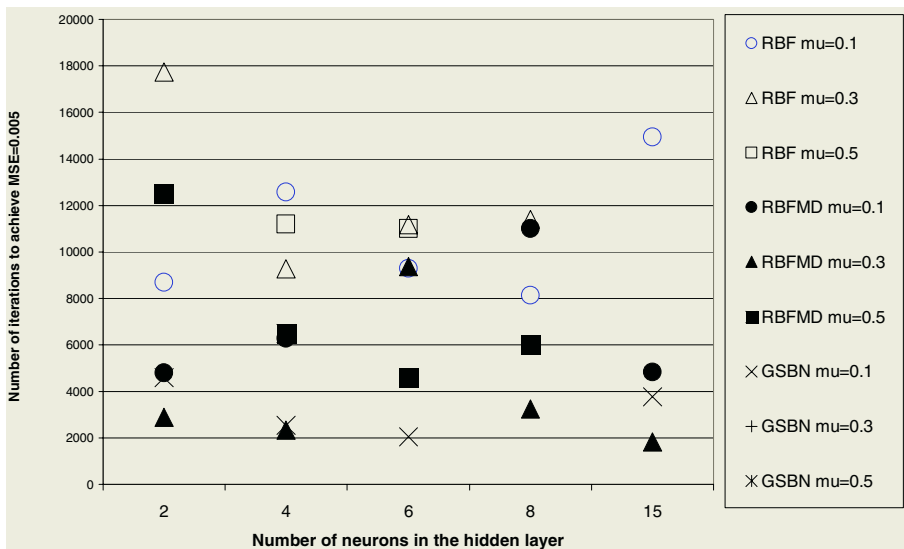
**Fig. 3.** Modeling over the training set (left) and the test set (right) provided by one of the networks for two significant error values: 0.009 in top graphs and 0.005 in bottom graph

The relevant differences between these types of ANNs arise from their basic structural units as shown in Fig. 1. These units have two input neurons ( $x$  and  $y$ ) and one output neuron ( $z$ ). The top graph of Fig. 1 corresponds to a RBF neural network,

in which a single parameter per synapse (the centers  $C_x$  and  $C_y$ ) should be trained. The middle graph corresponds to a RBFMD neural network where two parameters per synapse must be trained (the centers  $C_x$  and  $C_y$  and the deviations  $d_x$  and  $d_y$ ). Finally, at the bottom is a GSBN neural network where 3 parameters per synapse should be trained ( $a_x, b_x, c_x, a_y, b_y$  and  $c_y$ ). The right part of Fig. 1 provides a representation of the decision boundaries implementable by each of these three ANNs when the input variables range between -3 and 3. This corresponds to the possible outputs provided by a certain neuron of the hidden layer to the next layer. As it can be seen, RBF networks (top graph) provide symmetric decision boundaries in both axes while the RBFMD ones (middle graph) are able to filter independently on each axis presenting a more flexible decision boundary. This capability is very suitable for our purposes because the network has higher degree of freedom to filter the information. In the case of the GSBN network (bottom graph) we can see a more complex decision boundary because the filtering appears in both axes independently again and, in addition, the shared zone could be discriminated through the value of the sigmoids in the neurons.

The main reason for studying networks that provide more complex and flexible decision boundaries is the high dimensionality of the search space and the complexity of the subspace to discriminate in hyperspectral unmixing problems. The application of these networks increases the number of parameters that must be trained and, consequently, the computational cost but it should be compensated by the decrease of the minimum necessary network size and, what is more important, the speed of the network training stage as well as the need to use fewer training samples.

In the next section we will show the result of the comparison between the three types of networks presented applied to a hyperspectral image.



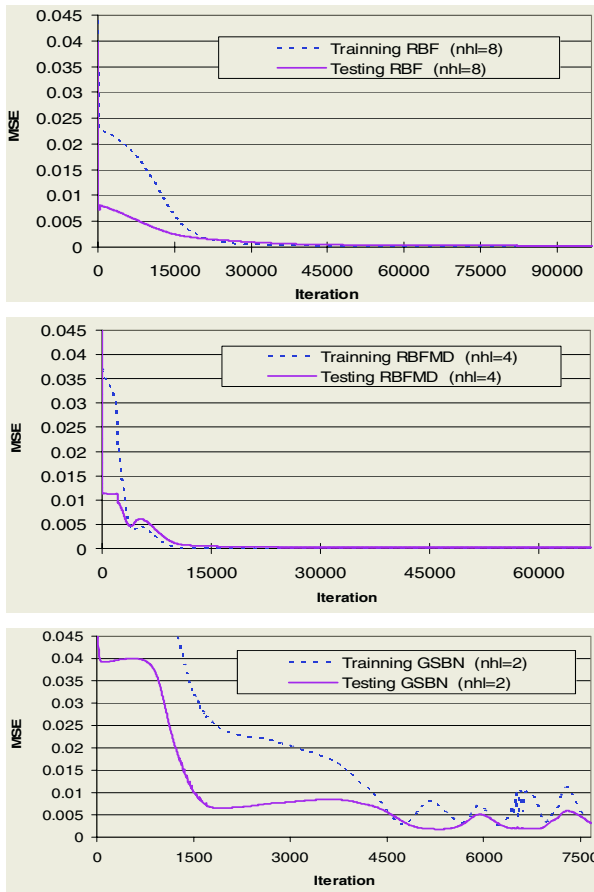
**Fig. 4.** Number of neurons in the hidden layer versus the number of iterations to achieve the desired for the 3 types of networks (RBF represented by hollows, RBFMD by solids and GSBN by marks) with 3 different  $\mu$  values for each one



### 3 Comparison of Networks

To compare the networks we have used a set of 5 synthetic endmembers developed by Graña et al. [14] which is shown in Fig. 2. A training set containing 10 spectra resulting from the linear combination of these 5 endmembers and a test set of 1000 spectra have been created. The training algorithms applied are variations of the classical backpropagation algorithm modified for the radial basis functions [13] and for the Gaussian synapses [12]. This training process is applied over the training set for each type of network and every iteration the network is tested over the test set.

First of all, in Fig. 3 we have represented the modeling over the training and the test sets provided by one of the networks for two significant error values. From the curves we can establish that the learning has been successful when the MSE is lower than 0.005 according to the modeling over the test set.



**Fig. 5.** MSE evolution with respect to the number of iterations in the training process and in the test process using a RBF network with 8 neurons (top graph), a RBFMD with 4 neurons (middle graph) and a GSBN with 2 neurons (bottom graph)

To measure the quality of the network obtained we will focus on two main parameters: the number of iterations required to achieve a given error value and the stability of error evolution (to detect overtraining problems). In the first case, in order to compare the three types of networks we have used two parameters: the learning coefficient  $\mu$  and the number of neurons in the hidden layer. In Fig. 4 we show number of neurons in the hidden layer versus the number of iterations to achieve the desired error for the 3 types of networks (RBF represented by hollow symbols, RBFMD by solid symbols and GSBN by line symbols) and with 3 different  $\mu$  values for each one. The data shown in the figure provide a good indicator of several facts. On one hand, RBF based networks are a lot slower than other types of networks in every case, even when taking into account the same number of parameters. Take into account, that an RBF with 6 neurons has the same number of trainable parameters than a RBFMD with 3 neurons and a GSBN with 2 neurons. The RBFMD is clearly more efficient in terms of reaching the solution in less iterations, but the one that performs the best is the GSBN. This is so for just about any size of the network. The figure shows that the influence of the  $\mu$  parameter is only significant in the case of GSBN networks where just low  $\mu$  values provide valid errors, if training is carried out with high  $\mu$  the networks take very long to achieve a result and often they will end up overtraining..

In terms of overtraining, taking into account the different number of parameters that characterize each type of network, we must select networks with different number of neurons in the hidden layer in order to perform an equivalent comparison. Thus, in Fig. 5 we have represented the MSE evolution against the number of iterations in the training process and in the test process using a RBF network with 8 neurons (top graph), a RBFMD with 4 neurons (middle graph) and a GSBN with 2 neurons (bottom graph). In this test, we have provided a little advantage for the RBF in terms of number of parameters, but we wanted to make sure it did not overtrain. As we can see, in the case of the RBF network, the evolution of the error in the test process is highly stable because the network does not overtrain. On the other hand, GSBN and, mainly, the RBFMD networks are more unstable they are very sensitive to overtraining and having achieved a given error for the test set as training progresses this level of error may increase and even create large oscillations in the case of RBFMD with large  $\mu$ ..

This way, from these results the main conclusion we can extract is that although the three types of networks achieve the desired error value ( $MSE=0.005$ ) in almost all cases, the RBF networks are the one that need more iterations for training for the same number of trainable parameters. The GSBN networks are the ones that need less iterations. Finally, RBFMD are in an intermediate point in both features but are quite sensitive to overtraining, probably because of the deviation parameter in the denominator of the exponential.

## 4 Conclusions

In this paper we have compared three types of Gaussian based artificial neural networks in terms of their performance when addressing the spectral unmixing problem in a set of benchmark synthetic hyperspectral images. From this comparison it is clear that for equivalent number of parameters in the networks, the Gaussian synapse based networks converge to very good classification results in less iterations

than the other two, whereas the radial basis functions do so in a much smoother fashion. In terms of the overtraining behavior, it is clear to see that the RBF networks are much harder to overtrain and thus perform much better on the test sets independently of the length of the training period (which in many cases is very hard to set beforehand), obviously, at the cost of a much slower training process.

## Acknowledgements

This work was funded by the MCYT of Spain through project VEM2003-20088-C04-01 and Xunta de Galicia through project PGIDIT03DPI099E.

## References

1. Fingas, M.F. and Brown, C.E.: Review of Oil Spill Remote Sensing, in Proceedings of the Fifth International Conference on Remote Sensing for Marine and Coastal Environments, Environmental Research Institute of Michigan, Ann Arbor, Michigan, pp. 1211-218. 2000
2. Regional Marine Pollution Emergency Response Centre for the Mediterranean Sea (REMPEC): MEDIPOL 2003 - European workshop on satellite imagery and illicit oil spills in Europe and in the Mediterranean. Final Recommendations, 2003
3. Bissett, W.P., Arnone, R., Davis, C.O., Dickey, T., Dye, D.R., Kohler, D.D.R., Gould, R.: From meters to kilometers - a look at ocean color scales of variability, spatial coherence, and the need for fine scale remote sensing in coastal ocean optics. *Oceanography*, 2003
4. Campbell, J.: Introduction to Remote Sensing. The Guilford Press, 1996
5. Hsu, P.H. and Tseng, C.E.: Feature Extraction for Hyperspectral Image. Proceedings of the 20th Asian Conference on Remote Sensing, Vol. 1, 405-410, 1999
6. Merényi, E., Minor, T. B., Taranik, J. V., and Farrand, W.H.: Quantitative Comparison of Neural Network and Conventional Classifiers for Hyperspectral Imagery, In Summaries of the Sixth Annual JPL Airborne Earth Science Workshop, Pasadena, CA, March 4-8, 1996, Vol. 1: AVIRIS Workshop, Ed. R.O.Green, 1996.
7. Ghosh, J.: Adaptive and neural methods for image segmentation. In Al Bovik, editor, Handbook of Image and Video Processing, chapter 4.10, pages 401-414. Academic Press, 2000.
8. Tadjudin, S. and Landgrebe, D.: Covariance Estimation with Limited Training Samples, *IEEE Trans. Geos. Rem. Sensing*, 37(4), 2113- 2118, 1999
9. Tadjudin, S. and Landgrebe, D.: Parameter estimation for mixture model, *IEEE Trans. Geos. Rem. Sensing*, 38(1): 439, 2000
10. M. Murat Dundar and David Landgrebe: Toward an Optimal Supervised Classifier for the Analysis of Hyperspectral Data *IEEE Transactions on Geoscience and Remote Sensing*, Volume 42, No. 1, pp 271-277, January, 2004.
11. J. L. Crespo, R.J. Duro, F. López Peña: Unmixing Low Ratio Endmembers through Gaussian Synapse ANNs in Hyperspectral Images Proceedings CIMSA 2004 IEEE International Symposium on Computational Intelligence for Measurement Systems and Applications. V1, pp: 150 -154, 2003
12. Duro, R.J., Crespo, J.L., and Santos, J.: Training Higher Order Gaussian Synapses. LNCS, Vol. 1606 Springer-Verlag, Berlín, 537-545, 1999
13. Karayiannis, N.B.: "Reformulated radial basis neural networks trained by gradient descent" *IEEE Transactions on Neural Networks*, vol. 10 no. 3, 657 -671, 1999
14. Graña, M., Raducanu, B., Sussner P., and Ritter. G.: On Endmember Detection in Hyperspectral Images with Morphological Associative Memories. Presented at IBERAMIA 2002, Sevilla, Spain, 526-535, 2002

# Voice Activity Detection Using Higher Order Statistics

J.M. Górriz, J. Ramírez, J.C. Segura, and S. Hornillo

Dept. Teoría de la Señal, Telemática y comunicaciones,  
Facultad de Ciencias , Universidad de Granada,  
Fuentenueva s/n, 18071 Granada, Spain  
`gorriz@ugr.es`

**Abstract.** A robust and effective voice activity detection (VAD) algorithm is proposed for improving speech recognition performance in noisy environments. The approach is based on filtering the input channel to avoid high energy noisy components and then the determination of the speech/non-speech bispectra by means of third order autocumulants. This algorithm differs from many others in the way the decision rule is formulated (detection tests) and the domain used in this approach. Clear improvements in speech/non-speech discrimination accuracy demonstrate the effectiveness of the proposed VAD. It is shown that application of statistical detection test leads to a better separation of the speech and noise distributions, thus allowing a more effective discrimination and a tradeoff between complexity and performance. The algorithm also incorporates a previous noise reduction block improving the accuracy in detecting speech and non-speech.

## 1 Introduction

Nowadays speech/non-speech detection is a complex problem in speech processing and affects numerous applications including robust speech recognition [1], discontinuous transmission [2, 3], real-time speech transmission on the Internet [4] or combined noise reduction and echo cancellation schemes in the context of telephony [5]. The speech/non-speech classification task is not as trivial as it appears, and most of the VAD algorithms fail when the level of background noise increases. During the last decade, numerous researchers have developed different strategies for detecting speech on a noisy signal [6] and have evaluated the influence of the VAD effectiveness on the performance of speech processing systems [7]. Most of them have focussed on the development of robust algorithms with special attention on the derivation and study of noise robust features and decision rules [8, 9, 10]. The different approaches include those based on energy thresholds [8], pitch detection [11], spectrum analysis [10], zero-crossing rate [3], periodicity measure [12], higher order statistics in the LPC residual domain [13] or combinations of different features [3, 2]. This paper explores a new alternative towards improving speech detection robustness in adverse environments and the performance of speech recognition systems. The proposed VAD proposes a noise

reduction block that precedes the VAD, and uses Bispectra of third order cumulants to formulate a robust decision rule. The rest of the paper is organized as follows. Section 2 reviews the theoretical background on Bispectra analysis and shows the proposed signal model, analyzing the motivations for the proposed algorithm by comparing the speech/non-speech distributions for our decision function based on bispectra and when noise reduction is optionally applied. Section 3 describes the experimental framework considered for the evaluation of the proposed statistical decision algorithm. Finally, section summarizes the conclusions of this work.

## 2 Model Assumptions

Let  $\{x(t)\}$  denote the discrete time measurements at the sensor. Consider the set of stochastic variables  $y_k$ ,  $k = 0, \pm 1 \dots \pm M$  obtained from the shift of the input signal  $\{x(t)\}$ :

$$\mathbf{y}_k(t) = \mathbf{x}(t + k \cdot \tau) \tag{1}$$

where  $k \cdot \tau$  is the differential delay (or advance) between the samples. This provides a new set of  $2 \cdot m + 1$  variables by selecting  $n = 1 \dots N$  samples of the input signal which can be represented using the associated Toeplitz matrix.

Using this model the speech-non speech detection can be described by using two essential hypothesis(re-ordering indexes):

$$H_o = \begin{pmatrix} \mathbf{y}_0 = n_0 \\ \mathbf{y}_{\pm 1} = n_{\pm 1} \\ \dots \\ \mathbf{y}_{\pm M} = n_{\pm M} \end{pmatrix}; \quad H_1 = \begin{pmatrix} \mathbf{y}_0 = s_0 + n_0 \\ \mathbf{y}_{\pm 1} = s_{\pm 1} + n_{\pm 1} \\ \dots \\ \mathbf{y}_{\pm M} = s_{\pm M} + n_{\pm M} \end{pmatrix} \tag{2}$$

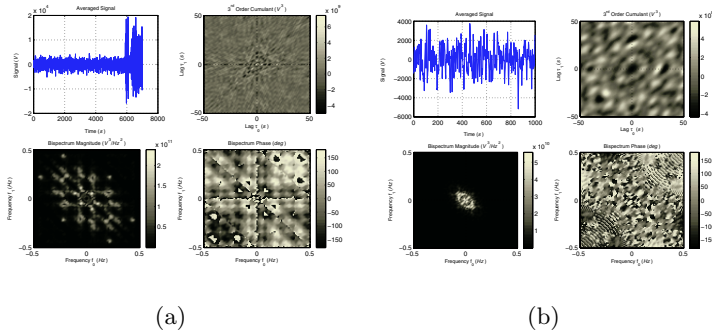
where  $s_k$ 's/ $n_k$ 's are the speech/non-speech (any kind of additive background noise i.e. gaussian) signals, related themselves with some differential parameter. All the process involved are assumed to be jointly stationary and zero-mean. Consider the third order cumulant function  $C_{y_k y_l}$  defined as:

$$C_{y_k y_l} \equiv E[y_0 y_k y_l]; \quad C_{y_k y_l}(\omega_1, \omega_2) = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} C_{y_k y_l} \cdot \exp(-j(\omega_1 k + \omega_2 l)) \tag{3}$$

and the two-dimensional discrete Fourier transform (DFT) of  $C_{y_k y_l}$ , the bispectrum function. The sequence of cumulants of the voice speech is modelled as a sum of coherent sine waves:

$$C_{y_k y_l} = \sum_{n,m=1}^K a_{nm} \cos[kn\omega_0^1 + lm\omega_0^2] \tag{4}$$

where  $a_{nm}$  is amplitude,  $K \times K$  is the number of sinusoids and  $\omega$  is the fundamental frequency in each dimension. It follows from equation 4 that  $a_{mn}$  is related to the energy of the signal  $\mathcal{E}_s = E\{s^2\}$ . The VAD proposed in the later



**Fig. 1.** Different Features allowing voice activity detection. (a) Features of Voice Speech Signal. (b) Features of non Speech Signal

reference only works with the coefficients in the sequence of cumulants and is more restrictive in the model of voice speech. Thus the Bispectra associated to this sequence is the DTF of equation 4 which consist in a set of Dirac’s deltas in each excitation frequency  $n\omega_0^1, m\omega_0^2$ . Our algorithm will detect any high frequency peak on this domain matching with voice speech frames, that is under the above assumptions and hypotheses, it follows that on  $H_0$ ,

$$C_{y_k y_l}(\omega_1, \omega_2) \equiv C_{n_k n_l}(\omega_1, \omega_2) \simeq 0 \tag{5}$$

and on  $H_1$ :

$$C_{y_k y_l}(\omega_1, \omega_2) \equiv C_{s_k s_l}(\omega_1, \omega_2) \neq 0 \tag{6}$$

Since  $s_k(t) = s(t + k \cdot \tau)$  where  $k = 0, \pm 1 \dots \pm M$ , we get

$$C_{s_k s_l}(\omega_1, \omega_2) = \mathcal{F}\{E[s(t + k \cdot \tau)s(t + l \cdot \tau)s(t)]\} \tag{7}$$

The estimation of the bispectra (equation 3) is deep discussed in [14] and many others, where conditions for consistency are given. The estimate is said to be (asymptotically) consistent if the squared deviation goes to zero, as the number of samples tends to infinity.

### 2.1 Detection Tests for Voice Activity

The decision of our algorithm implementing the VAD is based on statistical tests from references [15] (Generalized likelihood ratio tests) and [16] (Central  $\chi^2$ -distributed test statistic under  $H_0$ ). We will call the tests GLRT and  $\chi^2$  tests. The tests are based on some asymptotic distributions and computer simulations in [17] show that the  $\chi^2$  tests require larger data sets to achieve a consistent theoretical asymptotic distribution. Then we decline to use it unlike the GLRT tests.

If we reorder the components of the set of  $L$  Bispectrum estimates  $\hat{C}(n_l, m_l)$  where  $l = 1, \dots, L$ , on the fine grid around the bifrequency pair into a  $L$  vector  $\beta_{ml}$  where  $m = 1, \dots, P$  indexes the coarse grid [15] and define  $P$ -vectors

$\phi_i(\beta_{1i}, \dots, \beta_{Pi}), i = 1, \dots, L$ ; the generalized likelihood ratio test for the above discussed hypothesis testing problem:

$$H_0 : \mu = 0 \quad \text{against} \quad H_1 : \eta \equiv \mu^T \sigma^{-1} \mu > 0 \tag{8}$$

where  $\mu = 1/L \sum_{i=1}^L \phi_i$  and  $\sigma = 1/L \sum_{i=1}^L (\phi_i - \mu)(\phi_i - \mu)^T$ , leads to the activity voice speech detection if:

$$\eta > \eta_0 \tag{9}$$

where  $\eta_0$  is a constant i.e. the probability of false alarm.

### 2.2 Noise Reduction Block

Almost any VAD can be improved just placing a noise reduction block in the data channel before it. The noise reduction block for high energy noisy peaks, consists of four stages(1) Spectrum smoothing 2)Noise estimation 3)Wiener Filter (WF) design and 4)Frequency domain filtering) and was first developed in [18].

### 2.3 Some Remarks About the Algorithm

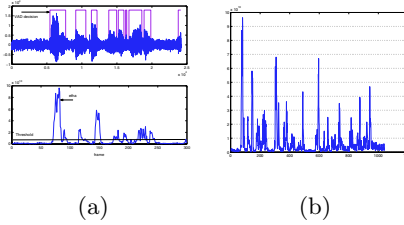
We propose a alternative decision based on an average of the components of the bispectrum (the absolute value of it). In this way we define  $\eta$  as:

$$\eta = \frac{1}{L \cdot N} \sum_{i=1}^L \sum_{j=1}^N \left| \hat{C}(i, j) \right| \tag{10}$$

where  $L, N$  defines the selected grid (high frequencies with noteworthy variability). We also include long term information (LTI) in the decision of the on-line VAD [19] which essentially improves the efficiency of the proposed method as is shown the following pseudocode:

- Initialize variables
- Determine  $\eta_0$  of noise in the first frame
- for  $i=1$  to end:
  1. Consider a new frame (i)
    - calculate  $\eta(i)$
  2. if  $H_1$  then
    - VAD(i)=1
    - apply LTI to VAD(i- $\tau$ )
  - else
    - Slow Update of noise parameters:  $\eta_0(i + 1) = \alpha\eta_0 + \beta\eta(i),$   
 $\alpha + \beta = 1 \quad \alpha \rightarrow 1$
    - apply LTI to VAD(i- $\tau$ )

Fig. 2 shows the operation of the proposed VAD on an utterance of the Spanish SpeechDat-Car (SDC) database [20]. The phonetic transcription is: [“siete”, “thinko”, “dos”, “uno”, “otSo”, “seis”]. Fig 2(b) shows the value of  $\eta$  versus time. Observe how assuming  $\eta_0$  the initial value of the magnitude  $\eta$  over the



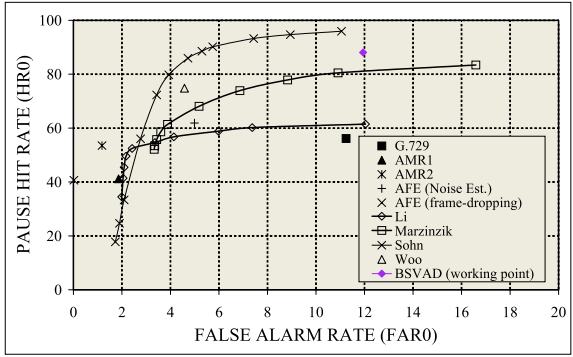
**Fig. 2.** Operation of the VAD on an utterance of Spanish SDC database. (a) Evaluation of  $\eta$  and VAD Decision. (b) Evaluation of the test hypothesis on an example utterance of the Spanish SpeechDat-Car (SDC) database [20]

first frame (noise), we can achieve a good VAD decision. It is clearly shown how the detection tests yield improved speech/non-speech discrimination of fricative sounds by giving complementary information. The VAD performs an advanced detection of beginnings and delayed detection of word endings which, in part, makes a hang-over unnecessary. In Fig 1 we display the differences between noise and voice in general and in figure we settle these differences in the evaluation of  $\eta$  on speech and non-speech frames.

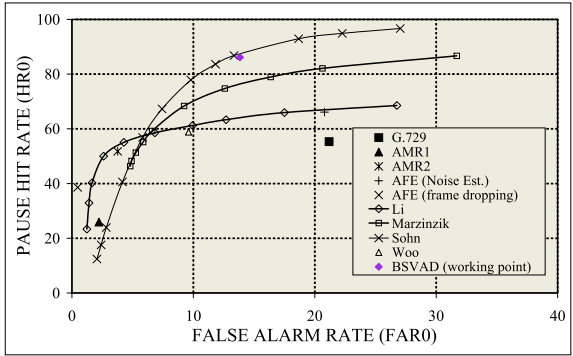
### 3 Experimental Framework

The ROC curves are frequently used to completely describe the VAD error rate. The AURORA subset of the original Spanish SpeechDat-Car (SDC) database [20] was used in this analysis. This database contains 4914 recordings using close-talking and distant microphones from more than 160 speakers. The files are categorized into three noisy conditions: quiet, low noisy and highly noisy conditions, which represent different driving conditions with average SNR values between 25dB, and 5dB. The non-speech hit rate (HR0) and the false alarm rate (FAR0= 100-HR1) were determined in each noise condition being the actual speech frames and actual speech pauses determined by hand-labelling the database on the close-talking microphone. Fig. 3 shows the ROC curves of the proposed VAD (BiSpectra based-VAD) and other frequently referred algorithms [8, 9, 10, 6] for recordings from the distant microphone in quiet, low and high noisy conditions. The working points of the G.729, AMR and AFE VADs are also included. The results show improvements in detection accuracy over standard VADs and similarities over representative set VAD algorithms [8, 9, 10, 6]. The benefits are especially important over G.729, which is used along with a speech codec for discontinuous transmission, and over the Li's algorithm, that is based on an optimum linear filter for edge detection. On average ( $\frac{HR0+HR1}{2}$ ), the proposed VAD is similar to Marzinzik's VAD that tracks the power spectral envelopes, and the Sohn's VAD, that formulates the decision rule by means of a statistical likelihood ratio test. These results clearly demonstrate that there is no optimal VAD for all the applications. Each VAD is developed and optimized for specific purposes. Hence, the evaluation has to be conducted according to the

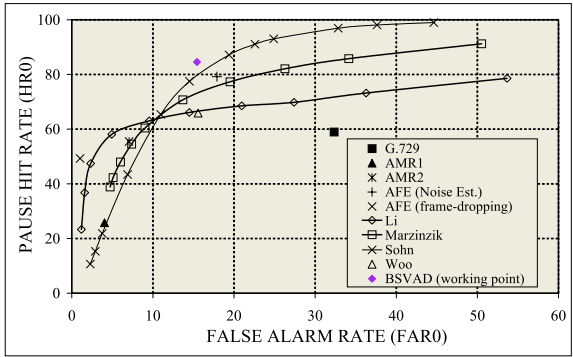




(a)



(b)



(c)

**Fig. 3.** ROC curves obtained for different subsets of the Spanish SDC database at different driving conditions: (a) Quiet (stopped car, motor running, 12 dB average SNR). (b) Low (town traffic, low speed, rough road, 9 dB average SNR). (c) High (high speed, good road, 5 dB average SNR)

**Table 1.** Average speech/non-speech hit rates for SNRs between 25dB and 5dB. Comparison of the proposed BSVAD to standard and recently reported VADs

|         | G.729  | AMR1   | AMR2      | AFE (WF) | AFE (FD)     |
|---------|--------|--------|-----------|----------|--------------|
| HR0 (%) | 55.798 | 51.565 | 57.627    | 69.07    | 33.987       |
| HR1 (%) | 88.065 | 98.257 | 97.618    | 85.437   | 99.750       |
|         | Woo    | Li     | Marzinzik | Sohn     | <b>BSVAD</b> |
| HR0 (%) | 62.17  | 57.03  | 51.21     | 66.200   | 85.150       |
| HR1 (%) | 94.53  | 88.323 | 94.273    | 88.614   | 86.260       |

specific goal of the VAD. Frequently, VADs avoid loosing speech periods leading to an extremely conservative behavior in detecting speech pauses (for instance, the AMR1 VAD). Thus, in order to correctly describe the VAD performance, both parameters have to be considered. On average the results are conclusive (see table 1).

## 4 Conclusion

This paper presented a new VAD for improving speech detection robustness in noisy environments. The approach is based on higher order Spectra Analysis employing noise reduction techniques and order statistic filters for the formulation of the decision rule. The VAD performs an advanced detection of beginnings and delayed detection of word endings which, in part, avoids having to include additional hangover schemes. As a result, it leads to clear improvements in speech/non-speech discrimination especially when the SNR drops. With this and other innovations, the proposed algorithm outperformed G.729, AMR and AFE standard VADs as well as recently reported approaches for endpoint detection. We think that it also will improve the recognition rate when it was considered as part of a complete speech recognition system.

## Acknowledgements

This work has received research funding from the EU 6<sup>th</sup> Framework Programme, under contract number IST-2002-507943 (HIWIRE, Human Input that Works in Real Environments) and SESIBONN project (TEC2004-06096-C03-00) from the Spanish government. The views expressed here are those of the authors only. The Community is not liable for any use that may be made of the information contained therein.

## References

1. L. Karray and A. Martin, "Towards improving speech detection robustness for speech recognition in adverse environments," *Speech Communication*, no. 3, pp. 261–276, 2003.

2. ETSI, "Voice activity detector (VAD) for Adaptive Multi-Rate (AMR) speech traffic channels," *ETSI EN 301 708 Recommendation*, 1999.
3. ITU, "A silence compression scheme for G.729 optimized for terminals conforming to recommendation V.70," *ITU-T Recommendation G.729-Annex B*, 1996.
4. A. Sangwan, M. C. Chiranth, H. S. Jamadagni, R. Sah, R. V. Prasad, and V. Gaurav, "VAD techniques for real-time speech transmission on the Internet," in *IEEE International Conference on High-Speed Networks and Multimedia Communications*, 2002, pp. 46–50.
5. S. Gustafsson and et al., "A psychoacoustic approach to combined acoustic echo cancellation and noise reduction," *IEEE Trans. on S.&A. Proc.*, vol. 10, no. 5, pp. 245–256, 2002.
6. J. Sohn and et al., "A statistical model-based vad," *IEEE S.Proc.L.*, vol. 16, no. 1, pp. 1–3, 1999.
7. R. L. Bouquin-Jeannes and G. Faucon, "Study of a voice activity detector and its influence on a noise reduction system," *Speech Communication*, vol. 16, pp. 245–254, 1995.
8. K. Woo and et al., "Robust vad algorithm for estimating noise spectrum," *Electronics Letters*, vol. 36, no. 2, pp. 180–181, 2000.
9. Q. Li and et al., "Robust endpoint detection and energy normalization for real-time speech and speaker recognition," *IEEE Trans. on S.&A. Proc.*, vol. 10, no. 3, pp. 146–157, 2002.
10. M. Marzinzik and et al., "Speech pause detection for noise spectrum estimation by tracking power envelope dynamics," *IEEE Trans. on S.&A. Proc.*, vol. 10, no. 6, pp. 341–351, 2002.
11. R. Chengalvarayan, "Robust energy normalization using speech/non-speech discriminator for German connected digit recognition," in *Proc. of EUROSPEECH 1999*, Budapest, Hungary, Sept. 1999, pp. 61–64.
12. R. Tucker, "Vad using a periodicity measure," *IEE Proceedings, Communications, Speech and Vision*, vol. 139, no. 4, pp. 377–380, 1992.
13. E. Nemer and et al., "Robust vad using hos in the lpc residual domain," *IEEE Trans. S.&A. Proc.*, vol. 9, no. 3, pp. 217–231, 2001.
14. D. Brillinger and et al., *Spectral Analysis of Time Series*. Wiley, 1975, ch. Asymptotic theory of estimates of kth order spectra.
15. T. S. Rao, "A test for linearity of stationary time series," *Journal of Time Series Analysis*, vol. 1, pp. 145–158, 1982.
16. J. Hinich, "Testing for gaussianity and linearity of a stationary time series," *Journal of Time Series Analysis*, vol. 3, pp. 169–176, 1982.
17. J. Tugnait, "Two channel tests for common non-gaussian signal detection," *IEE Proceedings-F*, vol. 140, pp. 343–349, 1993.
18. J. Ramírez and et. al., "An effective subband osf-based vad with noise reduction for robust speech recognition," *In press IEEE Trans. on S.&A. Proc.*
19. J. Ramírez, J. C. Segura, M. C. Benítez, A. de la Torre, and A. Rubio, "Efficient voice activity detection algorithms using long-term speech information," *Speech Communication*, vol. 42, no. 3-4, pp. 271–287, 2004.
20. A. Moreno and et al., "SpeechDat-Car: A Large Speech Database for Automotive Environments," in *II LREC Conference*, 2000.

# Matched Filter as Pre-processing Tool for Multi-user Detection in DS-CDMA System

O. Chakkor<sup>1</sup>, C.G. Puntonet<sup>1</sup>, B. Pino<sup>1</sup>, and J.M. Gorriz<sup>2</sup>

<sup>1</sup> Dept. of Architecture and Computer Technology

<sup>2</sup> Dept. Signal Theory and Communications,

University of Granada, Spain

o.chakkor@iam.ma

**Abstract.** Due to the demand for cellular wireless services, recent interests are in techniques, which can improve the capacity of CDMA systems. On such technique is multi-user detection. Multi-user Detection (MUD) is the intelligent estimation/demodulation of transmitted bits in the presence of Multiple Access Interference (MAI). In this paper, we will show the role of matched filter used as pre-processing tool for MUD in DS-CDMA system.

## 1 Introduction

Code Division Multiple Access (CDMA), as a spread spectrum technique has become widely accepted as the future of cellular systems. In an orthogonal DS-CDMA cellular system, multiple users are allowed to transmit simultaneously by assigning to them orthogonal signature waveforms. The orthogonality of these signature waveforms ensures that each user's signal can be extracted from its superposition with all the other signals at the receiver end. However, theoretical limits on the number of orthogonal sequences places an upper bound on the number of users that can be supported by such a system.

Also practically, the inherent nature of the wireless channel introduces multiple path fading and delays, which destroy the orthogonality of these signature waveforms. In such a scenario, it is often desirable to use signature waveforms with low cross correlation's (almost orthogonal) to ensure minimum degradation in performance. The role of multi-user detection becomes increasingly crucial in non-orthogonal CDMA systems.

A communication system can achieve significant capacity gains if the negative effect of Multiple-User Interference (MUI) can be removed. In multi-user detection we exploit the correlation properties of the signature waveforms of different users to extract the desired signal(s). Then instead of acting as interference, the other users' signals are used for mutual benefit by implementing joint detection at the receiver. The optimal ML receiver for multi-user detection was found by Verdu in [6]. However, this receiver has exponential complexity per bit and thus is not feasible for implementation in a real time system. Since then, a wide range of multi-user receivers have been proposed as a trade-of between complexity and optimality.

In this pursuit, the criteria of optimality have varied from ML sequence estimation to MMSE to marginally ML optimal. An important measure of any multi-user detector is its near-far resistance. In a situation when there is a large variation in the energies of the signals received from different users, the detection of relatively lower energy signals becomes increasingly erroneous even in high SNR regions due to multiple user interference. Near-far resistance is defined as the ability of a receiver to reject the structured interference offered by the other users.

A conventional multi-user detector does not account for the structure of the MUI, considering it as just background noise and does not have good near far resistance.

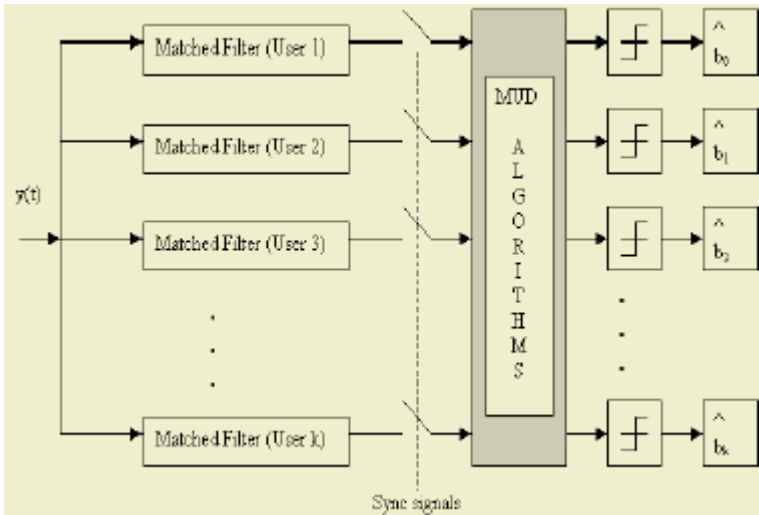


Fig. 1.1. A typical multi-user detector

## 2 Matched Filter Receiver

The simplest scheme to demodulate CDMA signal is to use the conventional matched filter detector, which is optimal in AWGN noise but is sub-optimal because the MAI does not necessarily resemble white Gaussian noise. But in this paper, the purpose is to show the importance for MF detector as pre-processing step of any MUD.

The received signal is passed through a bank of matched filters attached in rake configuration (figure 1.1) that coherently demodulates and dispreads each of the received paths. The problem of this receiver arises from the fact that, even if the powers of all users are equal, some cross correlation among signals might still have high values due to different path delays. Therefore even by adjusting the power level using fast power control and selecting codes with low cross correlations, the performance of the matched filter is limited and so is the capacity since, to maintain acceptable interference limits, the number of users have to be reduced.

The decision statistic at the output of the  $K_{th}$  matched filter is given by:

$$y_k = \int_0^T y(t) \cdot s_k(t) dt \tag{1.1}$$

where  $y(t)$  and  $s_k(t)$  is given by :

$$y(t) = \sum_{k=1}^K A_k b_k s_k(t) + n(t) \tag{1.2}$$

$$s_k(t) = \sum_{k=0}^K a_k \cdot p_T(t - kTc)$$

expanding (1.1), 
$$y_k = \int_0^T \left\{ \sum_{j=1}^K A_j b_j s_j(t) + n(t) \right\} s_k(t) dt \tag{1.3}$$

using  $\rho_{ij} = \langle s_i s_j \rangle = \sum_{k=1}^N s_i(k) s_j(k)$  (1.4)

where  $N$  is the length of the signature sequence. We have :

$$y_k = \sum_{j=1}^K A_j b_j \rho_{jk} + n_k \tag{1.5}$$

where 
$$n_k = \int_0^T n(t) s_k(t) dt \tag{1.6}$$

since  $\rho_{ij} = 1$ , (1.5) simplifies to 
$$y_k = A_k b_k + \sum_{\substack{j=1 \\ j \neq k}}^K A_j b_j \rho_{jk} + n_k \tag{1.7}$$

The 2nd term in (1.7) is the MAI. The matched filter treats the MAI just as white noise.

The noise variance at the output of the matched filter is given by:

$$\begin{aligned} E(n_k^2) &= E \left[ \int_0^T n(t) s_k(t) dt \int_0^T n(s) s_k(s) ds \right] = \int_0^T \int_0^T E[n(t)n(s)] s_k(s) s_k(t) dt ds \\ &= \int_0^T \int_0^T N_0 \delta(t-s) s_k(s) s_k(t) dt ds = \int_0^T N_0 s_k^2(t) dt = N_0 \end{aligned}$$

Similarly, the noise covariance can be shown to be :

$$E(n_i, n_j) = N_0 \rho_{ij}$$

hence the noise covariance matrix can be defined as :

$$E(nn^T) = \{N_0\rho_{ij}\}_{ij} = N_0R$$

where  $R$  is given by :

$$R = \begin{bmatrix} \rho_{11} & \rho_{12} & \dots & \rho_{1k} \\ \rho_{21} & \rho_{22} & \dots & \rho_{2k} \\ \vdots & \vdots & & \vdots \\ \rho_{k1} & \rho_{k2} & & \rho_{kk} \end{bmatrix} \quad \text{and } n = [n_1, n_2, \dots, n_k]^T$$

Stacking up (1.7) for all the users we get in matrix notation:  $y=RAb+n$ , and we use in our simulation the two types of MUD algorithms:

|                        | MUD Algorithm |
|------------------------|---------------|
| Decorrelating detector | $R^{-1}$      |
| MMSE linear detector   | $R+N_0A^{-2}$ |

### 3 Independent Component Analysis and CDMA

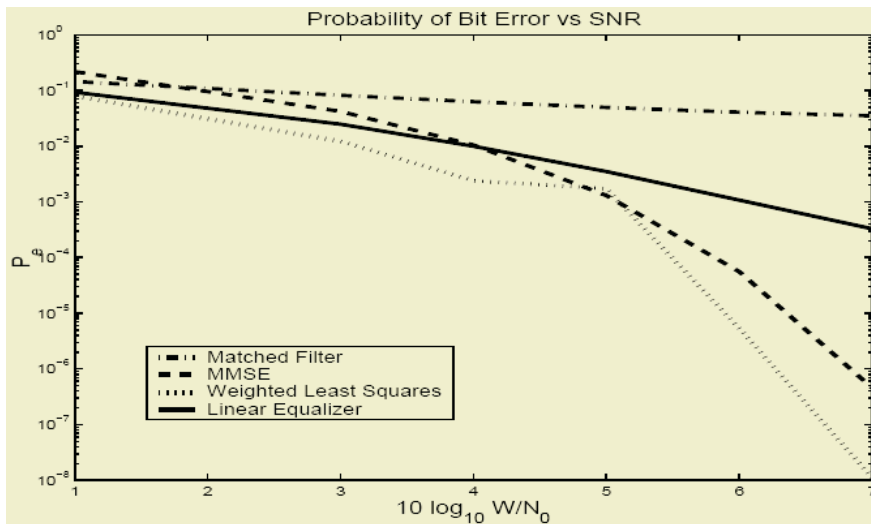
Independent component analysis is a statistical technique where the goal is to represent a set of random variables as a linear transformation of statistically independent component variables. One of the most promising solution for linear ICA/BSS problem is FastICA [8] due its simplicity and fast convergence.

To obtain a better performance, we use the RAKE-ICA as a receiver in DS-CDMA downlink system used in [1]. ICA method try to estimate the source process assuming nothing about the mixture but its linearity. The novel scheme of receiver ICA-based receiver uses the statistical independence of signals using FastICA. ICA is considered here as an additional element, attached to RAKE receiver.

### 4 Simulation

In this section we give simulation results for the performance of matched filter and the other linear receivers when the MF detector is used as pre-processing step in DS-CDMA system in the presence of MUI and thermal noise. We consider the bit error rate probabilities BER of different receivers using MF in first step of the signal-to-noise ratio SNR for K=18. pseudorandom noise are used as the spreading code of the users of length N=127.

We can observe that the linear receivers are effective in over coming the near-far problem, and perform much better that the MF detector. This implies that the MF detector is usually used as front end of MUD to minimizes the probability of error  $P_e$ .



**Fig. 1.2.** BER vs SNR for conventional matched filter, MMSE detector, linear detector and WLS receiver for  $K=18$  and  $N=127$

## 5 Conclusion

The choice of the MUD algorithm depends on a lot of factors like the application, channel information available, availability of training sequences, complexity cost and overhead involved. To pick one out which is the optimal one or the best one is not an easy task! and the matched filter is used as the front end of any MUD. Most MUD therefore have the matched filter as the front end.

## References

- [1] O.chakkor,"A new algorithm for symbol separation in a DS-CDMA system downlink, Phd-thesis 2004.
- [2] R. Lupas and S.Verdu, "Linear multi-user detectors for synchronous code-division multiple-access channels," IEEE Trans. Inform. Theory, vol. 35 pp. 123-136, Jan. 1989.
- [3] Z. Xie, R. T. Short and C. K. Rushforth, "A family of suboptimum detectors for coherent multiuser communications," IEEE J. Selected Areas Commun., pp. 683-690,May 1990.
- [4] S. Verdu, "Multiuser detection," in Advances in Detection and Estimation, JAI Press,1993.
- [5] U. Madhow and M. Honig, "MMSE interference suppression for direct sequence spreadpectrum CDMA," IEEE Trans. Commun., vol. 42, pp. 3178-3188, Dec. 1994.
- [6] S. Verdu, "Minimum probability of error for asynchronous gaussian multiple-access channels," IEEE Trans. Inform. Theory, vol. IT-32, pp. 85-96, Jan. 1986.
- [7] C. G. Puntonet, A. Mansour, C. Jutten, "*Geometrical Algorithm for Blind Separation of Sources*", Proc. of 15me. Colloque sur le Traitement du Signal et des Images (GRETSI'95), Vol.1, pp. 273-276. Juan les Pins, 18-22 Sept. 1995.
- [8] A. Hyvarinen and E.Oja, " A fast Fixed-Point Algorithm for Independent Component Analysis", Neural Computation, 9:1483-1492, 1997.



# A Robust Multiple Feature Approach to Endpoint Detection in Car Environment Based on Advanced Classifiers\*

C. Comas<sup>1</sup>, E. Monte-Moreno<sup>1</sup>, and J. Solé-Casals<sup>2</sup>

<sup>1</sup> TALP Research Center,  
Universitat Politècnica de Catalunya, Spain  
enric@gps.tsc.upc.es

<sup>2</sup> Signal Processing Group, University of Vic, Spain  
jordi.sole@uvic.es

**Abstract.** In this paper we propose an endpoint detection system based on the use of several features extracted from each speech frame, followed by a robust classifier (i.e Adaboost and Bagging of decision trees, and a multilayer perceptron) and a finite state automata (FSA). We present results for four different classifiers. The FSA module consisted of a 4-state decision logic that filtered false alarms and false positives. We compare the use of four different classifiers in this task. The look ahead of the method that we propose was of 7 frames, which are the number of frames that maximized the accuracy of the system. The system was tested with real signals recorded inside a car, with signal to noise ratio that ranged from 6 dB to 30dB. Finally we present experimental results demonstrating that the system yields robust endpoint detection.

## 1 Introduction

In speech and speaker recognition a fast and accurate detection of the speech signal in noise environment is important because the presence of non-voice segments or the omission of voice segments can degrade the recognition performance [2,3,4]. On the other hand in a noise environment there are a set of phonemes that are easily masked, and the problem of detecting the presence of voice cannot be solved easily. The problem is further complicated by the fact that the noise in the environment can be time variant and can have different spectral properties and energy variations. Also there are limitations on the admissible delay between the input signal, and the decision of the presence or absence of voice. Therefore the variability of the environment justifies the use of different features, which might be adapted to discriminate voice from different kind of environmental noise sources. The variability of circumstances justifies the use of the aggregation of classifiers which are trained differently. The use of FSA is justified by the fact that the classifiers make bursts of nearly consecutive mistakes; these bursts of false alarms or false positive decision are easily filtered by the FSA.

---

\* This work has been partially supported by the Spanish CICYT project ALIADO, the EU integrated project CHIL and the University of Vic under the grant R0912.

## 2 General Structure of the System

We designed a system which consisted of a frame level classifier followed by a FSA. The idea behind the design was to use a robust classifier with input features, which in isolation have proved to yield good performance in different on the environmental conditions, and a FSA which implemented the decision logic that filters short bursts of nearly consecutive false alarms or false positives.

### 2.1 Selected Features

The selected features were:

- The Teager energy [1] is a measure of the energy of a system in a simple harmonic motion, which is  $E \propto A^2 \omega^2$ . In [2] this measure is proposed for endpoint detection. We used the windowed mean of this energy of frame ‘j’:

$$TE_j = \sum_{i=1}^n [x^2(i) - x(i+1)x(i-1)] w(i) \tag{1}$$

- Differential Teager energy: The derivative of the Teager energy was computed by filtering the Teager energy with:  $H(z) = 1 - z^{-2}$
- Zero crossing rate: was computed for each frame by

$$CX_j = \sum_{i=1}^n | \text{sign}(x(i)) - \text{sign}(x(i-1)) | w(i) \tag{2}$$

- Spectral entropy: [3]: This method is based on measuring the normalized spectral power density function for each frame, which can be interpreted as the probability of a certain frequency. The associated entropy is computed, and non discriminative frequencies are windowed out

$$H_j = - \sum_{k=1}^n w_k p_k \log p_k \tag{3}$$

In case of speech, for certain phonemes, the energy is concentrated in a few frequency bands, and therefore will have low entropy, while in the case of noise with flat spectrum or low pass noise, the entropy will be higher.

- Spectral coherence between consecutive frames gives a measure of the similarity between of two frames

$$|\gamma(\omega)|^2 = \frac{S_{xy}(\omega)S_{yx}(\omega)}{S_x(\omega)S_y(\omega)} \tag{4}$$

We use as feature the values of  $|\gamma(\omega)|^2$  computed by means of the DFT.

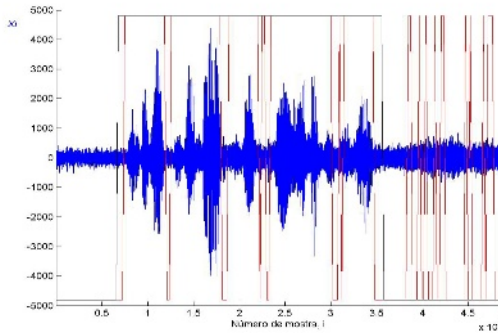
### 2.2 Classifiers

We did experiments with four different classifiers: a linear discriminant, the AdaBoost of linear classifiers the bagging of decision trees, and a Multilayer Percep-

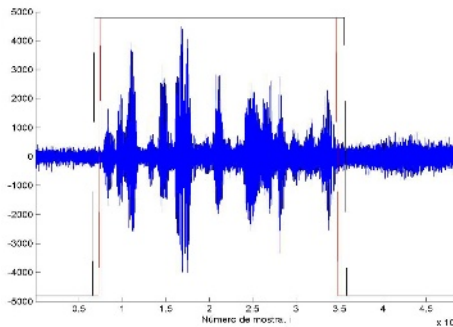
tron (MLP). The linear discriminant was selected as benchmark. As can be seen in figure 1, when the classification (with AdaBoost) is done at the frame level without the finite state automata, there is a high number of false alarms. This phenomenon was common to all classifiers. The distribution of the false alarms is such that they can be easily filtered by the finite state automata. Nevertheless, it also means that at the frame level the confusion between classes is high. The use of a decision tree alone was discarded because of the poor results, when the tests were done with noisy signals. The decision trees in these cases grow specific branches for similar cases with different labels, which means that the decision trees tend to grow in excess. Pruning the trees degraded abruptly the performance. The best results with a decision tree were slightly worse than the linear discriminant.

We do not present results with support vector machines either, because the high overlap between classes in the feature space in the low SNR case, yields bad results and an extremely high number of support vectors.

The use of bagging of the decision trees is justified because of this low ‘hit’ rate at frame level. Bagging decision trees improved the performance because we were able to grow trees (trained by bootstrapping on the training database) with a high number



**Fig. 1.** Segmentation done by a frame level classifier (AdaBoost), for the utterance: ‘seleccionar centro de la ciudad’. Note the false alarms and false negative points



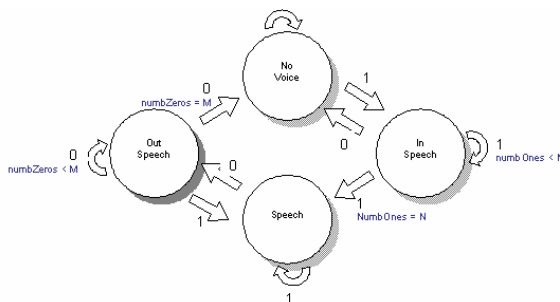
**Fig. 2.** Segmentation done by the FSA on the frame level classification. Comparison on an example of a manual segmentation (dark line) with the segmentation done by our system (clear line)

of nodes, i.e. adapted to the specificities of the training database, and afterwards the aggregation of trees smoothed the variance and therefore reduced the error rate [5].

The AdaBoost [6] was selected in order to improve the accuracy of a linear discriminant. As the classification rate of a linear discriminant is low, the use of AdaBoost creates a set of classifiers that specialize in the distribution of the misclassified frames in the feature space. Also in the case of AdaBoost it is known that the performance degrades when there is a high overlap between cases in the feature space, consequently in order to reduce the degradation caused by this overlap, we selected the number of classifiers by cross validation with a criterion based on the final recognition rate, i.e. the accuracy after the finite state automata. The use of AdaBoost in combination with decision trees was discarded because the preliminary experiments yielded extremely big trees, which were computationally prohibitive. This is explained because the high overlap between classes made the trees to specialize in contradictory examples.

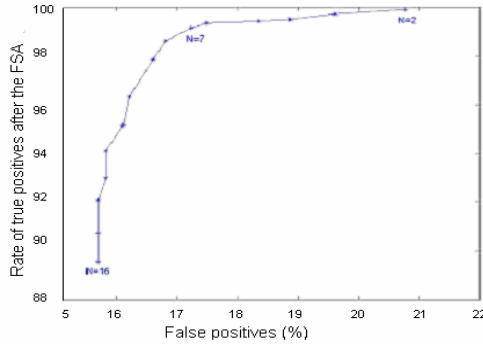
### 2.3 The FSA and the Decision Criterion

After the classifier step we used as decision logic a FSA, which had four states, and the transition from one state to the other was controlled by the number of frames that corresponded to each class:  $1 \rightarrow$ voice,  $0 \rightarrow$ no voice. A diagram is shown in figure 3. The parameters for the FSA:  $N$  = number of contiguous voice frames,  $M$  = number of contiguous non voice frames; were tuned by a compromise between: the number of consecutive frames of false negatives, frames of false positives, and the rate of real negatives (absence of voice) and real positives (presence of voice) after applying the FSA. The rates were computed on the training database. The compromise was obtained by inspection of the ROC curve (Receiver Operating Characteristics), (see figure 4) and histograms of duration of bursts of false negatives, and false positives. These parameters were adjusted as follows:



**Fig. 3.** Finite State Automata that filters the bursts of false alarms and false negatives

We plotted histograms of: a-consecutive false positive frames, and real positive (consecutive frames of voice), for different values of  $M$ , and b-histograms of consecutive false negative frames, as compared with the real negative frames (absence of voice), for different values of  $M$ .



**Fig. 4.** The ROC obtained by the use of the FSA, with  $M=7$  frames as out-speech for the channel with a  $SNR=8,5dB$ . The frame level classifier was based on the AdaBoost

We selected the value of  $M$  that in both cases corresponded approximately to the crossing of the false positive and real positive cases. Then with the value of  $M$  fixed, we plotted the ROC curve for different values of  $N$  and selected a value of  $N$  that gave a hit rate of 98%.

### 3 Experimental Results

The experiments were done with a subset of the SpeechDat Car [7], in Spanish. The subset consisted of 100 files with speakers of both sexes, and phrases of different kinds. The files were hand labeled. Each file contained four recordings of the same signal, one recorded near the mouth of the speaker, which had a mean  $SNR$  of about 30dB. The other three were recorded in different places of the car, and had a mean  $SNR$  of about 8,5 dB. The sampling frequency was 16 kHz. The signal was divided into frames of 33 ms, with an overlap of 50%. A pre-emphasis was done before the processing of the signal. For each frame we computed the following features: Teager energy, Differential Teager energy, zero crossings, spectral entropy, and spectral coherence between successive frames. Then the classifier made a decision at frame level, and the FSA made a final labeling based on the history of the decisions made by the classifier. Therefore the delay introduced by the system is the delay of the FSA, which is of 7 frames for ‘in speech’ and ‘out speech’.

The endpoint detection results were computed by a 5 fold cross validation. This was done in order not to make the results dependent of the random distribution of files between training and testing. This gave a mean of 13500 for train and 3300 frames for test in each cycle of the cross validation process. Of these frames approximately 65% was no-voice and 35% voice.

The number of linear discriminants in the AdaBoost was selected to be 10 after experiments with cross validation on the training database. For comparison purposes, the number of trees in the bagging experiment was selected also to be 10. Increasing the number of bagged trees did not give significant improvements. The topology of the MLP was decided by the performance on the training database, and the training was stopped by the performance on a validation subset of the training database.

In order to quantify the performance of the system we will use the following parameters:

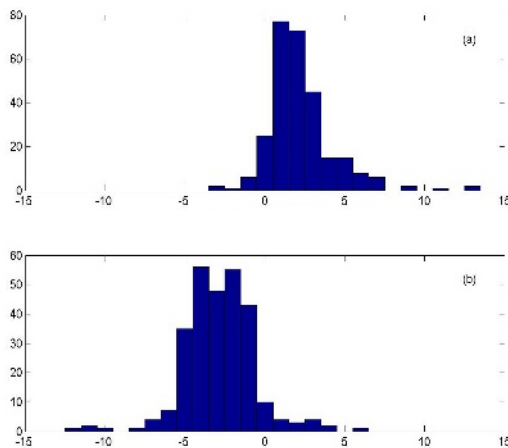
- Accuracy (ACC), the ratio between the total numbers of predictions that were correct to the total number of cases.
- True positive rate (TPR), which is the ratio of positive cases that were correctly classified to the total number of positive cases.
- False positive rate (FPR), which is the ratio of negative cases that were incorrectly classified as positive to the total number of negative cases.
- Precision (PRC), which is the ratio of predicted positive cases that were correct to total number of positive cases.

**Table 1.** Results (%) of a channel with mean SNR of 30dB

|     | Linear Classifier | AdaBoost linear classifiers | Bagging decision trees | MLP(5;3;1) |
|-----|-------------------|-----------------------------|------------------------|------------|
| ACC | 86                | 92                          | 92                     | 93         |
| TPR | 77                | 89                          | 89                     | 91         |
| FPR | 0.6               | 3                           | 2                      | 2          |
| PRC | 99                | 97                          | 98                     | 98         |

**Table 2.** Results (%) of a channel with mean SNR of 8dB

|      | Linear Classifier | AdaBoost linear classifiers | Bagging decision trees | MLP(5;3;1) |
|------|-------------------|-----------------------------|------------------------|------------|
| ACC. | 78                | 83                          | 81                     | 84         |
| TPR  | 77                | 85                          | 82                     | 86         |
| FPR  | 19                | 18                          | 20                     | 18         |
| PRC. | 85                | 87                          | 86                     | 87         |



**Fig. 5.** Histogram of the distance of a frame to the nearest endpoint. Upper histogram is the transition from ‘no voice’ to voice. Lower from voice to ‘no voice’

The results are presented in table 1 and 2. We have not presented the confidence margins. Due to the high number of frames, in all cases they lower than  $10e-3$ . The results have to be interpreted in the light of figure 5. Most of the errors came from a misplacement of the transition between voice/no voice, which in most cases can be quantified in a few frames. The manual labeling of the boundary between the segments of voice/ no voice is subjective in a margin of a few frames. This is reflected in the fact that the histogram of placement of the boundary is slightly biased to the right or left depending on the case of beginning or ending of speech. This bias comes from the fact that at this point the classifiers gives bursts of false alarms, which are filtered by the FSA. Figure 2 shows an example of this bias on the endpoints.

## 4 Conclusions

In this paper, we propose a system for real-time endpoint detection. The system is based in a frame level classifier followed by a finite state automaton that filters false alarms or false positives. The frame level classifier is based in five different features. The delay introduced is of 7 frames. The experiments were done on speech recorded in a car environment. In the future we will evaluate the endpoint detection with a speech recognition system.

## References

1. J.F. Kaiser, "On a Simple Algorithm to Calculate the Energy of a Signal," *Proc. ICASSP*, 381-384, 1990.
2. G. S. Ying, C. D. Mitchell, and L. H. Jamieson, "Endpoint Detection of Isolated Utterances Based on a Modified Teager Energy Measurement", *Proc. ICASSP*, II.732-735, 1993.
3. Jia-lin Shen, Jieh-weih Hung, Lin-shan Lee, "Robust Entropy-based Endpoint Detection for Speech Recognition in Noisy Environments", *Proc. ICSLP CD-ROM* 1998.
4. W.-H.Shin, B.-S.Lee,Y.-K.Lee,J.-S.Lee, "Speech/Non-Speech Classification Using Multiple Features For Robust Endpoint Detection", *Proc. ICASSP*, 1399-1402, 2000.
5. L. Breiman. "Bagging predictors". *Machine Learning*, 24(2):123-140, 1996
6. Y. Freund and R. E. Schapire."Experiments with a new boosting algorithm". *Proc. 13 th International Conference*, pp 148--156. Morgan Kaufmann, 1996.
7. Asunción Moreno, Borge Lindberg, Christoph Draxler, Gaël Richard, Khalid Choukri, Stephan Euler, Jeff Allen "SPEECH DAT CAR. A Large Speech Database For Automotive Environments" *Proc. of the II Language Resources European Conference*. Athens, 2000

# Canonical Correlation Analysis Using for DOA Estimation of Multiple Audio Sources

Gaoming Huang<sup>1,2</sup>, Luxi Yang<sup>2</sup>, and Zhenya He<sup>2</sup>

<sup>1</sup> Naval University of Engineering, Wuhan, China 430033

<sup>2</sup> Department of Radio Engineering, Southeast University,  
Nanjing, China, 210096

redforce@sohu.com, {lxyang, zyhe}@seu.edu.cn

**Abstract.** In this paper we study direction of arrival (DOA) estimation of multiple audio sources by canonical correlation analysis (CCA), which is based on a sparse linear arrays. This array is composed of two separated subarrays. From the receiving data set, we can obtain the separate components by CCA. After a simple correlation, time difference can be obtained, and then we can compute the azimuth of different audio sources. The important contribution of this new estimation method is that it can reduce the effect of inter-sensor spacing to DOA estimation and the computation burden is light. Simulation result confirms the validity and practicality of the proposed approach. Results of DOA estimation are more accurate and stable based on this new method.

## 1 Introduction

Multiple audio sources direction of arrival (DOA) estimation is of great interest to many applications, such as hearing aids, fault location and target tracking etc. A sparse linear array is considered as the main conditions. The main problem of this topic is how to separate different sources only by the receiving signals. People often think of applying blind source separation (BSS) on it [1]. But BSS method exists one fatal disadvantage is how to determine the inter-sensor spacing. In other words, this method can not solve the contradictory of phase ambiguity and signal correlation [2]. Consider about the simplicity and practicality, we propose a new DOA estimation method based on canonical correlation analysis (CCA), which is an important method of multivariate statistical since it was proposed by H.Hotelling [3]. The main character of CCA is it can find the basic vectors from two sets of variables, similarly to our ears. There are detail descriptions of CCA in [4][5]. CCA has been applied in some preliminary work [6][7][8] in recent years. Uncorrelated components can be obtained by CCA, in addition, has maximum spatial or temporal correlation within each component. Then we can apply CCA to seek the correlate components of the data from double receiving sensors. After a general correlation of the two canonical components, we can obtain the time difference and complete the estimation of DOA based on time difference. The structure of this paper is as follows: In the Section 2, we will analyze the issue of multiple audio sources DOA estimation and pose the problem. In the Section 3, the estimation algorithm will be analyzed. Then some experiments of the algorithm applied in this paper are conducted in the Section 4. Finally a conclusion is given.



## 2 Problem Formulation

The model of DOA estimation between the audio sources received by separate sensor is shown as in Fig. 1.

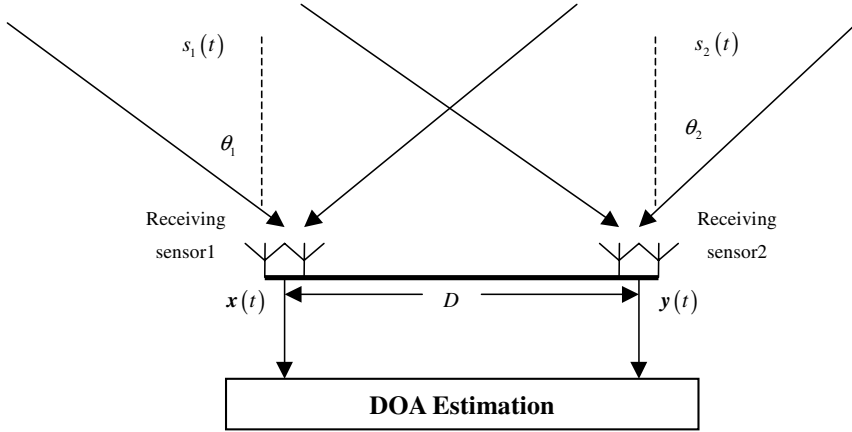


Fig. 1. System receiving model

In generally, supposing there are  $m$  receiving sensors, the distance between the phase centers of the subarrays is  $D$ . If have  $n(n \leq m)$  audio signals,  $\mathbf{r}(\theta_i)$  is the sensor respond to the signal on direction  $\theta_i$ , then  $\mathbf{r}(\theta_i) = [1, e^{j\phi_i}, \dots, e^{j(m-1)\phi_i}]^T$ , where  $\phi_i = 2\pi D \sin \theta_i / \lambda$ ,  $\lambda$  is the wavelength. The receiving model could be described as:

$$\mathbf{x}(t) = \mathbf{H} \cdot \sum_{i=1}^n \mathbf{r}(\theta_i) s_i(t - \tau_{ij}) + \mathbf{n}(t) = \mathbf{A} \cdot \mathbf{s}(t) + \mathbf{n}(t) \tag{1}$$

where  $\mathbf{x}(t)$  is the receiving signals.  $\mathbf{A}$  is a  $m \times n$  mixing matrix, which is the product of responding function  $\mathbf{R} = [\mathbf{r}(\theta_1), \dots, \mathbf{r}(\theta_n)]$  and the mixing matrix  $\mathbf{H}$  during the signal transmitting process.  $\mathbf{s}(t)$  are the source signals.  $\tau_{ij}$  are the delays from source  $i$  to sensor  $j$ .  $\mathbf{n}(t)$  are  $m \times 1$  dimension noise signals. The signals are assumed as mutually independent and independent with noise in the following analysis. In order to describe the problem more clearly, here we assume that there are two audio sources and two receiving sensors, which includes two inter-sensors separately. As the inter-sensor spacing is more less than the distance of audio sources to receiving sensors, the receiving model can be simplified as:

$$\begin{cases} x_1(t) = a_{11}s_1(t) + a_{12}s_2(t - \tau_2) \\ x_2(t) = a_{21}s_1(t) + a_{22}s_2(t - \tau_2) \end{cases} \quad \begin{cases} y_1(t) = b_{11}s_1(t - \tau_1) + b_{12}s_2(t) \\ y_2(t) = b_{21}s_1(t - \tau_1) + b_{22}s_2(t) \end{cases} \tag{2}$$

The key problem of DOA estimation of multiple audio sources is how to estimate the time difference of each source to sensors in such setting environments, which is the basic condition of DOA estimation and the main problem that this paper wants to solve.

### 3 DOA Estimation Algorithm

The new DOA estimation algorithm includes three steps: The first step is CCA, by which we can obtain the separate sources from the two receiving sensors. Then we conduct a correlation of the two groups of separate sources, time difference of two audio sources can be obtained. The last step is DOA estimation by the time difference estimation results.

#### 3.1 Canonical Correlation Analysis

The main difference between CCA and the other statistical methods is that CCA is closely related to mutual information [6]. Hence CCA can be easily motivated in information based on tasks and our natural selection.

Consider two sets of input data  $x_1, x_2, \dots, x_p$  and  $y_1, y_2, \dots, y_q, p \leq q$ , we attempt to find the coefficient  $\mathbf{a} = (a_{i1}, a_{i2}, \dots, a_{ip})$  and  $\mathbf{b} = (b_{i1}, b_{i2}, \dots, b_{iq})$  by the idea of principle components. The two sets of data can be written as combination of some pairs of variables  $\xi_i$  and  $\eta_i$ , which can be described as:

$$\begin{cases} \xi_1 = a_{11}x'_1 + \dots + a_{1p}x'_p & \eta_1 = b_{11}y'_1 + \dots + b_{1q}y'_q \\ \dots & \dots \\ \xi_p = a_{p1}x'_1 + \dots + a_{pp}x'_p & \eta_p = b_{p1}y'_1 + \dots + b_{pq}y'_q \end{cases} \quad (3)$$

where  $x', y'$  are the standardization value of  $x, y$  respectively.  $\xi_1, \eta_1$  are the first pair of canonical variable, the correlation coefficient can be describe as  $r_{\xi_1\eta_1}$ , brief written as  $r_1$ .  $\xi_2, \eta_2$  are the second pair of canonical variable, the correlation coefficient can be describe as  $r_{\xi_2\eta_2}$ , brief written as  $r_2$ . Then  $p$  pairs of canonical variables and  $p$  canonical correlation coefficients can be obtained.

Mutual independent variables can be obtained by the method of canonical correlation. Here we introduce a theorem [5] (which will be proven in appendix A).

**Theorem.** Let  $\mathbf{x} = (x_1, x_2, \dots, x_p)^T, \mathbf{y} = (y_1, y_2, \dots, y_q)^T$  are two sets of random variables,  $\text{cov}(\mathbf{x}) = \Sigma_{xx}, \text{cov}(\mathbf{y}) = \Sigma_{yy}, \text{cov}(\mathbf{x}, \mathbf{y}) = \Sigma_{xy}, \text{cov}(\mathbf{y}, \mathbf{x}) = \Sigma_{yx}, \Sigma_{xx}, \Sigma_{yy}$  are positive definite, then  $\Sigma_{xx}^{-1}\Sigma_{xy}\Sigma_{yy}^{-1}\Sigma_{yx}$  and  $\Sigma_{yy}^{-1}\Sigma_{yx}\Sigma_{xx}^{-1}\Sigma_{xy}$  have the same non-zero latent roots  $\lambda_1^2 \geq \lambda_2^2 \geq \dots \geq \lambda_r^2 > 0$ . If their mutual orthogonal identity eigenvectors are  $\alpha_1, \alpha_2, \dots, \alpha_r$  and  $\beta_1, \beta_2, \dots, \beta_r$  respectively, then  $\mathbf{a}_i^* = \Sigma_{xx}^{-1/2}\alpha_i, \mathbf{b}_i^* = \Sigma_{yy}^{-1/2}\beta_i$  ( $i=1, 2, \dots, r$ ) are the  $i$ th pair of canonical correlation variables,  $\lambda_i$  is the  $i$ th canonical correlation coefficient.

By the theorem, canonical correlation variable and coefficients can be obtained by the follow steps:

**Step 1:** computing the correlations of the two sets of variables as:

$$\Sigma = \begin{bmatrix} r_{x_1x_1} & \cdots & r_{x_1x_p} & r_{x_1y_1} & \cdots & r_{x_1y_q} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ r_{x_px_1} & \cdots & r_{x_px_p} & r_{x_py_1} & \cdots & r_{x_py_q} \\ r_{y_1x_1} & \cdots & r_{y_1x_p} & r_{y_1y_1} & \cdots & r_{y_1y_q} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ r_{y_qx_1} & \cdots & r_{y_qx_p} & r_{y_qy_1} & \cdots & r_{y_qy_q} \end{bmatrix} = \begin{bmatrix} \Sigma_{xx} & \Sigma_{xy} \\ \Sigma_{yx} & \Sigma_{yy} \end{bmatrix} \quad (4)$$

The right matrix in (4) is a partitioned matrix.

**Setp 2:** Computing the canonical correlation coefficients  $r_i$ . Firstly, we compute two matrices  $L$  and  $M$ , where  $L = \Sigma_{xx}^{-1} \Sigma_{xy} \Sigma_{yy}^{-1} \Sigma_{yx}$ ,  $M = \Sigma_{yy}^{-1} \Sigma_{yx} \Sigma_{xx}^{-1} \Sigma_{xy}$ , secondly, we compute the eigenvalue  $\lambda_i$  of matrix  $L$  and  $M$ , we can obtain  $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p \geq \cdots \geq \lambda_p$ , then  $r_i = \sqrt{\lambda_i}$  can be obtained.

**Step 3:** Computing the canonical variables  $\xi_i$  and  $\eta_i$ . First computing the eigenvectors of matrix  $L$  about  $\lambda_i$ , we can obtain the coefficient matrix  $A$ . Second computing the eigenvectors of matrix  $M$  about  $\lambda_i$ , we can obtain the coefficient matrix  $B$ . Thirdly we introduce standardization variables  $x'_j$  and  $y'_j$ , then the canonical variables  $\xi_i$  and  $\eta_i$  can be obtained.

Then canonical variables  $\xi_i$  and  $\eta_i$  can be obtained in turns, which is the basis of time difference estimation. From the two receiving data sets, two groups of separate sources can be obtained by CCA.

### 3.2 Time Difference Estimation

After the canonical variables  $\xi_i$  and  $\eta_i$  having been obtained, the following step is to estimate the time difference  $\Delta\tau$  of the two independent receiving signals, viz. the canonical variables  $\xi_i$  and  $\eta_i$ . Here we assuming  $s_{x1}(n), s_{x2}(n)$  and  $s_{y1}(n), s_{y2}(n)$  are the corresponding separate audio sources by CCA. To a same audio source, the receiving time is different as the as the different receiving position. In this case, we must consider the resemble property of the two signals must be considered during the time varying. Within the observation time  $T$ , assuming the total number of samples is  $N$ , the correlation function can be described as:

$$\begin{cases} \hat{R}_{s_1}(\tau) = \frac{1}{N-\tau} \sum_{k=\tau}^{N-1} s_{x1}(k) s_{y1}(k-\tau) \\ \hat{R}_{s_2}(\tau) = \frac{1}{N-\tau} \sum_{k=\tau}^{N-1} s_{x2}(k) s_{y2}(k-\tau) \end{cases} \quad (5)$$

Then the separate time difference can be obtained as:

$$\begin{cases} \tau_1 = \arg \left\{ \max_{\tau} \left[ \hat{R}_{s_1}(\tau) \right] \right\} \\ \tau_2 = \arg \left\{ \max_{\tau} \left[ \hat{R}_{s_2}(\tau) \right] \right\} \end{cases} \quad (6)$$

As the distance between the phase centers of the subarrays is fixed, we can begin the DOA estimation by the time difference  $\tau_1, \tau_2$ .

### 3.3 DOA Estimation

After time difference  $\tau_1, \tau_2$  have been obtained, the estimation of DOA is relatively simple. In order to clarify the estimation of DOA clearer, here we set the time difference is  $\Delta\tau$ . A simplified coordinate can be established as in Fig. 2.

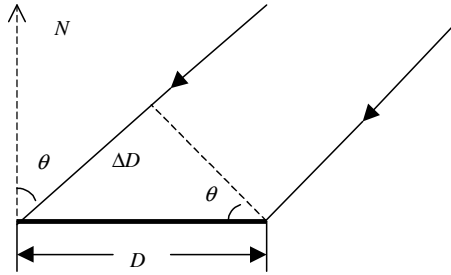


Fig. 2. DOA estimation by time difference

From Fig.2, we can get  $\Delta D = D \sin \theta$ ,  $\Delta \tau c = \Delta D = D \sin \theta$ , where  $c = 340.29m/s$  is the velocity of sound, DOA can be obtained as:

$$\theta = \arcsin((c \cdot \Delta\tau)/D) \tag{7}$$

From (7) we can see that the precision of DOA decided by  $D$  and the measure precision  $\Delta\tau$ , where  $\Delta\tau = (D \sin \theta)/c$ . Partial derivative of  $\Delta\tau$  can be written as:

$$d(\Delta\tau) = \frac{\partial(\Delta\tau)}{\partial\theta} d\theta + \frac{\partial(\Delta\tau)}{\partial D} dD + \frac{\partial(\Delta\tau)}{\partial c} dc \tag{8}$$

where  $D, c$  are all constant, so the result of partial derivative is  $d(\Delta\tau) = D \cos \theta d\theta/c$ , then the estimation precision can be written as:

$$d\theta = (c \cdot d(\Delta\tau))/(D \cos \theta) \tag{9}$$

From (9) we can know that the precision of DOA is limited by the distance between the phase centers of the subarrays and the time difference estimation precision, it also relates to the direction of sources.

## 4 Simulations

In order to verify the validity of this DOA estimation algorithm applied in this paper, here a series of experiments have been conducted. The background of the experiments is assumed as: there is two spatially separated receiving subarrays. The distance be-

tween the phase centers of the subarrays is  $3m$ , two audio sources come from different direction and the sampling frequency is  $8000\text{Hz}$ , time differences are set as  $\tau_1 = 28$  and  $\tau_2 = 32$ . According to the DOA estimation algorithm proposed in this paper, CCA was conducted at first. The receiving signals are shown in Fig. 3 and the separation results by CCA are shown in Fig.4. The details are shown in Fig. 5.

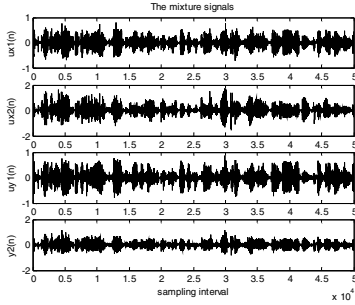


Fig. 3. Mixing signals

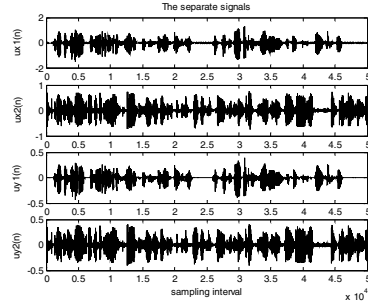


Fig. 4. Separate signals by CCA

Then a cross correlation between the corresponding separation signals is conducted. The correlation results are shown as in Fig.6.

From the correlation results, the time difference of different audio source can be obtained as follow:  $\tau_1 = -28, \tau_2 = 32$ , which is corresponding with the experiments setting. Submitting these results to (7), we can obtain that  $\theta_1 = -23.39^\circ, \theta_2 = 26.98^\circ$ . However, if by direct correlation method or GCC method, we often can not obtain the different time difference.

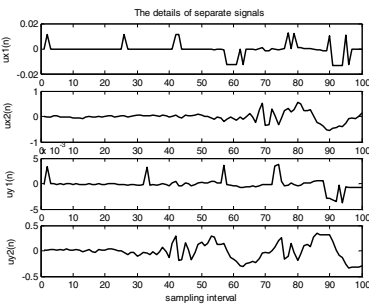


Fig. 5. Separation details

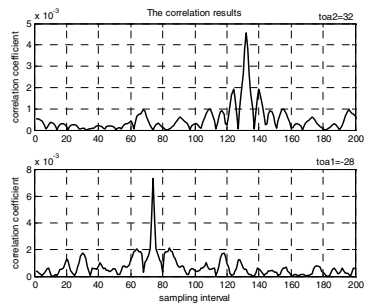
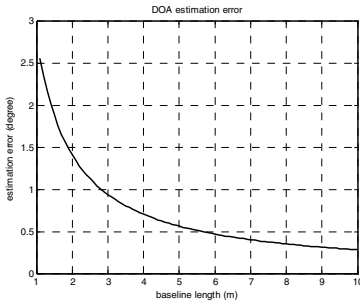


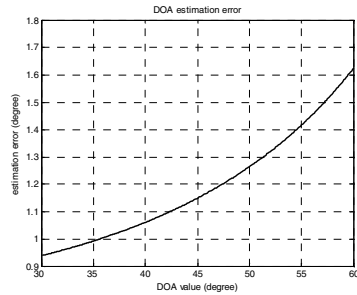
Fig. 6. Correlation results

After completing the estimation of DOA, here we conduct some experiments to analysis the error, which is corresponding with the varying of distance between the phase centers of the subarrays, precision of time difference estimation and DOA. The

estimation error results are separately shown in Fig.7 and Fig.8. From the results, we can see that the estimation precision is not only decided by the estimation precision of time difference, but also decided by other parameter.



**Fig. 7.** DOA estimation error while distance between the phase centers of the subarrays changing



**Fig. 8.** DOA estimation error while the direction of the audio sources changing

From the error curve of different case, we can see that DOA estimation error is inverse proportion to distance between the phase centers of the subarrays. However, DOA estimation error is direct proportion to DOA value within a definite range.

## 5 Conclusion

In this paper, we have investigated the fundamental limitations in multiple audio sources DOA estimation with a sensor array consisting of two spatially separated subarrays. We propose a novel DOA estimation method based on CCA, which can effectively overcome the contradictory of phase ambiguity and signal correlation and will play an important role in military and civilian affairs.

**Acknowledgement.** This work was supported by NSFC (60496310, 60272046), National High Technology Project (2002AA123031) of China, NSFJ (BK2002051) and the Grant of PhD Programmers of Chinese MOE (20020286014).

## References

1. Torkkola, K.: Blind separation of delayed sources based on information maximization. Proc. IEEE ICASSP, (1996) 3509-3512
2. Huang, G.M., Yang, L.X. and He, Z.Y.: Time-Delay Direction Finding Based on Canonical Correlation Analysis, accepted by ISCAS 2005, Kobe, Japan, May 23-26, 2005
3. Hotelling, H.: Relations between two sets of variates, *Biometrika*, 1936(28) 321-377
4. Anderson, T. W.: An Introduction to Multivariate Statistical Analysis. John Wiley & Sons, second edition, 1984

5. Zhang, R.T. and Fang ,K.T.: An Introduction to Multivariate Statistical Analysis. Science House, 1982
6. Borga, M.: Learning Multidimensional Signal Processing. PhD thesis, Linköping University, Sweden, SE-581 83 Linköping, Sweden, 1998. Dissertation No 531, ISBN 91-7219-202-X. 1998
7. Bach, F. R. and M. I. Jordan: Kernel independent component analysis, Journal of Machine Learning Research, 2002(3) 1-48
8. Fyfe, C. and Lai, P. L.: Ica using kernel canonical correlation analysis. ICA2000, 2000(8) 279-284

### Appendix A: Proof of Theorem

Computing the variable is to choose  $\mathbf{a}, \mathbf{b}$  in the condition of  $\mathbf{a}^T \Sigma_{xx} \mathbf{a} = 1, \mathbf{b}^T \Sigma_{yy} \mathbf{b} = 1$ , then we will obtain the maximum value of  $\mathbf{a}^T \Sigma_{xy} \mathbf{b}$ . Taking Lagrangean function as the assistant function as :

$$\Phi(\mathbf{a}, \mathbf{b}, s, t) = \mathbf{a}^T \Sigma_{xy} \mathbf{b} + s(\mathbf{a}^T \Sigma_{xx} \mathbf{a} - 1) + t(\mathbf{b}^T \Sigma_{yy} \mathbf{b} - 1) \tag{A1}$$

In order to maximize  $\Phi$ ,  $\Phi$  must content with following condition:

$$\begin{cases} \partial\Phi/\partial\mathbf{a} = \Sigma_{xy} \mathbf{b} + 2s\Sigma_{xx} \mathbf{a} = 0 \\ \partial\Phi/\partial\mathbf{b} = \Sigma_{yx} \mathbf{a} + 2t\Sigma_{yy} \mathbf{b} = 0 \end{cases} \tag{A2}$$

Then we can obtain:

$$2s\mathbf{a} = -\Sigma_{xx}^{-1} \Sigma_{xy} \mathbf{b} \tag{A3}$$

$$2t\mathbf{b} = -\Sigma_{yy}^{-1} \Sigma_{yx} \mathbf{a} \tag{A4}$$

Equation (A3) left multiplies  $\mathbf{a}^T \Sigma_{xx}$ , then  $2s = -\mathbf{a}^T \Sigma_{xy} \mathbf{b}$ . Equation (A4) left multiplies  $\mathbf{b}^T \Sigma_{yy}$ , then  $2t = -\mathbf{b}^T \Sigma_{yx} \mathbf{a}$ , so  $s = t$ .  $\mathbf{a}, \mathbf{b}$  can be written as:

$$\begin{cases} \mathbf{a} = \Sigma_{xx}^{-1} \Sigma_{xy} \Sigma_{yy}^{-1} \Sigma_{yx} \mathbf{a} / 4s^2 \\ \mathbf{b} = \Sigma_{yy}^{-1} \Sigma_{yx} \Sigma_{xx}^{-1} \Sigma_{xy} \mathbf{b} / 4s^2 \end{cases} \tag{A5}$$

By (A5), we can know that  $\mathbf{a}, \mathbf{b}$  are the eigenvectors of  $\Sigma_{xx}^{-1} \Sigma_{xy} \Sigma_{yy}^{-1} \Sigma_{yx}$ ,  $\Sigma_{yy}^{-1} \Sigma_{yx} \Sigma_{xx}^{-1} \Sigma_{xy}$  and they have the same eigenvalue.

# Multilayer Perceptrons Applied to Traffic Sign Recognition Tasks

R. Vicen-Bueno, R. Gil-Pita, M. Rosa-Zurera, M. Utrilla-Manso,  
and F. López-Ferreras\*

Departamento de Teoría de la Señal y Comunicaciones,  
Escuela Politécnica Superior, Universidad de Alcalá,  
Ctra. Madrid-Barcelona, km. 33.600, 28805, Alcalá de Henares - Madrid, Spain  
{raul.vicen, roberto.gil, manuel.rosa, manuel.utrilla,  
francisco.lopez}@uah.es

**Abstract.** The work presented in this paper suggests a Traffic Sign Recognition (TSR) system whose core is based on a Multilayer Perceptron (MLP). A pre-processing of the traffic sign image (blob) is applied before the core. This operation is made to reduce the redundancy contained in the blob, to reduce the computational cost of the core and to improve its performance. For comparison purposes, the performance of the a statistical method like the k-Nearest Neighbour (k-NN) is included. The number of hidden neurons of the MLP is studied to obtain the value that minimizes the total classification error rate. Once obtained the best network size, the results of the experiments with this parameter show that the MLP achieves a total error probability of 3.85%, which is almost the half of the best obtained with the k-NN.

## 1 Introduction

Systems dedicated to Traffic Sign Recognition (TSR) tasks usually have two specific stages. The first one is related to the detection of traffic signs in a video sequence or image. And the second one is related to the recognition of these detected signs, which is paid special attention in this work. The performance of these stages highly depends on lighting conditions of the scene and the state of the road sign due to deterioration or vandalism. Another problem to surpass is the rotation, translation or inclination of the sign. Its perfect position is perpendicular to the trajectory of the vehicle, however many times it is not like that. Problems related to the traffic sign size are of special interest too. Although traffic sign sizes are normalized, we can find signs of different ones. So, the recognition of a traffic sign in this environment is not easy.

The TSR problem has been studied many times in the literature. The works [1][2][3] solve this problem using the correlation between the traffic sign and the elements of a data base. This technique involves great computational cost.

---

\* This work is supported by Spanish MEC Project TEC2004/03511/TCM.



In other works [4], Matching Pursuit (MP) is used in two stages: training and testing. The training process finds a set of best MP filters for each road sign. The testing process projects the input unknown road sign to different MP filters to find the best match. This method also implies great computational cost, specially when the number of elements grows up. In a recent work [5], a Neural Network (NN) following the Adaptive Resonance Theory is used as classification technique. This work applies this technique to the whole image, where many traffic signs can exist. This involves that the NN complexity must be very high in order to recognize all the signs contained in the image.

The objective of this paper is the study of different classification techniques applied to the TSR problem. The methods studied in this paper are the k-Nearest Neighbour (k-NN) and a Neural Network (NN) based method using the Multilayer Perceptrons (MLPs). In order to reduce the MLP complexity, the classification is applied after the traffic signs have been extracted from the image.

## 2 System Overview

The TSR system used is presented here. Also, the traffic signs obtained with it and considered for the experiments are shown. The TSR stage is the most important block of the system. It is divided in two parts: traffic sign pre-processing stage and TSR core. Two different methods are proposed for its core. The first one is based on statistical methods (k-NN) and it is taken as a reference for comparison purposes with the second one (MLP).

The blocks that composes the TSR system are shown in Fig. 1. The *Video Camera* block takes a *video sequence*. The *Image Extraction* block makes the video sequence easier to read and it is the responsible to create *images*. The *Sign Detection and Extraction Stage* extracts all the traffic signs contained in each image and generates the small images called *blobs*, one per sign. Fig. 1 also shows an example of the way this block works. The *Form Recognition Stage* is the responsible to discern among the different forms: circular, square, triangular and others. Once the blob form is classified, the *TSR Stage* has the responsibility to recognize which is the exact type of signal.

## 3 Traffic Sign Pre-processing Stage

Each blob presented at the input of the TSR stage has the information of the red (R), green (G) and blue (B) colours. The blob dimension is 30x30 pixels for each component (R, G and B). So, the total size of each blow is 2700 pixels. Due to the dimensions of the blob, the purpose of this stage is to reduce the redundancy of information given to the TSR core in order to reduce the computational cost, which is an important parameter in real-time implementations.

Consider  $\mathbf{B}$  is the matrix that contains the three colour components of the blob. Also, consider  $\mathbf{B}'$  results from representing  $\mathbf{B}$  in a grey scale. This change from RGB to grey scale is calculated with (1). The values of  $b_{i,j}$  and  $b'_{i,j}$  are the

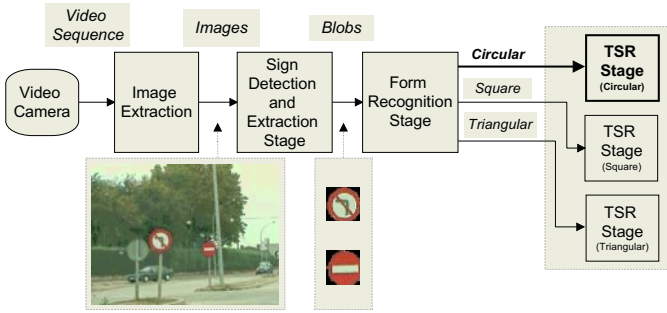


Fig. 1. Blocks of a TSR System

elements of the  $i$ -th row and  $j$ -th column of the matrix  $\mathbf{B}$  and  $\mathbf{B}'$ , respectively, where both indexes ( $i$  and  $j$ ) varies from 1 to 30.

$$b'_{i,j} = 0.49b_{i,j} + 0.29b_{i+30,j} + 0.22b_{i+60,j} \tag{1}$$

The normalized averages to the maximum pixel value ( $2^8$ ) of R ( $MR$ ), G ( $MG$ ) and B ( $MB$ ) are calculated with (2), (3) and (4), respectively. They are normalize between 0 to 1 because the application of this data to the TSR core.

$$MR = \frac{1}{256} \left( \frac{1}{900} \sum_{i=1}^{30} \sum_{j=1}^{30} b_{i,j} \right) \tag{2}$$

$$MG = \frac{1}{256} \left( \frac{1}{900} \sum_{i=31}^{60} \sum_{j=1}^{30} b_{i,j} \right) \tag{3}$$

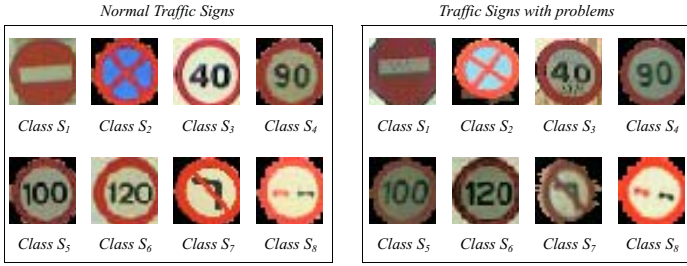
$$MB = \frac{1}{256} \left( \frac{1}{900} \sum_{i=61}^{90} \sum_{j=1}^{30} b_{i,j} \right) \tag{4}$$

The vertical ( $\mathbf{vh}$ ) and horizontal ( $\mathbf{hh}$ ) histograms are calculated with (5) and (6), respectively.

$$vh_i = \frac{1}{30} \sum_{j=1}^{30} (b'_{i,j} > T) \quad , \quad i = 1, 2, \dots, 30 \tag{5}$$

$$hh_j = \frac{1}{30} \sum_{i=1}^{30} (b'_{i,j} > T) \quad , \quad j = 1, 2, \dots, 30 \tag{6}$$

$T$  is the adaptive threshold calculated with (7) for each blob.  $vh_i$  is the  $i$ -th value of the  $\mathbf{vh}$  and corresponds to the ratio of values of column  $j$ -th that are greater than the threshold  $T$ .  $hh_j$  is the  $j$ -th value of the  $\mathbf{hh}$  and corresponds to the ratio of values of row  $i$ -th that are greater than the threshold  $T$ .



**Fig. 2.** Eight different classes of international traffic signs (normal and with problems)

$$T = \frac{1}{900} \sum_{i=1}^{30} \sum_{j=1}^{30} b'_{i,j} \tag{7}$$

This pre-processing provides an observation vector  $\mathbf{x}$  of 63 elements. It is composed of:  $\mathbf{vh}$  (30 elements),  $\mathbf{hh}$  (30 elements),  $MR$ ,  $MG$  and  $MB$ .

### 3.1 Traffic Sign Data Base Description

For the experiments presented in this paper eight different types of circular traffic signs were considered. These signs belong to the international traffic code. Fig. 2 (Normal Traffic Signs) shows the different classes of traffic signs considered in this work. These signs have been collected with the TSR system presented above. So, they present distortions due to the problems described in Sect. 1. Some examples are shown in Fig. 2 (Traffic Signs with problems). The problems caused by vandalism are shown in the examples of classes  $S_1$  and  $S_3$ . The problems related to the blob extraction in the *Sign Detection and Extraction Stage* (not a correct fit in the square image) are shown in the examples of classes  $S_1$ ,  $S_3$  and  $S_7$ . Examples of signs with problems of rotation, translation or inclination are those of classes  $S_3$ ,  $S_4$ ,  $S_5$ ,  $S_7$  and  $S_8$ . Finally, the differences of brightness are observed in both parts of Fig. 2.

The data base has been divided into three sets: train, validation and test. The first one is used as reference in the k-NN and to train the MLP. The second one is used during the training of the MLP to obtain good generalization results[6]. And the last one is used to evaluate the performance of the k-NN and the MLP.

The total number of traffic signs (blobs) considered for the experiment is 235. So, after pre-processing each blob, a total number of 235 observation vectors of 63 samples length each is obtained. The size of the train, validation and test sets are 79, 78 and 78 observation vectors (patterns), respectively.

## 4 TSR Core

TSR can be formulated as a multiple hypothesis test. The objective is to minimize a risk function that is given as the average cost  $\bar{C}$ , defined in (8) for  $L$  hypothesis.

$$\bar{C} = \sum_{i=1}^L \sum_{j=1}^L C_{i,j} P(D_i|S_j)P(S_j) \tag{8}$$

$P(D_i|S_j)$  is the probability of deciding in favor of  $S_i$  (decision  $D_i$ ) when the true hypothesis is  $S_j$ ,  $C_{i,j}$  is the cost associated with this decision and  $P(S_j)$  is the prior probability of hypothesis  $S_j$ .

The classifier performance can be specified with the *probability of correct classification* ( $P_{cc}$ ) and the *probability of misclassification* ( $P_{mc}$ ) for each hypothesis (class) or the *total correct rate* ( $P_c$ ) and the *total error rate* ( $P_e$ ) for all the hypotheses (classes). The  $P_{cc}$  is the probability of a pattern is correctly classified and the  $P_{mc}$  is the probability that a pattern is wrongly classified ( $P_{mc}=1-P_{cc}$ ). The  $P_c$  and  $P_e$  express the percentage of total classification successes and errors ( $P_e=1-P_c$ ) for all the hypothesis (classes), respectively.

### 4.1 Statistical Methods: k-NN

The k-NN approach is a widely-used statistical method. It is usually applied in other signal processing areas related to recognition. For example, it is used in *RADAR*, in the field of *Automatic Target Recognition (ATR)*[7].

This technique assumes that the data sets contain  $M_i$  points of class  $S_i$  and  $M$  points in total, so that  $\sum_i M_i = M$ . Then a hypersphere around the observation point  $\mathbf{x}$  is taken, which encompasses  $k$  points irrespective of their class label. Suppose this sphere, of volume  $V$ , contains  $k_i$  points of class  $S_i$ , then  $p(\mathbf{x}|S_i) \approx \frac{k_i}{M_i V}$  provides an approximation to this class-conditional density. The unconditional density can be estimated using  $p(\mathbf{x}) \approx \frac{k}{MV}$ , while the priors can be estimated using  $p(S_i) \approx \frac{M_i}{M}$ . Then applying Bayes' theorem, it's obtained (9)[8].

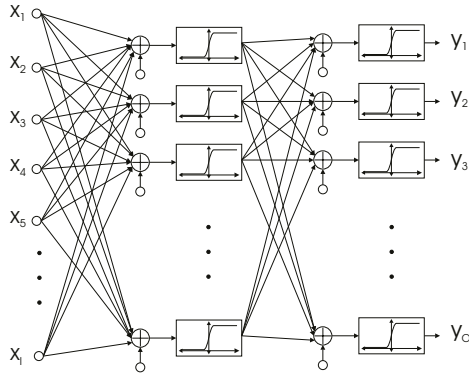
$$p(S_i|\mathbf{x}) = \frac{p(\mathbf{x}|S_i)p(S_i)}{p(\mathbf{x})} \approx \frac{k_i}{k} \tag{9}$$

Thus, to minimize the probability of misclassifying a vector  $\mathbf{x}$ , it should be assigned to the class  $S_i$  for which the ratio  $k_i/k$  is highest.

The way to apply this method consists in comparing each pattern of the test set with all of the train set and deciding which class  $S_i$  is the most appropriate.  $k$  indicates the number of patterns that take part in the final decision of classifying an observation vector in class  $S_i$ .

### 4.2 NNs: MLP

The Perceptron was developed by F. Rosenblatt [9] in the 1960s for optical character recognition. The Perceptron has multiple inputs fully connected to an output layer with multiple outputs. Each output  $\mathbf{y}_j$  is the result of applying the linear combination of the inputs to a non linear function called activation function. MLPs extend the Perceptron by cascading one or more extra layers of processing elements. These layers are called hidden layers, since their elements are not connected directly to the external world. Fig. 3 shows an MLP with  $I$



**Fig. 3.** Structure of a MLP

inputs (observation vector  $\mathbf{x}$ ), one hidden layer with  $N$  neurons and  $O$  outputs (classification vector  $\mathbf{y}$ ).

Cybenko’s theorem [10] states that any continuous function  $f : \mathbf{R}^n \rightarrow \mathbf{R}$  can be approximated with any degree of precision by log-sigmoidal functions. Therefore, it is selected a MLP with one hidden layer using the log-sigmoidal function given in (10) as the activation function, where  $z$  is its input.

$$L(z) = \frac{1}{1 + \exp(-z)} \tag{10}$$

The *Back-Propagation Algorithm* [6] with cross-validation is used to train the MLP. This algorithm tries to find the minimum of the error surface given by the *Mean Square Error* (MSE) criterion.

## 5 Results

The first TSR core proposed is based on the k-NN for comparison purposes. Table 1 shows the  $P_e$  of the test set for each  $k$ . The best classification results (minimum  $P_e$ ) are obtained for  $k=1,2,3$ . The parameter selected is  $k=1$  because it gives the least computational cost. Table 3 shows the  $P_{mc}$  for each class. As can be observed, the patterns that belongs to the classes  $S_1, S_4, S_5$  and  $S_8$  are correctly classified, whereas the patterns of class  $S_3$  are those that produces the greater  $P_{mc}$ . The patterns of classes  $S_2, S_6$  and  $S_7$  have the same  $P_{mc}$ .

The second TSR core proposed is based on a MLP. Its parameters are:  $I=63$  inputs (pattern length), one hidden layer with  $N$  neurons and  $O=8$  outputs (number of classes  $S_i$ ). After training different MLPs (63/ $N$ /8) with the train and validation sets, the  $P_e$  obtained with the test set for each  $N$  is shown in table 2. The best classification results (minimum  $P_e$ ) are obtained for a total number of 58 neurons in its hidden layer (63/58/8). For this network size, table 3 shows the  $P_{mc}$  for each class. As can be observed, the patterns related to the classes

**Table 1.**  $P_e(\%)$  versus  $k$  parameter for each k-NN experiment

| $k$   | 1    | 2    | 3    | 4    | 5     | 6    | 7     | 8     | 9     | 10    |
|-------|------|------|------|------|-------|------|-------|-------|-------|-------|
| $P_e$ | 7.69 | 7.69 | 7.69 | 8.97 | 10.26 | 8.97 | 11.54 | 14.10 | 12.82 | 14.10 |

**Table 2.**  $P_e(\%)$  versus N for each trained MLP (63/N/8)

| $N$   | 10    | 18   | 26   | 34   | 42   | 50   | 58   | 66   | 74   | 82   |
|-------|-------|------|------|------|------|------|------|------|------|------|
| $P_e$ | 15.38 | 7.69 | 8.97 | 7.69 | 5.13 | 7.69 | 3.85 | 8.97 | 7.69 | 7.69 |

**Table 3.**  $P_{mc}(\%)$  for k-NN( $k=1$ )|MLP(63/58/8) experiments

|          | $S_1$     | $S_2$     | $S_3$     | $S_4$     | $S_5$     | $S_6$     | $S_7$     | $S_8$     |
|----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
| $P_{mc}$ | 0.00 0.00 | 1.28 0.00 | 3.84 1.28 | 0.00 0.00 | 0.00 0.00 | 1.28 1.28 | 1.28 1.28 | 0.00 0.00 |

**Table 4.**  $P(D_i|S_j)(\%)$  for k-NN( $k=1$ )|MLP(63/58/8) experiments

|       | $S_1$       | $S_2$       | $S_3$       | $S_4$       | $S_5$       | $S_6$       | $S_7$       | $S_8$       |
|-------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| $D_1$ | 100.0 100.0 | 0.00 0.00   | 0.00 0.00   | 0.00 0.00   | 0.00 0.00   | 0.00 0.00   | 0.00 0.00   | 0.00 0.00   |
| $D_2$ | 0.00 0.00   | 92.31 100.0 | 0.00 0.00   | 0.00 0.00   | 0.00 0.00   | 0.00 0.00   | 12.50 12.50 | 0.00 0.00   |
| $D_3$ | 0.00 0.00   | 0.00 0.00   | 57.14 85.71 | 0.00 0.00   | 0.00 0.00   | 16.67 16.67 | 0.00 0.00   | 0.00 0.00   |
| $D_4$ | 0.00 0.00   | 7.69 0.00   | 14.29 0.00  | 100.0 100.0 | 0.00 0.00   | 0.00 0.00   | 0.00 0.00   | 0.00 0.00   |
| $D_5$ | 0.00 0.00   | 0.00 0.00   | 0.00 0.00   | 0.00 0.00   | 100.0 100.0 | 0.00 0.00   | 0.00 0.00   | 0.00 0.00   |
| $D_6$ | 0.00 0.00   | 0.00 0.00   | 28.57 14.29 | 0.00 0.00   | 0.00 0.00   | 83.33 83.33 | 0.00 0.00   | 0.00 0.00   |
| $D_7$ | 0.00 0.00   | 0.00 0.00   | 0.00 0.00   | 0.00 0.00   | 0.00 0.00   | 0.00 0.00   | 87.50 87.50 | 0.00 0.00   |
| $D_8$ | 0.00 0.00   | 0.00 0.00   | 0.00 0.00   | 0.00 0.00   | 0.00 0.00   | 0.00 0.00   | 0.00 0.00   | 100.0 100.0 |

$S_1, S_2, S_4, S_5$  and  $S_8$  are correctly classified. While the patterns of classes  $S_3, S_6$  and  $S_7$  are not correctly classified and have the same  $P_{mc}$ .

The  $P_e$  for the k-NN ( $k=1$ ) is 7.69% and for the MLP (63/58/8) is 3.85%. Although both methods have low error rates, the MLP is the better solution because it achieves an error rate reduction of 49,99%, i.e., the  $P_e$  is nearly reduced to the half. Table 4 shows the way the error is distributed for the classes in the terms of the  $(P(D_i|S_j))$  for each experiment. As can be observed, the classes  $S_1, S_4, S_5$  and  $S_8$  are correctly classified with both techniques. The patterns of class  $S_2$  are correctly classified with the MLP, while some patterns of this class are classified as  $S_4$  with the k-NN ( $k=1$ ). Also, the patterns of classes  $S_6$  and  $S_7$  are classified in the same way with both methods. There are some patterns that are classified as  $S_3$  when they are  $S_6$ , and some of them are recognized as  $S_2$  when they are  $S_7$ . Finally, the great difference between the performance of the methods relies on the classification of patterns of class  $S_3$ . The MLP is more robust against errors of classifying this class as  $S_4$  and  $S_6$ .

The MLP is robust against the error of classifying patterns of classes  $S_3$  as  $S_4$  and the k-NN is not. And its error of classifying patterns of  $S_3$  as  $S_6$  is slower than the k-NN one.

## 6 Conclusions

The classification results obtained in this work show the robustness of the TSR system against problems like lighting conditions, rotation and traffic sign sizes.

Both TSR strategies presented obtain low error rates. This is due to the redundancy minimization in the blobs achieved with the pre-processing stage. It's important to note that best results are obtained with the MLP, specially for the traffic signs that belongs to classes  $S_2$  and  $S_3$ . Moreover, it's observed that a relationship between classes  $S_3$  and  $S_6$  exists independently the TSR core used.

The  $P_e$  achieved with the MLP (63/58/8) is 3.85%, which implies an error rate reduction of 49.93% comparing with the k-NN ( $k=1$ ), i.e., the error rate is nearly reduced to the half. So, according to the results of the MLP, we can consider it a good solution to be implemented in a real-time TSR systems.

## References

1. Miura, J., et. al., "An active vision system for real-time traffic sign recognition", *Proc. IEEE Intelligent transportation systems*, 2000, pp. 52-57.
2. Pérez, E. and Javidi, B., "Nonlinear distortion-tolerant filters for detection of road signs in background noise", *IEEE trans. on Vehicular Technology*, 2002, vol. 51, no. 3, pp. 567-576.
3. Escalera, A. de la, et. al., "Visual sign information extraction and identification by deformable models for intelligent vehicles", *IEEE trans. on intelligent transportation systems*, 2004, vol. 5, no. 2, pp. 57-68.
4. Hsu, S. H. and Huang, C. L., "Road sign detection and recognition using matching pursuit method", *Image and Vision Computing*, 2001, vol. 19, pp. 119-129.
5. Escalera, A. de la, et. al., "Traffic sign recognition and analysis for intelligent vehicles", *Image and vision computing*, 2003, vol. 21, pp. 247-258.
6. S. Haykin, *Neural networks. A comprehensive foundation (second edition)*, Prentice-Hall Inc., London, 1999.
7. A. A. Kisienski, et al., "Low-frequency approach to target identification", *Proceedings of the IEEE*, 1975, vol. 63, pp. 1651-1659.
8. C.M. Bishop, *Neural networks for pattern recognition*, Oxford University Press Inc., 1995.
9. F. Rosenblatt, *Principles of Neurodynamics*, Spartan books, New York, 1962.
10. G. Cybenko, "Approximation by superpositions of a sigmoidal function", *Mathematics of Control, Signals and Systems*, 1989, vol. 2, pp. 303-314.

# Shape Classification Algorithm Using Support Vector Machines for Traffic Sign Recognition

P. Gil-Jiménez, S. Lafuente-Arroyo, S. Maldonado-Bascón,  
and H. Gómez-Moreno

Dpto. de Teoría de la señal y Comunicaciones, Universidad de Alcalá,  
28871 Alcalá de Henares (Madrid), Spain

{pedro.gil, sergio.lafuente, saturnino.maldonado, hilario.gomez}@uah.es

**Abstract.** In this paper, a new algorithm for traffic sign recognition is presented. It is based on a shape detection algorithm that classifies the shape of the content of a sign using the capabilities of a Support Vector Machine (SVM). Basically, the algorithm extracts the shape inside a traffic sign, computes the projection of this shape and classifies it into one of the shapes previously trained with the SVM. The most important advances of the algorithm is its robustness against image rotation and scaling due to camera projections, and its good performance over images with different levels of illumination. This work is part of a traffic sign detection and recognition system, and in this paper we will focus solely on the recognition step.

## 1 Introduction

A traffic sign recognition system basically consists on an image processing system mounted over a vehicle recording the road, and the goal of the system is the detection of all traffic signs present on the scene, and also its classification according with its shape, colors and meaning. Table 1 shows the meaning of traffic signs according with its color and shape. All the signs and properties described are for the Spanish traffic signs.

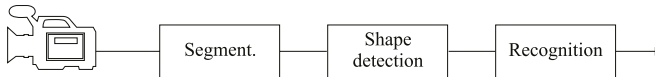
In most cases, traffic sign research is divided into two basic steps, namely detection and recognition. The detection step is the process that determines which parts of the images are candidate to be a traffic sign. In many works [1], [2], [3], [4], the detection is based on a color-based segmentation, taking advantage of the main colors used on traffic signs, as we can see in table 1. For that reason, red, blue, yellow and/or white are the most frequently colors used on the segmentation process.

Once candidate blobs have been extracted from the image, some approaches implement a pre classification step according with its shape [1], [5]. From table 1, the shapes used in traffic signs are the equilateral triangle, the circle, the octagon and the square. With this additional step, we reduce the classification problem to a smaller number of classes, therefore reducing the time employed in the classification stage.



**Table 1.** Meaning of traffic signs according to its color and shape

| Color          | Shape     | Meaning                           |
|----------------|-----------|-----------------------------------|
| Red Rim        | Circle    | Prohibition                       |
| Red Rim (Up)   | Triangle  | Danger                            |
| Red Rim (Down) | Triangle  | Yield                             |
| Red            | Octagonal | Stop                              |
| Blue           | Square    | Recommendation                    |
| Blue           | Circle    | Obligation                        |
| White          | Circle    | End of prohibition                |
| Yellow         | Circle    | End of prohibition (construction) |



**Fig. 1.** Block diagram

In figure 1 we can see the block diagram of the traffic sign recognition system we have described. The algorithm described in this paper is related with the last step in figure 1, that is, once every possible blob has been classified according with its shape and color, the last step consists on the recognition of the content of the sign, and so, the identification of the sign. A complete description for the rest of the system can be found in [5].

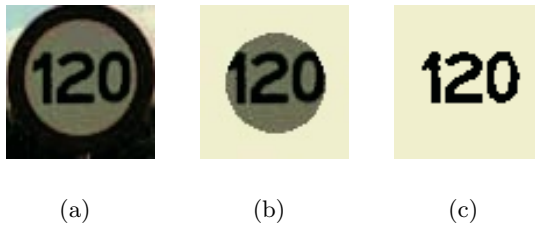
Many works have been proposed for the recognition step. In [4], a NN is used to perform the classification of every possible blob. In [3], the identification of signs is carried out by a normalized correlation-based pattern matching using a traffic sign image database. In this work the classification task is performed by a SVM [6] as it will be described later. We have focused our algorithm on circular signs, since on these kind on signs, object rotations appear as a problem than can not be solved as easily as with other kind of shapes, where reorientation of the blob can be done from the edges of the sign.

## 2 Shape Classification

Since the inside part of a typical traffic sign is normally composed of one or two colors, plus the background, a new segmentation process can be performed over the blob to separate the content of the sign from the background. After this process, the problem is reduced to classify the shape of the segmented object into one of all possible objects that can be found inside a traffic sign, that is, classify the sign according with its meaning.

### 2.1 Segmentation

Figure 2 shows an example of sign segmentation for circular images with white background. In 2(a) we can see the blob corresponding to the sign. This blob is



**Fig. 2.** Segmentation example. (a) Original blob, (b) Original blob without the red rim, (c) Segmented image

the output of the previous steps, as mentioned above. Since the previous steps has classified the sign into the circular group, we can approximately determine the center and the radius of the sign. With this two parameter, removing the red rim from the sign is straightforward. Obviously, this step must be different for every kind of sign, according with its shape and colors, but the procedure is the same for signs from the same category. In 2(b), the same sign is shown, where the red circular rim has already been removed. Once the rim is removed, the last process is the segmentation of the content of the sign. In figure 2(c) the segmented image is shown, where the segmentation has been performed over the luminance matrix, using an adaptive threshold. This process gives a certain robustness against illumination changes.

## 2.2 Object Projection

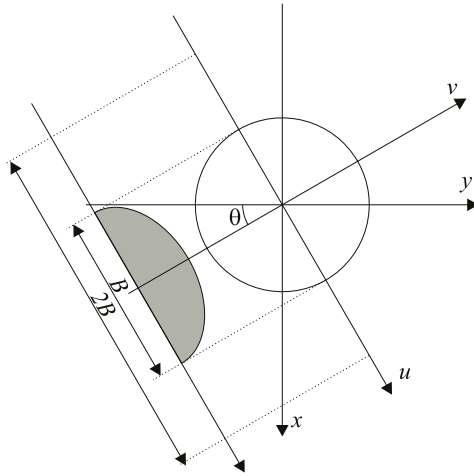
The classification of the segmented object now is performed computing the projections of the objects. These projections are computed at several values of  $\theta$ , according with figure 3. As we will see later, this procedure give a great robustness against object rotations. For each value of  $\theta$ , a new coordinates system, denoted  $uv$ , is generated, where the transformation matrix between this system and the image coordinates system  $(xy)$  is given by:

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix}$$

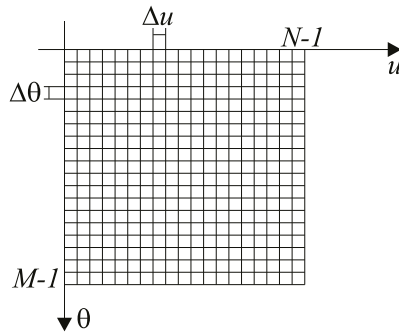
Since the projection of the object is computed for each value of  $\theta$ , the whole operation will yield a two dimension matrix of size  $M \times N$  (see figure 4), being  $M$  the number of values of  $\theta$  where the projections are computed, and  $N$  is the number of samples of each projection. For a particular value of  $M$ , the value of  $\theta$  for each iteration is incremented according with the following step:

$$\Delta\theta = \frac{2\pi}{M}$$

The projections are computed over the coordinate  $u$ , (see figure 3). To make the results independent of the object size, and hence, robust to image scaling,  $N$  is a constant in the system, and the step between samples in the coordinate  $u$  are



**Fig. 3.** *uv* and *xy* coordinates system



**Fig. 4.** Projection matrix

determined taking into account the size of the object. The maximum width of the object can be estimated from the second order moments in the angle of the least moment of inertia [7]. For a rectangle of height  $H$ , width  $B$ , and  $B > H$ , the second order moments are:

$$\mu_{20} = \frac{B^3 H}{12}$$

$$\mu_{02} = \frac{H^3 B}{12}$$

From these expressions, we can compute the width of the rectangle,  $B$ :

$$B = \sqrt[3]{\frac{12^2 \cdot \mu_{20}^3}{\mu_{02}}}$$

For an arbitrary shape, the previous expression allows us to estimate the maximum width of the object from its second order moments. In this case, the previously computed value of  $B$  is not its maximum width, but it leads us to determine the starting and ending values in the coordinate  $u$  where the projections are to be computed. These limits will be:

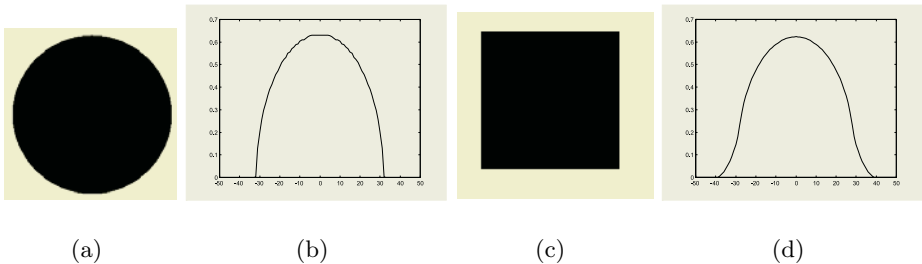
$$-k \frac{B}{2} < u < k \frac{B}{2}$$

where  $k$  is a constant chosen according with the following discussion. Large values of  $k$  implies that many samples of the projections will be null for values of  $u$  near  $-kB/2$  and  $kB/2$ , because we are evaluating beyond the object. With small values of  $k$ , samples of the projection can be loosen because we are not analyzing the whole object. The value actually chosen was  $k = 2$ , and so the projections are computed for  $-B < u < B$  and  $-B < v < B$  (see figure 3). With this value, the step between samples will be:

$$\Delta u = \frac{2 \cdot B}{N}$$

From now on, we will call this matrix the *projection matrix*. The next step consists on the computation of the mean value for every column in the projection matrix to obtain a vector of  $N$  elements. As with the previous matrix, we will call this vector the *projection vector*. With this step we make this vector invariant to object rotation, since a rotation on the object is reflected as a circular shift on the projection matrix in coordinate  $\theta$ , and since we compute the mean value for every value of  $\theta$ , the result is independent of the orientation of the object. In figure 5 it is shown the projection vector for a circle (5(a) and 5(b)) and the projection vector for a square (5(c) and 5(d)).

As it can be seen easily, the projection vector is symmetric, and hence, only half of the vector is used. This new projection vector, of size  $N/2$ , will be the vector used in the classification process. Figure 6 shows the projection vector for some different shapes used in the evaluation section, along with its corresponding image and its segmented mask.



**Fig. 5.** Projection vector examples



(a) Category A (Maximum speed limit 120)



(b) Category B (Maximum speed limit 40)



(c) Category C (Maximum speed limit 90)



(d) Category D (No overtake)



(e) Category E (Maximum speed limit 100)

**Fig. 6.** Projection vector examples for the test images

### 3 Experimental Results

A set of 154 blobs were chosen to test the proposed scheme. These blobs were extracted from an image database we have created for the evaluation of different traffic sign detection and recognition algorithms, and it is available at <http://roadanalysis.uah.es>. In this set we have included signs from 5 different types, as can be seen in figure 6. The algorithm basically extracts the projection vector as described above, taking  $N$  equal 100, that lead to a final vector of  $N/2 = 50$  samples. The value of  $N$  was chosen as a tradeoff between precision and speed of the algorithm. We also take  $M$  equal 90, leading to a step of 4 degrees in  $\theta$ . This value of  $M$  is again a tradeoff between precision and speed. Higher values of  $M$  implies more computational load, with no significant improvement in the results. Smaller values of  $M$  may reduce the robustness of the algorithm to object rotations.

In the experiment, an exhaustive search over the parameters  $\gamma$  and  $C$  was performed in order to find the values where the total number of errors in the experiment were minimum. The results for this search show that the optimal values lay near the values  $\gamma = 1$  and  $C = 100$ . Once the optimal values were found, from now on, all remaining experiments use these two particular values for parameters  $\gamma$  and  $C$ . We used 4 blobs of each category, which account for a total of 20 blobs, in the search step. For the rest of the set, one third of the blobs of each category was taken as the training set, leaving the rest of the blobs as the testing set. This experiment was repeated three times, taking as the training set for a particular experiment a different set of blobs than the ones used on the other experiments. The processes were performed with a SVM [6] with a Gaussian Kernel. The implementation uses the library LIBSVM [8].

Table 2 shows the results for the optimal values. This table shows, for each category, the success probability as the number of blobs properly classified with respect to the total number of blobs in the prediction set. Note that the experiment was repeated three times, and these values are the mean values for the three experiments. It also shows the number of blobs used for the training step and the number of blobs used in the prediction step.

From the results, we must conclude that the success probability is not good enough for categories with a small number of samples, especially for categories  $A$  and  $B$ , whereas for categories with enough number of samples are satisfactory, which makes the overall success probability acceptable. We also have to take into account that the images used in the experiment are realistic images, that

**Table 2.** Results for  $\gamma = 1$  and  $C = 100$

| Category          | A  | B  | C  | D  | E  | Total      |
|-------------------|----|----|----|----|----|------------|
| Number of blobs   | 18 | 18 | 29 | 46 | 23 | <b>134</b> |
| Training          | 6  | 6  | 9  | 15 | 7  | <b>43</b>  |
| Prediction        | 12 | 12 | 20 | 31 | 16 | <b>91</b>  |
| Success prob. (%) | 80 | 78 | 89 | 97 | 87 | <b>90</b>  |

includes different illuminations, occlusions, deteriorated signs and other kind of problems that make the success probability decrease.

## 4 Conclusions

This paper describes a new algorithm for the recognition of traffic signs. It is based on a shape detector that focuses on the content of the sign to perform the recognition of traffic signs. The classification is done by a SVM where the input is a vector computed from the projections of the object at several angles to overcome orientation problems. Other advantages of the algorithm is its robustness against different illumination and scaling, and also its simplicity.

The direction of our future work must include the increase of the test set database in two ways: first, increase the number of categories, and second, increase the number of blobs for each category. We also need to focus our work in the previous steps, especially the segmentation, since this step is crucial for the correct operation of the whole system. We also have to deal with other kind of problems, like partial occlusions, shadows, bad illuminations, etc.

## Acknowledgement

This work was supported by the project of the Ministerio de Educación y Ciencia de España number TEC2004/03511/TCM.

## References

1. de la Escalera, A., Moreno, L., Salichs, M.A., Armingol, J.: Road traffic sign detection and classification. *IEEE trans. on industrial electronics* **44** (1997) 848–859
2. Kamada, H., Naoi, S., Gotoh, T.: A compact navigation system using image processing and fuzzy control. *Southeastcon Proceedings* **1** (1990) 337 – 342
3. Miura, J., Kanda, T., Shirai, Y.: An active vision system for real-time traffic sign recognition. *Proc. IEEE Intelligent transportation systems* (2000) 52–57
4. de la Escalera, A., Armingol, J.M., Mata, M.: Traffic sign recognition and analysis for intelligent vehicles. *Image and vision computing* **21** (2003) 247–258
5. Lafuente-Arroyo, S., García-Díaz, P., Acevedo-Rodríguez, F.J., Gil-Jiménez, P., Maldonado-Bascón, S.: Traffic sign classification invariant to rotations using support vector machines. *ACIVS'04* (2004)
6. Vapnik, V.: *The Nature of Statistical Learning Theory*. Springer-Verlag, New York (2000)
7. Jain, A.K.: *Fundamentals of Digital Image Processing*. Prentice Hall (1989)
8. Chang, C.C., Lin, C.J.: *LIBSVM: a library for support vector machines*. (2001) Software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.

# A Practical License Plate Recognition System for Real-Time Environments

Cemil Oz<sup>1</sup> and Fikret Ercal<sup>2</sup>

<sup>1</sup>Department of Computer Engineering of Sakarya University, Turkey

<sup>2</sup>Department of Computer Science of University of Missouri-Rolla, USA  
{ozc,ercal}@umr.edu

**Abstract.** A computer vision system to recognize license plates of vehicles in real-time environments is presented in this study. The images of moving vehicles are taken with a digital camera and analyzed in real-time. An artificial neural network (ANN) system is used to locate the area and position of the license plate. The system has the following stages: (i) Image acquisition and determination of the location of the vehicle license plate (VLP), (ii) segmentation of the VLP into separate characters using image processing techniques, and (iii) recognition of each symbol in VLP using a feedforward artificial neural network (ANN) and assembly of the characters. Performance results are presented at the end.

## 1 Introduction

Automated recognition and identification of vehicle license plates (VLP) has great importance in security and traffic systems. It can help in many ways in monitoring and regulating the road traffic. For the management of urban and rural traffic, there is a lot interest in the automation of the license plate recognition in order to regulate the traffic flow, to control access to restricted areas, and to survey traffic violations.

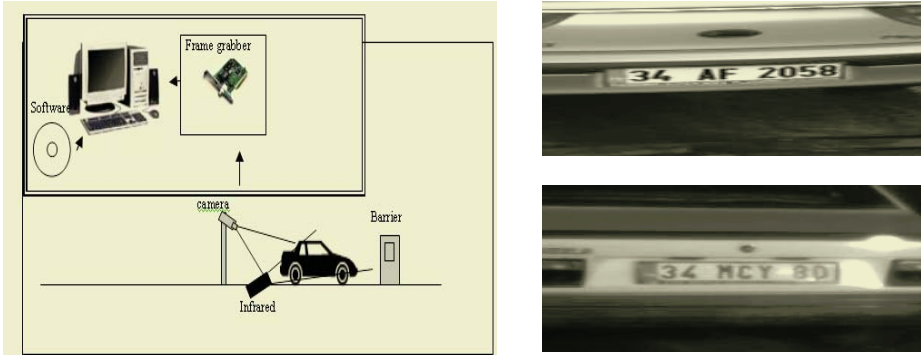
VLP identification is a difficult problem in computer vision and hence it is divided into several steps. The first step in the process is to locate the VLP. After that, characters in the VLP are recognized one by one using a recognition program such as a neural network. Character recognition is a popular and well-studied area in computer vision. In general, ANN and matching methods are used [1-5] for this purpose. In the literature, there are many studies which use different approaches for the VLP identification problem; ANN based [6-8], inductive learning method based [9], and techniques based on 2D coloration [10]. Some VLP identification studies focus only in the character recognition aspect of the problem [8, 9] while others [6, 7] also include the process of finding the location of VLP in the entire vehicle image. There is also some VLP recognition work for electronic tool collection and traffic management systems, and for the help of color blindness [10-13].

## 2 Licence-Plate Recognition System

An AVLP system consists of a camera, a frame grabber card, a computer, and software to process the images. Generally, various external conditions (such as



sunlight, shadows etc.) may affect the performance of the VLP system. Therefore, it is better to get the license plate illuminated with an infrared light source. A sample AVLP system is given Fig 1. Two sample images captured by the system in various external conditions are shown in Fig. 2.



**Fig. 1.** A sample A VLP system , the image on the left top is captured during daytime and the one on the right bottom is captured at night

### 3 Locating the License Plate in an Image

The VLP image has color and a complex background. We need to use the color properties and the composition of the complex background to locate the VLP in the entire image. A VLP consists of letters and numbers. Most studies in the literature use text detection and tracking for this purpose. Most commonly, these studies are either connected-components based (CC-based) [14, 15], or texture based [16, 17] or ANN based [18]. Here, we used a neural network as a classifier to identify the VLP pixels from the text areas. In this paper, VLP text pixels are classified as *text* and all other pixels are grouped as *non-text*. We used an ANN with three layers; input layer, hidden layer and the output layer.

The input layer receives the gray-level pixel values from the textual image. The output value of a hidden neuron is obtained as the dot product of the input vector and the weights assigned to the links connected to that neuron. The weights are adjusted during the training session through a back-propagation algorithm such as to minimize the sum of the squared error.

First, the frame images are filtered and then they are classified. Classification determines whether a pixel belongs to a text area or not. During the training session, a set of different patterns is used to adjust the weights of the network. Each pattern consists of the gray values of a pixel and its neighbors, along with the actual class of that pixel. Gray pixel values range from 0 to 256. We map these pixel values to 0 (text) or 1 (non-text). A pattern which consists of a 9x9 window is fed to the input layer and the output value indicates the class of the center pixel; text (0) or non-text (1). In the end, a classified image is obtained (see Figure 2). The last step in this process involves smoothing the classified image by removing the noise elements from

it. *Smoothing* is accomplished as follows. A pixel's class is determined as non-VLP text if the number of pixels, which are classified as text, in the 3x3 neighborhood is less than a threshold. Otherwise, the pixel's class is determined as text. The smoothed image is then projected along the y-axis and its histogram is computed. VLP zone is identified as a consecutive vertical zone in which the histogram values are above a certain threshold.

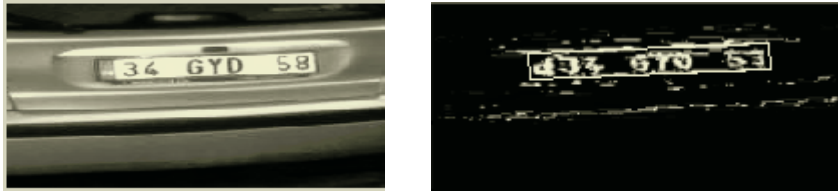


Fig. 2. Gary level image and classified image

## 4 Plate Segmentation

**Vehicle License Plate Properties.** A standard (recreational) Turkish VLP consists of three symbols; county code, a letter code, and a numeric code. Since there are a total of 85 counties in the country, county code starts at 01 and goes up to 85 in decimal. The second part in the VLP is a three-letter alpha numeric symbol using only capital letters A-Z. The third region has a maximum 4-digit decimal number as shown in Figure 3 (a). License plates for military and government vehicles use a different style. This study considers only civilian VLPs. The following steps are used in order to locate and segment the license plate image into individual characters.

**First Step.** VLP which is located by separating it from a complex background image is converted into a binary image with background taking the pixel value of 0. Horizontal projection is used for border clearing. For each row in the image, the number of zeros (0) are counted and the resulting histogram is plotted sideways as shown in Figure 3.

**Second Step.** After the VLPs' background is cleaned up, the individual characters are separated from each other using vertical projection (histogram). Basically, a zero value (0) in the vertical histogram indicates a border crossing between two characters. To save processing time in the next stage, we further eliminate the blank space between characters by drawing the character boundaries as close to the characters as possible.

**Third Step.** Normalization, in the original image, characters have different pixel sizes. It is important to normalize their size for further processing. In this step, every segmented character is mapped to a (19x11) pixel format. This makes it easier to design an ANN for the character recognition in the next stage.



Fig. 3. A Turkish VLP on the right, horizontal projection in the middle and VLP character segmentation on the left

### 5 Artificial Neural Network Model

A backpropagation algorithm is used for training the ANN model. The basic structure and formulation of backpropagation is summarized here. Training a neural network involves computing weights so as to get an output response to the input within an error limit. The input and target vectors make up a training pair. The backpropagation algorithm includes the following steps [19]:

1. Select the first training pair and apply the input vector to the net.
2. Calculate the net output.
3. Compare the actual output with the corresponding target and find the error.
4. Modify the weights so as to reduce the error.

These steps are repeated until the error is within the accepted limits. In the step 2, output sets for test inputs are calculated. If they are the same within an error range as the expected sets, then it is considered as the net has learnt the problem, and the final weights are stored so that they can be reused when needed. The developed ANN has a multi-layer feedforward structure as shown in Fig. 4. The variable definitions are given as follows [20,21]: **L=0**: input layer, **L=1**: hidden layer, **L=2**: output layer, **W<sub>1,ji</sub>**: weight matrix between the input layer and the hidden layer. **W<sub>2,tj</sub>**: weight matrix between the hidden and the output layer, **B<sub>1,j</sub>**: bias values of hidden neurons, **B<sub>2,t</sub>**: bias values of output neurons.

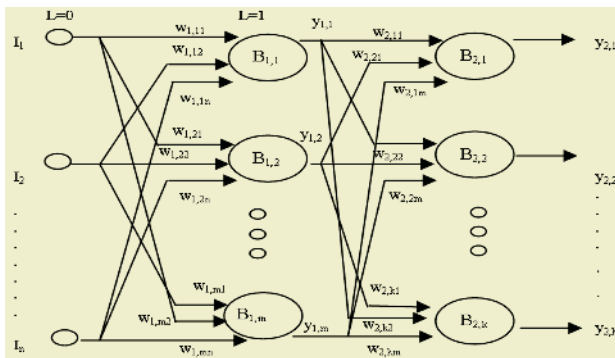


Fig. 4. A Multi-layer feedforward net structure

Equation (1) gives the output of the hidden layer.

$$y_{NET\ 1,j} = \sum_{i=1}^n w_{1,ji} I_i + B_{1,j} \tag{1}$$

$$y_{1,j} = f_j [y_{NET\ 1,j}] \quad j = 1,2,\dots, m$$

Equation (2) gives the output of the output layer.

$$y_{NET\ 2,t} = \sum_{j=1}^m w_{2,tj} y_{1,j} + B_{2,t} \tag{2}$$

$$y_{2,t} = f_t [y_{NET\ 2,t}] \quad t = 1,2,\dots, k$$

Activation function;

$$f(y_{NET}) = \frac{1}{1 - e^{-y_{NET}}} \tag{3}$$

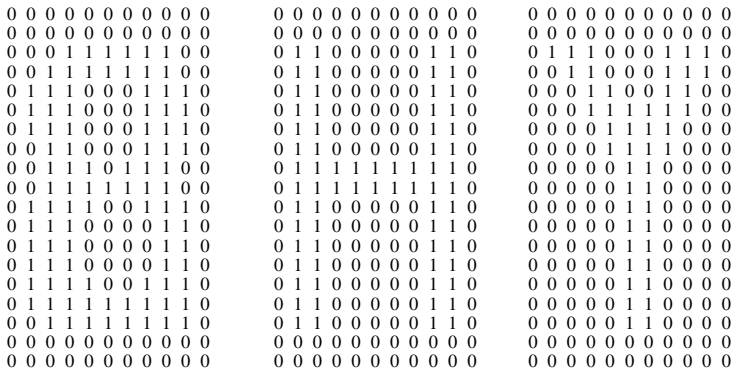
## 6 ANN Model Design for Character Recognition

Some characters used in the training set for the ANN are given in Table 1.

**Table 1.** Some Characters used in the training set

|   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|
| A | B | C | D | E | F | G | H | I |
| J | K | L | M | N | O | P | R | S |
| T | U | V | Y | Z | 0 | 1 | 2 | 3 |
| 4 | 5 | 6 | 7 | 8 | 9 |   |   |   |

The proposed ANN, a multilayer feedforward network, consists of 209 input variables, 104 hidden neurons, and 33 output variables and it is designed to recognize one single character at a time. Backpropagation algorithm is used for training. The training set is composed of two files; input and output. Input file contains image data belonging to 33 characters and each character has 209 image segments which are provided as inputs to the ANN. Fig. 5 shows three samples of segmented characters. The output file consists of 33 outputs each representing a single letter.



**Fig. 5.** Three sample of segmented characters

## 7 Character Recognition Using an ANN Model

ANN recognizes one character at a time. It recognizes the characters stored in the test file one at a time in sequential order starting from the first character until it reaches the last character. A special control mechanism is used for generating the outputs. As explained earlier, Turkish VLP has three regions containing, from left to right, the county code, a letter code, and a numeric code as shown in Figure 3. County code starts at 01 and goes up to 85 in decimal. Therefore, first two characters must be numeric. The second region in the VLP is a three-letter alpha numeric symbol using only capital letters A-Z. The third region has a 4-digit decimal number. This knowledge of the VLP composition helps in the recognition process. When recognizing the characters in the first and third plate regions, we only need to use the outputs  $O_{27}, \dots, O_{36}$ . Each time, we pick the one with the maximum value. Similarly, outputs  $O_1, \dots, O_{26}$  are used for recognizing the characters in the second plate region. The output of ANN is streamed into a text file.

## 8 Test Results

The Character recognition system was first trained with data from three samples. When three samples were not sufficient to generate satisfactory results, the number is gradually increased; first to six, and then nine, and finally 12 samples. At the testing stage, the system was tested starting with A and ending with Z, in order. The testing results are given in Table 2.

**Table 2.** Test results for individual character recognition (X unrecognized, + recognized)

| Samples |   |   |   |   | Samples |   |   |   |   | Samples |   |   |   |  |
|---------|---|---|---|---|---------|---|---|---|---|---------|---|---|---|--|
| 3       | 6 | 9 | 0 | 0 | 3       | 6 | 9 | 0 | 0 | 3       | 6 | 0 | 0 |  |
| X       | + | + | + | L | +       | + | + | + | Z | X       | + |   | + |  |
| +       | + | + | + | M | X       | + | + | + | 0 | +       | + |   | + |  |
| +       | + | + | + | N | +       | + | + | + | 0 | +       | + |   | + |  |
| X       | + | + | + | O | +       | + | + | + | 0 | +       | + |   | + |  |
| +       | X | + | + | P | X       | + | + | + | 3 | X       | + |   | + |  |
| +       | + | + | + | R | +       | + | + | + | 4 | X       | X |   | + |  |
| +       | + | X | + | S | X       | + | + | + | 5 | +       | + |   | + |  |
| +       | + | + | + | T | +       | + | + | + | 6 | +       | + |   | + |  |
| +       | + | + | + | U | +       | + | + | + | 7 | +       | + |   | + |  |
| +       | X | + | + | V | X       | X | + | + | 8 | X       | + |   | + |  |
| +       | + | + | + | Y | +       | + | + | + | 9 | +       | + |   | + |  |

We tested our VLP recognition technique using images taken from 40 vehicles at random during various periods of the day; around noon on a sunny day (15 images), on a rainy day (15 images), during sunset or sun rise, and at night time (10 images). The images had the following properties: they were taken with a simple HP Digital camera and they were shot from a fixed distance. Images were taken with a shooting angle ranging between  $0^{\circ}$  -  $15^{\circ}$ . Vehicles were either stationary or moving at a certain speed between 0-30 km/h.

The testing results were given in Table 3. Our program recognized 38 out of 40 license plates. This result translates into a 95% recognition rate which is considered to

be fairly high. The system failed to recognize only two VLPs. The reason for this failure is mainly due to the poor results obtained during the image processing phase. Basically, the quality of original image was so poor that the segmentation system failed to separate the characters correctly. And hence, ANN system could not recognize them. If the image quality is poor, some characters in VLP appear to be the same. The pairs (B, E), (F, P), and (P, R) are examples of such characters. There are other characters with similar properties too. If the image quality is not good and these characters are presented to ANN, they may be confused for one another. In only one case, the incorrect result was due to the failure of the ANN system. We have also applied our method to Missouri-USA license plates. We have shown that our approach can be used with minor changes for the recognition of Missouri license plates.

**Table 3.** Test results for license plate recognition

| Image type             | How many | Recognized | Un recognized |
|------------------------|----------|------------|---------------|
| Sunny day              | 05       | 04         | 0             |
| Rainy or close to dark | 05       | 05         | -             |
| Night time             | 00       | 9          | 0             |

## 9 Conclusions

A computer vision system to recognize license plates of vehicles is presented and tested in a real-time environment. The system used the following steps: (i) real-time image acquisition, (ii) determining the location and position of the vehicle license plate (VLP) using an artificial neural network (ANN), (iii) segmentation of the VLP into separate characters using image processing techniques, and (iv) recognition of each symbol in VLP using a feedforward artificial neural network (ANN) and assembly of the characters.

The system is tested using car images taken with a simple digital camera from random vehicles during various periods of the day and under different weather conditions. Vehicles were either stationary or moving at a certain speed (between 0-30 km/h). The system recognized 38 out of 40 license plates with 95% accuracy. The reason for the two failures was mainly due to the poor quality of images obtained initially. When the image quality is poor, some characters in VLP appear to be the same (such as the pairs (B, E), (F, P), and (P, R)). The recognition system is also tested on Missouri-USA license plates and satisfactory results are obtained.

In conclusion, we can state that the VLP recognition system can be safely used in real-time environments provided that good quality images (with good focus and resolution) are taken from moving vehicles.

## References

1. Pal, U., Chaudhuri, B.B.: Machine-printed and hand-written text lines identification. In: Pattern recognition letters, Elsevier, (2001) 431-441
2. Blumenstein, M, Verna, B.: Neural based segmentation and recognition technique for handwritten words. In: Proc. Of World Congress on Comp. Intell., Alaska, USA (1998) 1738-1742

3. Zheng, B., Qian W., Clarke, L.: Multistage neural network for pattern recognition in mammogram screening. In: Proc. of 1994 IEEE Int'l Conf. on Neural Networks (ICNN), Orlando, FL, USA (1994) 3437-3448.
4. Han, K, Sethi, I. K.: Handwritten signature retrieval and identification. In: Pattern Recognition Letters 14, (1993) 305-315
5. Lee, S. W.: Off-line recognition of totally unconstrained handwritten numerals using multilayer cluster neural network. In: IEEE Trans. on Patt. Anal. and Mach. Intell. 18, (1996) 648-650.
6. Draghici, S.: A neural network based artificial vision system for license plate recognition. In: Int'l J. of Neural Systems 8 (1997) 113-126
7. Wei, W., Wang, M., Huang, Z.: An automatic method of location for number-plate using color features. In: Proc. of 2001 IEEE Int'l Conf. on Image proc. (ICIP). Volume 1. (2001) 780-785.
8. Lee, C.: SIMNET: A neural network architecture for pattern recognition and data mining. PhD thesis, University of Missouri-Rolla (2003)
9. Aksoy M. S, Cagil G, Turker A. K.: Number plate recognition using inductive learning. Robotics and Autonomous Systems 33, (2000) 149-153.
10. Oz, C, Koker, R.: Vehicle License plate recognition using artificial neural Networks. In: Proc. of Second Int'l Conf. on Electrical and Electronics Engineering, Bursa, Turkey. (2001) 378-380
11. Grattoni, P, Pettiti, G, Rastello, M. L.: Experimental set-up for the characterization of automated number-plate recognizers. Measurements 26. (1999) 003-004
12. Zarrillo, M. L, Radwan, A. E, Aldeek, H. M.: Modeling traffic operations at electronic tool collection and traffic management systems. Computers and Industrial Engineering 33, (1997) 857-860
13. Kou, Y., Hsu, J.: Color-blindness plate recognition using a neuro-fuzzy approach. Engineering Applications of Artificial Intelligence 11, (1998) 530-547
14. Jain, A. K, Yu, B.: Automatic text location in images and video frames. In: Proc. of ICPR. (1998) 1499-1498
15. Zhou, J., Lopresti, D.: Extracting text from www images. In: Proc. of Fourth ICDAR, (1997) 048-050
16. Wu, V., Manmatha, R., Riseman, E. N.: Finding text in images. In: Proc. of 2<sup>nd</sup> ACM Int'l. Conf. on Digital Libraries, Images and Multimedia, (1997) 3-12
17. Jain, A. K, Bhattacharjee, S.: Text segmentation using gabor filters for automatic document processing. Machine Vision Applications 5 (1992) 069-084
18. Li, H., Doermann, D., Kia, O.: Automatic text detection and tracking in digital video. IEEE Transactions on Image Processing 9 (2000) 047-056
19. Zarrillo, M. L., Radwan, A. E., Aldeek, H. M.: Modeling traffic operations at electronic tool collection and traffic management systems. Computers Industrial Engineering 33, (1997) 857-860.
20. Abulafya, N.: Neural networks for system identification and control. MSc Thesis. University of London (1995)
21. Narendra, K.S.: Adaptive control using neural networks. Neural Networks for Control, MIT (1990)

# Matching Algorithm for Hangul Recognition Based on PDA

Hyeong-Gyun Kim<sup>1</sup>, Yong-Ho Kim<sup>2</sup>, and Jong-Geun Jeong<sup>3</sup>

<sup>1</sup> Dept of Computer & Internet, Dongkang College, South Korea

<sup>2</sup> Dept of Internet Information, Chosun College of Science & Technology, South Korea  
comkim@netian.com

<sup>3</sup> Dept of Computer Engineering, Honam University, South Korea  
jkjeong@honam.ac.kr

**Abstract.** Electronic Ink is a stored data in the form of the handwritten text or the script without converting it into ASCII by handwritten recognition on the pen-based computers and Personal Digital Assistants(PDA) for supporting natural and convenient data input. One of the most important issue is to search the electronic ink in order to use it. We proposed and implemented a script matching algorithm for the electronic ink. Proposed matching algorithm separated the input stroke into a set of primitive stroke using the curvature of the stroke curve. After determining the type of separated strokes, it produced a stroke feature vector. And then it calculated the distance between the stroke feature vector of input strokes and one of strokes in the database using the dynamic programming technique.

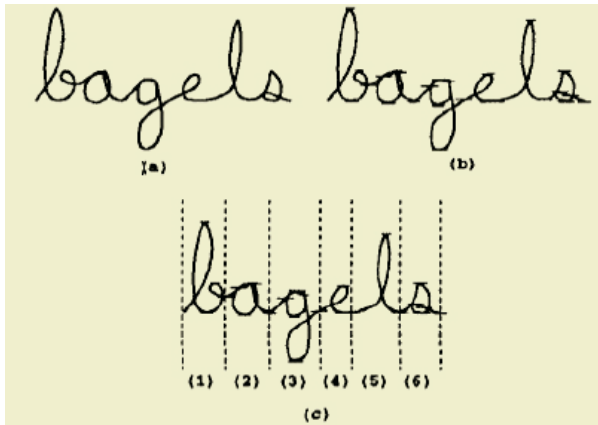
## 1 Introduction

There have been various research reports on the data processing and recognition for the electronic ink of PDA since the early 1990s. As the effective way of searching electronic ink data from the large data base, several approximate ink matching algorithms for the Hangul recognition of electronic ink data have been proposed[3][4], and methods utilizing the blind Markov Network and R-tree rate have been proposed[5][6]. However, these methods were developed with the main input type of English script. As compared writing English and Hangul scripts with pen, the English scripts can be decomposed into the continuous circular arcs, while the Hangul has the clear cornering position and direction of strokes. The proposed algorithm for matching the English scripts had been implemented[4] and applied for the Hangul scripts, showing very low matching rates. Consequently there is no meaning in directly applying the matching algorithm for the English scripts to the Hangul scripts, so that it is required to apply a Hangul matching algorithm for the electronic ink data mainly with the Hangul scripts, considering the geometric features of Hangul. This study proposes the Hangul matching system for storage and recognition in the form of electronic ink using the script data input with pen in the Hangul PDA.

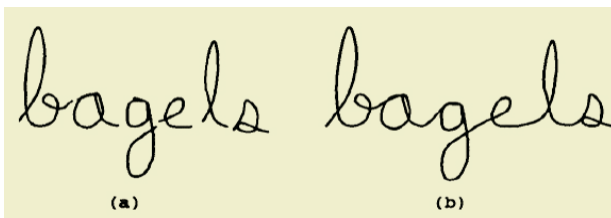


## 2 Related Work

In Figure 1, we show a pictogram and its segmentation into pen strokes and handwritten symbols. We can choose to represent pictograms by any of the three granules presented in the figure, i.e., as one entity containing the entire pictogram (Figure 1a), as a sequence of pen strokes (Figure 1b), or as a sequence of alphabet symbols (Figure 1). Of course, in order to select, for instance, the symbols as granules, we have to have a segmentation algorithm that properly separates the symbols. For instance, for strokes, a simple segmentation algorithm picks local minimum (or maximum) points and uses them to segment the curve.



**Fig. 1.** Example illustrating the segmentation of the pictogram in (a) into (b) strokes, (c) alphabet symbols



**Fig. 2.** An example of (a) a hand printed word, cursive handwritten word

Segmentation could be a difficult task for some types of pictograms, such as cursive handwritten words (see Figure 2b), or a simpler task as in hand printed words (see Figure 2a). Some languages, like Japanese, lend themselves easily to symbol segmentation. In Japanese, Kanji symbols are already separated by blank spaces. The choice of granularity has an impact on the type of indices we build.



Fig. 3. A list of pictograms, all representing concept “gold”

The second issue is that, for matching purposes, it is better to talk about pictogram (symbols, strokes) classes instead of individual pictograms. A pictogram class is the set of pictograms that have the same semantics, according to the user. Figure 3 shows a (non-exhaustive) list of pictograms that represent the concept “gold.” Of course, it would be impractical to store a pictogram class by storing the list of pictograms that belong to it. Alternatively, we have to choose a representative of the class and a distance metric.

Common features are: direction, velocity, change in direction, change in velocity, accumulative angle (with respect to the initial point), accumulative length and angle of bounding box diagonal (also with respect to the initial point), and accumulative sequence length.

The input format of our pictogram is an array  $s$  of  $P$  time-stamped sample points:

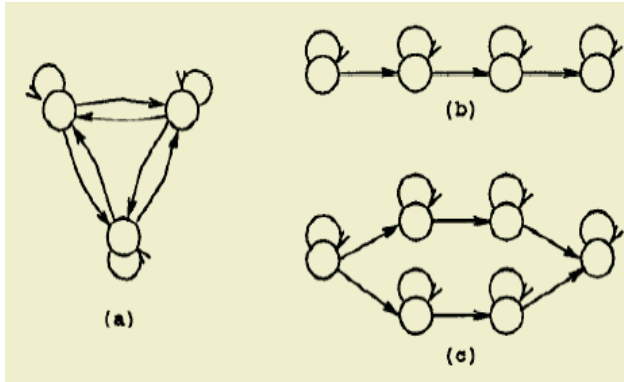
$$S_p = (x_p, y_p, t_p), 0 \leq p < P \quad (1)$$

After computing the local features, the pictogram can be represented by a sequence of feature vectors  $(v_1, t_1), (v_2, t_2), \dots$ . The dimensionality of the  $v_i$  corresponds to the number of local features at each point, Global Features: Global features are characteristics of the entire pictogram. Among them are: the bounding box coordinates, the total angle traversed by the pictogram (measured by the angle from the beginning point to the endpoint), and length of the bounding box diagonal. After computing global features, the pictogram can be represented by a vector of global feature values. We show here how to build two representative models. The choice of model also has an impact on the index technique that we use.

## 2.1 Hidden Markov Models

HMMs are already used in the field of speech and handwritten recognition as a powerful tool for speech and handwritten document matching.[2] Each pictogram in the database can be modeled by an HMM. The HMM is constructed so that it accepts the specific pictogram with high probability (relative to the other pictograms in the

database). In order to recognize a given input pictogram, we execute each HMM in the database and select the one that generates the input sequence with the highest probability. Since each HMM in the underlying sequence database has to be tested, this results in a linear process where the speed of execution is the primary difficulty. An HMM is a doubly stochastic process, where there is a probability distribution that governs the transitions between states and an output probability distribution that identifies the distribution of output symbols for each state.



**Fig. 4.** Several examples of HMMs: (a) the ergodic model with three states, (b) a left-to-right model with four states and (c) a parallel path left-to-right model with six states

We use one type of HMM structures, termed left-to-right HMM[4] (e.g., see Figure 4b). The left-to-right type of HMMs is useful for modeling temporal signals as in sound and cursive handwritten text because the underlying state sequence associated with the model has the property that, as time increases, the state index increases (or stays the same) — that is, the system states proceed from left to right. A left-to-right HMM can be constructed to model a handwritten word or an alphabet symbol. The HMM is constructed so that it accepts the word (or the symbol) with high probability (relative to the other words in the database). The probability  $y$  given by the HMM is the distance metric used for ranking purposes. Given an HMM that models a word (or symbol), we can run an input symbol against it and obtain as an output a matching probability. Given a set of stored words (or symbols), one can match an input word (or symbol) by running each one of the corresponding HMMs against the input and choosing those with the best matching probability. In fact, we can keep the size of the answer set as a parameter and choose the  $k$  best matches.

### 3 Proposed Hangul Matching Algorithm

In this study the Hangul matching algorithm was used to implement a system so that the prime script of Hangul is used in the form of the electronic ink data in the PDA. First divide the ink data into basic stroke units by the Hangul matching algorithm and then apply the dynamic programming technique. While considering hardware limitation of PDA, it is designed to store the Hangul

numeric data on the CF memory to enhance the recognition rate as well as to secure the effective speed. When converting Hangul data value by various electronic ink data on the PDA, the applied system matches the converted one with Hangul with the existing Hangul data.

### 3.1 Preprocessing

Shaking hand may cause mistakes on curves and strokes when writing. Mistakes are compensated through smoothing, filtering, removing mistakes and normalizing sizes.

$$X_i = \frac{(X_{i-3} + 3X_{i-2} + 6X_{i-1} + 7X_i + 6X_{i+1} + 3X_{i+2} + X_{i+3})}{27} \tag{2}$$

The current value is displaced by an average with the weight around 7 points, and the stroke curves are smoothed as shown in Eq. (2). The irregular spacing of points due to writing speeds are finalized uniform through DDA(Dot Density Algorithm) filtering.

### 3.2 Feature Extraction Process

The input strokes would show various forms of scribbles along with irregular lengths and angles in view of writing styles. So the feature points are utilized as the basic recognition unit to reduce unessential volume of information among the coordinate data column of input strokes[2][3]. However, several feature points may exist in specific parts of input strokes according to the writing speeds and angle variations, so that the process of removing rings, decorative lines, and neighboring feature points removes unnecessary feature points congregated within the given distance of marginal values. The feature and virtual vectors are information to match with the Hangul database, so that the feature vectors are procedural and directional information for the input strokes and the virtual vectors are information for position relation between strokes in the letter element or between letter elements[2][3]. In addition, the position relation and inclusive relation of position information as position information between strokes are extracted for letter element separation and recognition. The inclusive relation is the indication of adjacent level among the minimum circumscribed squares of strokes neighboring the minimum circumscribed square including the current strokes, composing inclusion, overlap, and separation.

Fig.5(a) shows a case of extracting inclusive relations for the input letter ‘강’. Dotted squares are the minimum circumscribed squares including strokes, and a number is the stroke number inside the dotted square, which means the dotted square 1 indicates the whole size for the stroke ‘ㄱ’. It is indicated that the dotted square 2 is included in the dotted square 1, the dotted squares 1,3, & 5 are separated each other, and the dotted squares 3 & 4 are overlapped. The position relation is a centroid of the current stroke and the direction information among the centroids of all input strokes, and is utilized as position relation information among the letter element, inside and

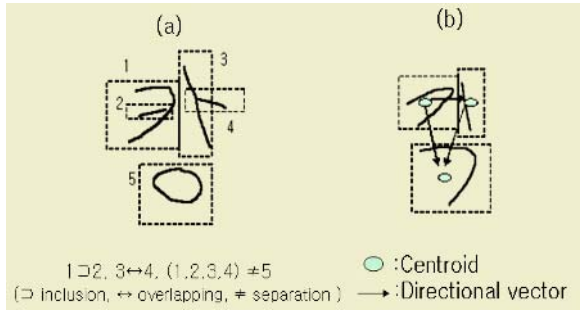


Fig. 5. Extraction of position relation among strokes

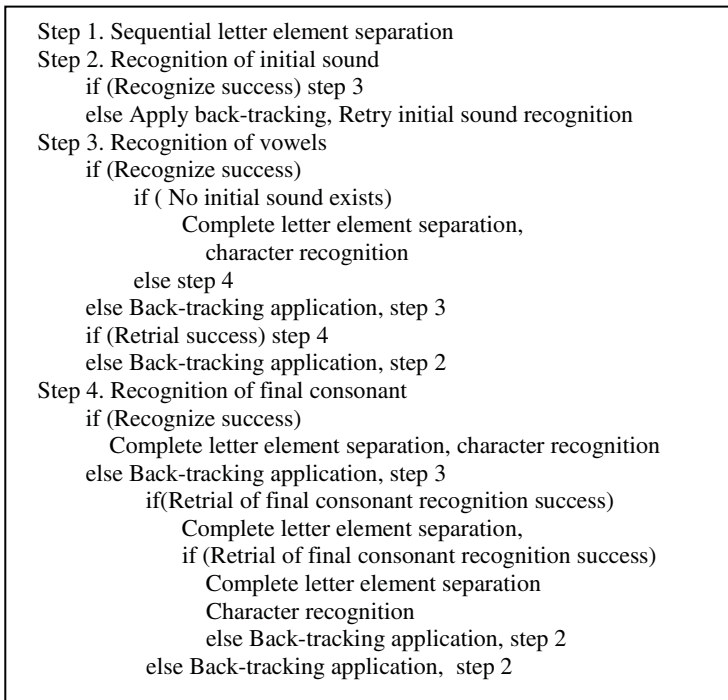
letter space. Fig.5(b) is an extraction case of input letter ‘ㄱ’, and shows the direction information on centroids among each letter elements comprising of letter. The position relation among strokes provides the position to decompose into the basic strokes from calculating the curvature for the points comprising strokes when considering input strokes as one curve. When indicating curvatures over the marginal value for the arbitrary points comprising strokes, the stroke is separated at that point.

### 3.3 Determination of Basic Stroke Units

Separation of letter element is prerequisite process for letter element recognition, and separation and recognition are in parallel processing in this study. The letter element recognition recognizes the letter element by matching stroke information extracted from the letter element separation and stroke information stored in the Hangeul database and converts information of each recognized letter element into the assigned value of the electronic ink data to generate a letter.

There are sequential letter element separation using position relation among strokes and back-tracking letter element separation performing recognition by changing the number of strokes comprising each letter element through matching Hangeul database. The sequential letter element separation recognizes a case of separate inclusive relation among strokes as the stroke from the other source, and case of overlapped or inclusive as the strokes from the same letter element. But in case of separate inclusive relation among strokes inside letter element, the sequential letter element separation shows insufficient letter separation capability, so that the back-tracking letter element separation is performed in case of false recognition or non-recognition. The existing back-tracking method shows the demerits of reprocessing all procedures. But the back-tracking letter element separation, based on stroke information extracted, separates the total strokes of input letter into the optimal number of strokes to recognize according to the recognition by letter element. There are sequential letter element separation using position relation among strokes and back-tracking letter element separation performing recognition by changing the

number of strokes comprising each letter element through matching Hangul database.



**Fig. 6.** Back-tracking letter element separation algorithm

The existing back-tracking method shows the demerits of reprocessing all procedures. But the back-tracking letter element separation, based on stroke information extracted, separates the total strokes of input letter into the optimal number of strokes to recognize according to the recognition by letter element. Fig.6 and Fig.7 shows an algorithm for the back-tracking letter element separation and a parallel processing for letter element separation and recognition, respectively. This study shows fast processing speed by performing the letter element separation differently according to configuration information of input letter and stroke types.

### 3.4 Similarity Calculation

The process separates the input data into the basic stroke unit by stroke curvatures. Seven basic strokes have been assigned to ensure fast matching speed in the PDA in consideration of the geometric characteristics of Hangul as shown in Fig.7. The next step decides types of basic strokes separated by the curvatures to create stroke feature vectors. The stroke feature vector has basic stroke unit information separated from the input data. Once the stroke feature

vector created, the dynamic programming calculates distances among the stroke feature vectors for the stroke feature vector of query and the ink data in the database, and compares the data having the shortest distance with the data value on the CF memory to return matching result.

### 3.5 Report on Matching Result

The letter element recognition comprises the synchronization process of values on the Hangul database on the CF memory, containing structural configuration information of Hangul and diverse stroke information of various users for each letter element. The structural configuration information of Hangul includes packing information among letter element using in recognizing the vowels, collection information, position relation among strokes, letter element classification information by configuration, etc. In addition, letter element models for recognition by letter element has the general stroke information.

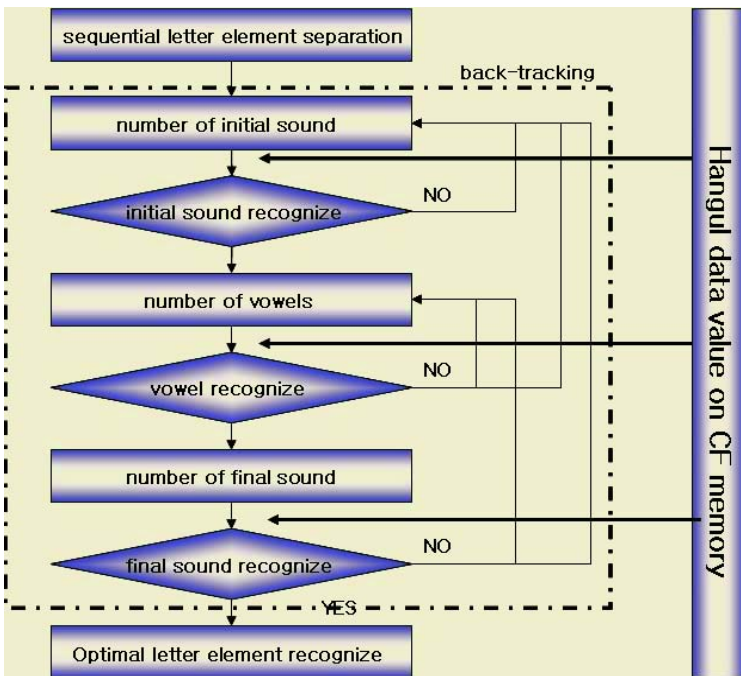


Fig. 7. Processing flow for Letter element separation and Recognition

However, the letter containing excessive curvatures and any scribbles between letter elements is difficult to extract the accurate information, then utilizing only straight line, smooth curve information and stroke configuration in various directions. Fig.8 shows the hierarchical recognition process for the Hangul data ink.

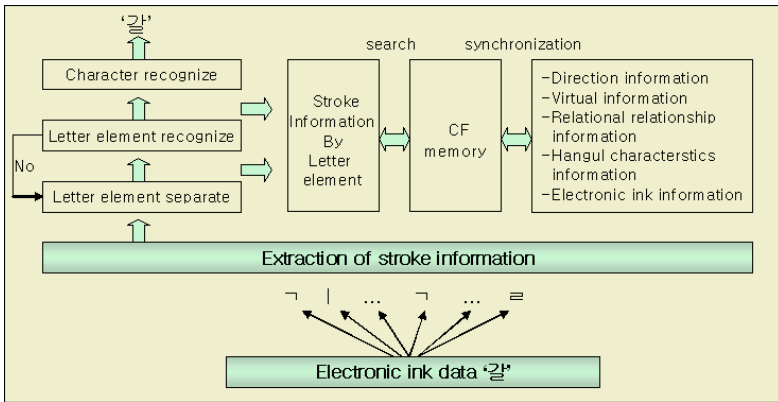


Fig. 8. Hierarchical recognition process

### 4 Implementation and Conclusion

The randomly extracted electronic ink data collection was used for the test, and input type was the Hangul scripts. Visual C# and MS-SQL were applied to construct Hangul data on the CF memory, and the mobile program was prepared in Embedded Visual C++ and performed with the test PDA(Compaq ipaq 3850). Matching rates were measured with the three element, and the model is as follows.

$$R = m(n, t, r) \tag{3}$$

The test results show merits of the matching algorithm as follows. First, the matching algorithm is practical. Generally the number of the electronic ink data using in the PDA and mobile computer is around 300. The suggested algorithm shows the matching rate of nearly 98% even with 300 database. Second, the fast searching speed shows an average 0.8 second in matching speed with 300 database, which is a very fast speed when considering the environment of the PDA having hardware limitations as compared with the desktop computer. Third, one merit of writer-dependent matching algorithm provides the security capability for anybody having different strokes. In case of the writer-dependent matching, the different strokes for the same letter recognize the different form of pattern. Therefore when anyone of different strokes wants to use, the probability of providing desired data becomes low, providing the security capability for the private information.

This study suggested and implemented the Hangul matching system for storing and recognizing the electronic ink form with the Pen-input script data in the PDA. The suggested Hangul matching system calculates the stroke curvature along with the preprocessing procedure and works through the process separating into the basic stroke units. Then it creates the stroke feature vectors by the decision of types of basic stroke unit for calculating the distance value with the dynamic programming. When calculating the distance value, the



edit operation includes deletion, insertion, and exchange operation as well as packing and separation operation in consideration of the characteristics of Hangul. The resulting values are used to compare the Hangul numeric value on the CF memory and recognize Hangul, applying it to the system.

## References

- [1] W. Aref, D. Barbarii, and P. Vallabhaneni. The Handwritten Trie: Indexing Electronic Ink. Technical report, M. I.T.L, October 1994.
- [2] Walid Aref and Daniel Barbar. The Hidden Markov Model Tree Index: A Practical Approach to Fast Recognition of Handwritten Documents in Large Databases. Technical Report MITL-TR-84-93, MITL, January 1994.
- [3] Walid G. Aref, Padmavathi Vallabhaneni, and Daniel Barbar. Towards a realization of handwritten databases: Training and recognition. Technical Report MITL-TR-98-94, Matsushita Information Technology Laboratory, Princeton, NJ, March 1994.
- [4] R. Bakis. Continuous speech word recognition via centisecond acoustic states. In PTOC. ASA Meeting, Washington, DC, April 1976.
- [5] Daniel Barbar. Method to index electronic handwritten documents. Technical Report MITL-TR-77-93, Matsushita Information Technology Laboratory, Princeton, NJ, November 1993.
- [6] R. Carr and D. Shafer. The Power of PenPoint. Addison-Wesley, 1991.
- [7] K. Landau, S. Major, and C. Wiederhold. The Role of PDA in the Office. Notes of the Seminar at PC-EXPO, June 1994.

# Block LDA for Face Recognition

Vo Dinh Minh Nhat and Sungyoung Lee

Kyung Hee University, South Korea  
{vdmnhat, sylee}@oslab.khu.ac.kr

**Abstract.** Linear Discriminant Analysis (LDA) technique is an important and well-developed area of image recognition and to date many linear discrimination methods have been put forward. Despite these efforts, there persist in the traditional LDA some weaknesses. In this paper, we propose a new LDA-based method called Block LDA (BLDA) that can outperform the traditional Linear Discriminant Analysis (LDA) methods. As opposed to conventional LDA, BLDA is based on 2D matrices rather than 1D vectors. That is, we firstly divides the original image into blocks. Then, we transform the image into a vector of blocks. By using row vector to represent each block, we can get the new matrix which is the representation of the image. Finally LDA can be applied directly on these matrices. In contrast to the between-class and within-class covariance matrices of LDA, the size of the these covariance matrices using BLDA is much smaller. As a result, BLDA has three important advantages over LDA. First, it is easier to evaluate the between-class and within-class covariance matrices accurately. Second, less time is required to determine the corresponding eigenvectors. And finally, block size could be changed to get the best results. Experiment results show our method achieves better performance in comparison with the other methods.

**Index Terms:** Linear Discriminant Analysis (LDA), Block Linear Discriminant Analysis (BLDA), face recognition.

## 1 Introduction

Face recognition research has been started in the late 70s and is one of the active and exciting researches in computer science and information technology areas since 1990 [1]. Generally, there are three phases for face recognition, mainly face representation, face detection, and face identification. Face representation is the first task, that is, how to model a face. The way to represent a face determines the successive algorithms of detection and identification. There are a variety of approaches for face representation, which can be roughly classified into three categories: template-based, feature-based, and appearance-based. The simplest template-matching approaches represent a whole face using a single template, i.e., a 2-D array of intensity, which is usually an edge map of the original face image. In a more complex way of template-matching, multiple templates may be used for each face to account for recognition from different viewpoints. Another important variation is to employ a set of smaller facial feature

templates that correspond to eyes, nose, and mouth, for a single viewpoint. The most attractive advantage of template-matching is the simplicity, however, it suffers from large memory requirement and inefficient matching. In feature-based approaches, geometric features, such as position and width of eyes, nose, and mouth, eyebrow's thickness and arches, face breadth, or invariant moments, are extracted to represent a face. Feature-based approaches have smaller memory requirement and a higher recognition speed than template-based ones do. They are particularly useful for face scale normalization and 3D head model-based pose estimation. However, perfect extraction of features is shown to be difficult in implementation. Eigenfaces approach is one of the earliest appearance-based face recognition methods, which was developed by M. Turk and A. Pentland [2] in 1991. This method utilizes the idea of the PCA and decomposes face images into a small set of characteristic feature images called eigenfaces. Recognition is performed by projecting a new face onto a low dimensional linear "face space" defined by the eigenfaces, followed by computing the distance between the resultant position in the face space and those of known face classes.

The Fisherface method [4] combines PCA and the Fisher criterion [9] to extract the information that discriminates between the classes of a sample set. It is a most representative method of LDA. Nevertheless, Martinez *et al.* demonstrated that when the training data set is small, the Eigenface method outperforms the Fisherface method [7]. Should the latter be outperformed by the former? This provoked a variety of explanations. Liu *et al.* thought that it might have been because the Fisherface method uses all the principal components, but the components with the small eigenvalues correspond to high-frequency components and usually encode noise [11], leading to recognition results that are less than ideal. In line with this theory, they presented two enhanced Fisher linear discrimination (FLD) models (EFMs) [11] and an enhanced Fisher classifier [12] for face recognition. Their experiential explanation lacks sufficient theoretical demonstration, however, and EFM does not provide an automatic strategy for selecting the components. Chen *et al.* proved that the null space of the within-class scatter matrix contains the most discriminative information when a small sample size problem takes place [13]. Their method is also inadequate, however, as it does not use any of the information outside the null space. In [5], Yu *et al.* propose a direct LDA (DLDA) approach to solve this problem. It removes the null space of the between-class scatter matrix firstly by doing eigen-analysis. Then a simultaneous diagonalization procedure is used to seek the optimal discriminant vectors in the subspace of the between-class scatter matrix. However, in this method, removing the null space of the between-class scatter matrix by dimensionality reduction would indirectly lead to the losing of the null space of the within-class scatter matrix which contains considerable discriminative information. Rui Huang [10] proposed the method in which the null space of total scatter matrix which has been proved to be the common null space of both between-class and within-class scatter matrix, and useless for discrimination, is firstly removed. Then in the lower-dimensional projected space, the null space of the resulting within-class scatter matrix is calculated. This lower-dimensional null space, combined with the previous projection, represents a subspace of the whole null space of within-class scatter matrix, and is really useful for discrimination. The optimal discriminant vectors of LDA are derived from it.

In the LDA-based face recognition technique, the 2D face image matrices must be previously transformed into 1D image vectors. The resulting image vectors of faces

usually lead to a high dimensional image vector space, where it is difficult to evaluate the between-class and within-class covariance matrices accurately due to its large size and the relatively small number of training samples. Fortunately, the eigenvectors can be calculated efficiently using the SVD techniques and the process of generating these covariance matrices is actually avoided. However, this does not imply that the eigenvectors can be evaluated accurately in this way since the eigenvectors are statistically determined by the between-class and within-class covariance matrices, no matter what method is adopted for obtaining them. In this paper, a new LDA approach called BLDA, is developed for image feature extraction. As opposed to conventional LDA, BLDA is based on 2D matrices rather than 1D vectors. That is, we firstly divides the original image into blocks. Then, we transform the image into a vector of blocks. By using row vector to represent each block, we can get the new matrix which is the representation of the image. Finally LDA can be applied directly on these matrices. In contrast to the between-class and within-class covariance matrices of LDA, the size of the image covariance matrix using BLDA is much smaller. As a result, BLDA has three important advantages over LDA. First, it is easier to evaluate the between-class and within-class covariance matrices accurately. Second, less time is required to determine the corresponding eigenvectors. And finally, block size could be changed to get the best results. The remainder of this paper is organized as follows: In Section 2, the traditional LDA method is reviewed. The idea of the proposed method and its algorithm are described in Section 3. In Section 4, experimental results are presented on the ORL face image database to demonstrate the effectiveness of our method. Finally, conclusions are presented in Section 5.

## 2 Linear Discriminant Analysis

Let us consider a set of  $N$  sample images  $\{x_1, x_2, \dots, x_N\}$  taking values in an  $n$ -dimensional image space, and assume that each image belongs to one of  $c$  classes  $\{C_1, C_2, \dots, C_c\}$ . Let  $N_i$  be the number of the samples in class  $C_i$  ( $i = 1, 2, \dots, c$ ),

$\mu_i = \frac{1}{N_i} \sum_{x \in C_i} x$  be the mean of the samples in class  $C_i$ ,  $\mu = \frac{1}{N} \sum_{i=1}^N x_i$  be the mean

of all samples. Then the between-class scatter matrix  $S_b$  is defined as

$$S_b = \frac{1}{N} \sum_{i=1}^c N_i (\mu_i - \mu)(\mu_i - \mu)^T = \frac{1}{N} \Phi_b \Phi_b^T \quad (1)$$

and the within-class scatter matrix  $S_w$  is defined as

$$S_w = \frac{1}{N} \sum_{i=1}^c \sum_{x_k \in C_i} (x_k - \mu_i)(x_k - \mu_i)^T = \frac{1}{N} \Phi_w \Phi_w^T \quad (2)$$

Also, the total scatter matrix or mixture scatter matrix  $S_t$  is defined as

$$S_t = S_b + S_w = \frac{1}{N} \sum_{i=1}^N (x_i - \mu)(x_i - \mu)^T = \frac{1}{N} \Phi_t \Phi_t^T \tag{3}$$

which is also the covariance matrix of all the samples.

In LDA, the projection  $W_{opt}$  is chosen to maximize the ratio of the determinant of the between-class scatter matrix of the projected samples to the determinant of the within-class scatter matrix of the projected samples, i.e.,

$$W_{opt} = \arg \max_w \frac{|W^T S_b W|}{|W^T S_w W|} = [w_1 w_2 \dots w_m] \tag{4}$$

where  $\{w_i | i=1, 2, \dots, m\}$  is the set of generalized eigenvectors of  $S_b$  and  $S_w$  corresponding to the  $m$  largest generalized eigenvalues  $\{\lambda_i | i=1, 2, \dots, m\}$ , i.e.,

$$S_b w_i = \lambda_i S_w w_i \quad i = 1, 2, \dots, m \tag{5}$$

### 3 Our Proposed LDA

In the LDA-based face recognition technique, the 2D face image matrices must be previously transformed directly into 1D image vectors. The resulting image vectors of faces usually lead to a high dimensional image vector space. However, in our proposed BLDA approach, we firstly divides the original image into  $s = h \times w$  size blocks with  $h, w$  are the height and width of the block. Then, we transform the image into a vector of blocks. By using row vector  $r$  with  $r^T \in \mathbb{R}^s$  to represent each block, we can get the matrix  $X \in \mathbb{R}^{k \times s}$  which is the representation of the image, with  $k$  is the number of blocks. See fig. 1 for the process.

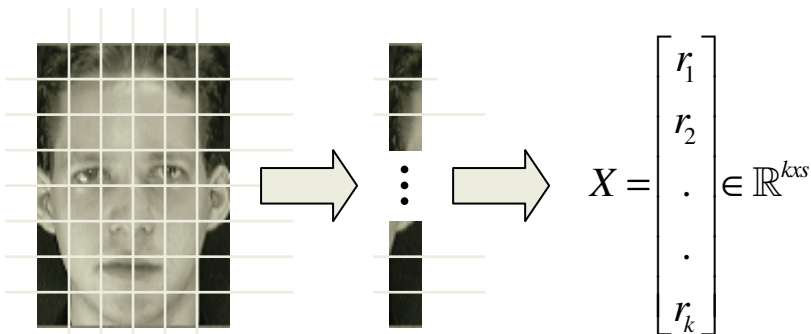


Fig. 1. The process of getting representation of each image

Now, set of  $N$  sample images are represented as  $\{X_1, X_2, \dots, X_N\}$  with  $X_i \in \mathbb{R}^{k \times s}$ . Then the between-class scatter matrix  $S_b$  is re-defined as

$$S_b = \frac{1}{N} \sum_{i=1}^c N_i (\mu_{C_i} - \mu_X)(\mu_{C_i} - \mu_X)^T \quad (6)$$

and the within-class scatter matrix  $S_w$  is re-defined as

$$S_w = \frac{1}{N} \sum_{i=1}^c \sum_{X_k \in C_i} (X_k - \mu_{C_i})(X_k - \mu_{C_i})^T \quad (7)$$

The total scatter matrix is re-defined as

$$S_T = \frac{1}{N} \sum_{i=1}^N (X_i - \mu_X)(X_i - \mu_X)^T \quad (8)$$

with  $\mu_X = \frac{1}{N} \sum_{i=1}^N X_i \in \mathbb{R}^{k \times s}$  is the mean image of all samples and  $\mu_{C_i} = \frac{1}{N_i} \sum_{X \in C_i} X$

be the mean of the samples in class  $C_i$ .

Similarly, a linear transformation mapping the original  $k \times s$  image space into an  $m \times s$  feature space, where  $m < k$ . The new feature matrices  $Y_i \in \mathbb{R}^{m \times s}$  are defined by the following linear transformation :

$$Y_i = W^T (X_i - \mu_X) \in \mathbb{R}^{m \times s} \quad (9)$$

where  $i = 1, 2, \dots, N$  and  $W \in \mathbb{R}^{k \times m}$  is a matrix with orthonormal columns. And the projection  $W_{opt}$  is chosen with the criterion same as that in (4).

After a transformation by BLDA, a feature matrix is obtained for each image. Then, a nearest neighbor classifier is used for classification. Here, the distance between two arbitrary feature matrices  $Y_i$  and  $Y_j$  is defined by using Euclidean distance as follows :

$$d(Y_i, Y_j) = \sqrt{\sum_{u=1}^k \sum_{v=1}^s (Y_i(u, v) - Y_j(u, v))^2} \quad (10)$$

Given a test sample  $Y$ , if  $d(Y, Y_c) = \min_j d(Y, Y_j)$ , then the resulting decision is  $Y$  belongs to the same class as  $Y_c$ .

### 4 Experimental Results

This section evaluates the performance of our proposed algorithm compared with that of the original LDA algorithm based on using ORL. In the ORL database, there are ten different images of each of 40 distinct subjects. For some subjects, the images were taken at different times, varying the lighting, facial expressions (open / closed eyes, smiling / not smiling) and facial details (glasses / no glasses). All the images were taken against a dark homogeneous background with the subjects in an upright, frontal position (with tolerance for some side movement).

In our experiments, firstly we tested the recognition rates with different number of training samples.  $k(k = 2, 3, 4, 5)$  images of each subject are randomly selected from the database for training and the remaining images of each subject for testing. For each value of  $k$ , 5 runs are performed with different random partition between training set and testing set. The block size 3 by 3 is used in this first experiment, and two methods which are LDA and BLDA are performed. Table 1. shows the recognition results of the best recognition accuracy among all the dimension of feature vectors. It means we test on all dimension of feature vectors and choose the best recognition accuracy.

**Table 1.** The recognition rates on ORL database with different training samples of two methods (LDA, BLDA – 3x3 block size)

| Training samples  | 2     | 3     | 4     | 5     |
|-------------------|-------|-------|-------|-------|
| LDA (Fisherfaces) | 78.83 | 86.9  | 91.03 | 93.6  |
| BLDA (3x3)        | 86.22 | 89.61 | 93.53 | 95.83 |

Next, we try to test BLDA approach performance when the block size is changed. And several results can be show in the Table 2. The same protocol as previous experiments, we choose the recognition result of the dimension feature vectors which give the best accuracy.

**Table 2.** The recognition rates with different block sizes

| Size of block | Training samples |       |       |       |
|---------------|------------------|-------|-------|-------|
|               | 2                | 3     | 4     | 5     |
| [2x2]         | 86.77            | 90.4  | 94.23 | 96.48 |
| [3x3]         | 86.22            | 89.61 | 93.53 | 95.83 |
| [5x5]         | 87.88            | 90.92 | 94.86 | 96.89 |
| [10x2]        | 86.17            | 90.41 | 93.6  | 95.98 |
| [10x10]       | 83.51            | 88.52 | 90.55 | 93.57 |

From Table 2, it seems to be that the block size 5x5 give the best recognition results among all. However we still not yet find the relationship between the block size and the recognition result.

## 5 Conclusions

A new LDA-based method (BLDA) for face recognition has been proposed in this paper. As opposed to conventional LDA, BLDA is based on 2D matrices rather than 1D vectors. That is, we firstly divides the original image into blocks. Then, we transform the image into a vector of blocks. By using row vector to represent each block, we can get the new matrix which is the representation of the image. Finally LDA can be applied directly on these matrices. In contrast to the covariance matrix of LDA, the size of the image covariance matrix using BLDA is much smaller. As a result, BLDA has three important advantages over LDA. First, it is easier to evaluate the covariance matrix accurately. Second, less time is required to determine the corresponding eigenvectors. And finally, block size could be changed to get the best results.

## References

1. W. Zhao, R. Chellappa, A. Rosenfeld, J. Phillips: Face Recognition: A Literature Survey. Technical Report, CFAR-TR00-948, Univ. of Maryland, 2000. (Revised 2002)
2. M Turk, A Pentland: Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, vol. 3 (1991) 71-86.
3. Jian Yang, Zhang D., Frangi A.F., Jing-yu Yang: Two-dimensional PCA: a new approach to appearance-based face representation and recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* , vol. 26 (2004) 131 – 137.
4. P. N. Belhumeur, J. P. Hespanha, D. J. Kriegman: Eigenfaces vs. fisherface: Recognition using class specific linear projection. *IEEE Trans. Pattern Anal. Machine Intel.*, vol. 19 (1997) 711–720.
5. H. Yu, J. Yang: A direct LDA algorithm for high-dimensional data with application to face recognition. *Pattern Recognit.*, vol. 34 (2001) 2067–2070.
6. M. Loog, R. P. W. Duin, R. Haeb-Umbach: Multiclass linear dimension reduction by weighted pairwise fisher criteria. *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 23 (2001) 762–766.
7. A. M. Martinez, A. C. Kak: PCA versus LDA. *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 23 (2001) 228–233.
8. D. H. Foley, J. W. Sammon: An optimal set of discrimination vectors. *IEEE Trans. Comput.*, vol. C-24 (1975) 281–289.
9. R. A. Fisher: The use of multiple measurements in taxonomic problems. *Ann. Eugenics*, vol. 7 (1936) 178–188.
10. Rui Huang, Qingshan Liu, Hanqing Lu, Songde Ma: Solving the small sample size problem of LDA. *Proceedings. 16th International Conference on Pattern Recognition*, vol. 3 (2002).
11. C. Liu, H. Wechsler: Robust coding scheme for indexing and retrieval from large face databases. *IEEE Trans. Image Processing*, vol. 9 (2000) 132–137.
12. Chengjun Liu, Wechsler H.: A shape- and texture-based enhanced Fisher classifier for face recognition. *IEEE Trans. Image Processing*, vol. 10 (2001) 598–608.
13. L. Chen, H. M. Liao, M. Ko, J. Lin, G. Yu: A new LDA-based face recognition system which can solve the small sample size problem. *Pattern Recognit.*, vol. 33 (2000) 1713–1726.



# Image Processing with CNN in a FPGA-Based Augmented Reality System for Visually Impaired People

F. Javier Toledo, J. Javier Martínez, F. Javier Garrigós,  
and J. Manuel Ferrández

Dpto. Electrónica, Tecnología de Computadoras y Proyectos,  
Universidad Politécnica de Cartagena, 30202 Cartagena, Spain  
[javier.toledo@upct.es](mailto:javier.toledo@upct.es)

**Abstract.** A cellular neural network is proposed as the main processing core in a novel FPGA-based augmented reality system. The described application is focused on visually impaired people aid. The aim is to enhance the user's knowledge of the environment with useful information extracted by image processing. A CNN architecture oriented to hardware implementation on FPGA is presented, and used as the image processor in a fully FPGA-based system. So, CNNs and FPGAs are combined in a system which makes the most of their characteristics to achieve high performance and versatility.

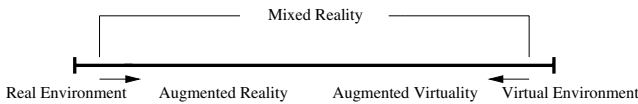
## 1 Introduction

Nowadays, the advantages of Cellular Neural Networks (CNNs) are already well known in image processing. It is specially interesting for the application described in this paper the facts that they make real time output feasible, that can be tuned to produce the desired result and the possibility to perform different processing algorithms just by changing the template set.

On the other hand, the Field Programmable Gate Array (FPGA) devices' internal structure makes them perfectly suitable for executing in parallel elementary digital processing tasks. This, together with their reconfigurability capabilities, explains why FPGAs constitute a competitive alternative for high performance signal [1], [2] and image processing [3], [4]. Moreover, because of their flexibility it is possible to implement not only specific algorithms but also interfaces, controllers, glue logic, even microprocessors, and so to integrate the whole system in just one chip (SOC, system on a chip concept). Therefore, parallelism and flexibility (and cost) become FPGAs a very interesting alternative to ASICs for CNN implementation. Several works have focused on the FPGA implementation of CNNs [5], [6], [7]. In this work, advantage of CNNs characteristics is taken by means of the implementation on FPGA for an augmented reality system.

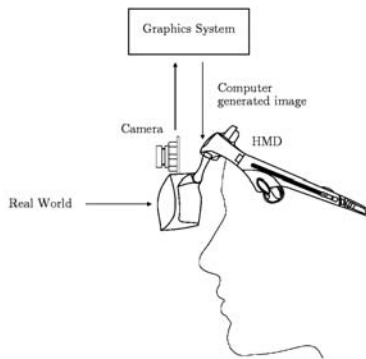
Augmented Reality (AR) is a highly interdisciplinary field which has received increasing attention since late 90s. Basically, it consists of a combination of the

real scene viewed by a user and a artificially generated image, running in real time. So, AR allows the user to see the real world supplemented with some information considered as useful, enhancing the user’s perception and knowledge of the environment (Fig. 1). Many areas of knowledge are involved in AR, including signal and image processing, wearable computing, computer vision, computer graphics and information visualization. The applications of AR spread on a wide range, from entertainment to military purposes, including medical visualization and training, engineering design, manufacturing and, as proposed in this work, impairment aids.



**Fig. 1.** Milgram’s reality-virtuality continuum. It shows how real and virtual worlds can merge. Augmented reality is nearer the real world than the virtual world because the real world perception predominates over the generated data

AR usually implies the use of a Head Mounted Display (HMD), in order to increase the sense of presence [8], [9]. Because of the particular characteristics of our application, an optical see-through HMD has been chosen in this work (Fig. 2), together with a camera mounted on the HMD to acquire an image of the environment.



**Fig. 2.** Optical see-through approach adopted in this work. Optical combiners in front of the user’s eyes inside the HMD allow to see the virtual image, generated in the graphics system from the camera image and, thanks to their transmissive properties, to see through them the real world

In this work, an AR application is developed for people affected by tunnel vision. As it will be described in section 2, people suffering from this visual disorder can be aided by enhancing their knowledge of the environment with

information obtained by processing images from a camera. The image processing is performed with a CNN. The CNN model is presented in section 3. The overall system architecture is described in section 4. The results of the implementation and some examples are shown in section 5 and, finally, conclusions are presented in section 6.

## 2 Target Application

Tunnel vision consists of a loss of peripheral vision, while retaining a high resolution central vision. It is associated mainly to several eye diseases such as retinitis pigmentosa and glaucoma. The loss of the peripheral visual field affects considerably the patient's ability to localize objects and people and navigate, and consequently, his relationship with people and the environment (Fig. 3).



**Fig. 3.** Simulation of patient's view affected by tunnel vision (right). The severe reduction of the visual field can be observed comparing with the normal vision (left)

The devices used to aid affected people are traditionally based on techniques for reducing optically the size of the objects. This minification process allows to see a wider field of view, but at the expense of lessening the high resolution central vision.

To overcome this disadvantage, Peli et al. [10] have proposed an augmented reality-based method to enhance the user's knowledge of the environment by increasing his field of view. The contour of the image obtained from a video camera is superimposed on his own view of the entourage seen with a see-through HMD. In their work, contour information is generated by an edge detection algorithm, performed by a four-pixel neighbour gradient filter and a threshold function, running on a laptop PC [11]. They draw the conclusion that, although patients consider the system useful for navigating and obstacle avoiding, a specifically designed system to perform image processing (edge detection) and increase frame rate is necessary. Obviously, an effective improvement of the user's environment perception requires real time processing.

With this aim, a system based on the FPGA implementation of the Canny edge detection algorithm to extract contour information, has been recently developed [12]. Now, we propose the use of CNNs to increase the versatility of the system through the possibility of using different templates, while achieving real time processing, implemented on reconfigurable hardware too.

### 3 The CNN Discrete Model

The discrete CNN implemented is based on the model shown in Figure 4, derived by approximating the continuous model with the Euler method. Three more models have been evaluated, obtained through transformation of the continuous model by differential algorithm (TDA-backward), numeric integration algorithm (TIA-Tustin) and response-invariant algorithm (RIT-first order impulse). The one chosen here offers the best results approximating the continuous model, while requiring the minimum computational cost [13]. As can be seen in Figure 4, we restrict the system to the use of the  $3 \times 3$  neighbourhood. We have adopted a pixel pipeline approach for data input and output, where 5 identical stages are connected in cascade.

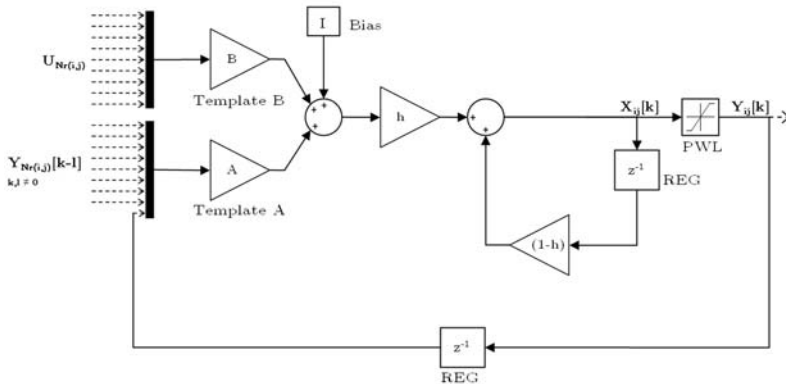


Fig. 4. The CNN discrete model

### 4 System Description

The proposed system consists basically of a camera to acquire images of the environment, a head mounting display to visualize the information that enhances the user’s vision, and a Xilinx Virtex-II FPGA as processor and controller unit.

The camera is the M4088, a monochrome camera module with 8 bits digital output that provides a very low cost solution for higher quality video application. It uses OmniVision’s CMOS image sensor OV5017 and outputs a  $384 \times 288$

pixels image at 50 frames per second. A digital interface facilitates the configuration and initialization of the camera, executed from the FPGA. The camera pixel clock frequency is 14.31 MHz.

The data from the camera are written on SRAM memory at the frequency determined by the camera frame rate, and simultaneously they are read from the memory to be processed by the FPGA. Due to the large size, data between the different stages of the processing are stored in external SRAM memories. The output of the last stage is also stored in SRAM memory, where data are read from at the VGA frame rate to be sent to the HMD. The memory interface for controlling the write and read operations is implemented on the own FPGA.

VGA synchronization signals for the HMD are generated by the FPGA to show a  $640 \times 480$  pixels image at 60 frames per second. The image pixels values are converted into analog signals by the ADV7123 from Analog Devices, a triple channel digital to analog converter for video applications.

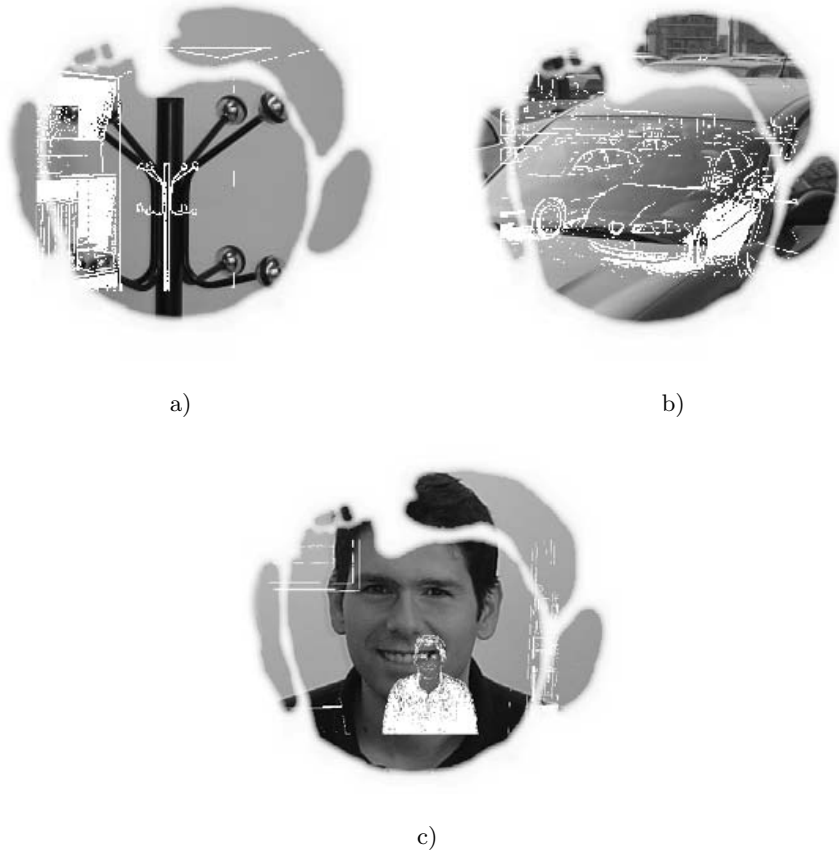
The head mounted display used in this work is the Sony Glasstron PLM-S700E. It is a binocular and high resolution device with SVGA/VGA input and adjustable see-through capability. Its design and light weight (120 gr. head) allows its use with correction glasses and makes it easy the movement of the head and the mobility of the person. All these characteristics become this HMD into a suitable choice for our application.

The processed image must be properly positioned in the HMD screen in order to maximize utility making use of the patient's visual remains. Therefore, the contour information is reduced in a minification factor and centered in the user's high resolution central vision field. This image minification factor depends on the width of the field of view where the scene is presented, and determines how small objects in the HMD look: the higher minification value, the narrower field of view and the smaller objects. The suitable value depends on each patient, not only on his visual conditions but also on his adaptability to the composed scene and his personal preferences. In the results shown in section 5 the adopted value was 4.

## 5 Results

The design has been developed in VHDL, synthesized with Xilinx XST and implemented on a Xilinx XC2V4000 FPGA. It uses a 22.3% of the slices available on the Xilinx XC2V4000 FPGA and a 83% of the embedded multipliers.

The results obtained in different situations, both indoor and outdoor environments, are shown in Figure 5. In all of them, the contour information extracted from the image of the camera by the CNN is superimposed on the central vision field. A residual  $10^\circ$  field of view has been considered. In these examples, a  $57.9^\circ$  (H)  $\times$   $45.7^\circ$  (V) lens has been used. A wider field of view can be acquired by the camera with the appropriate lens.



**Fig. 5.** Simulation of patient's view through the HMD: a) in an indoor environment. It can be observed that the clothes stand is situated on a corner and beside a shelf. b) in an outdoor environment, showing a car in a parking. c) looking at a person sitting, who is between a shelf and a picture

## 6 Conclusions

An augmented reality system has been developed to aid visually impaired people. In order to achieve the requirements of performance and flexibility, a CNN has been the adopted solution to perform image processing, and an FPGA device has been chosen as the hardware platform. With the proposed system, the patient's limited view of the environment is enhanced superimposing on it the contour information extracted from a video camera image by means of a head mounted display with see-through capability. So, the person's abilities to localize objects, orientate and navigate are improved. The results make clear the viability and utility of the system.

Nevertheless, the CNN architecture must be optimized, in order to reduce the used resources. Future works will focus on this topic firstly. Also, the increase of functionality by performing the CNN other image processing algorithms, and the inclusion of textual information to the augmented scene or a graphic interface to facilitate user's customization of different configuration parameters will be evaluated.

## Acknowledgements

This research is being funded by Ministerio de Ciencia y Tecnología TIC 2003-09557-C02-02 and Fundación Séneca PI-26/00852/FS/01.

## References

1. Tessier, R., Burlleson, W.: "Reconfigurable computing for digital signal processing: a survey", *Journal of VLSI Signal Processing*, vol. 28, no. 1, 2001, 7–27.
2. Goslin, G.R.: "Using FPGA in digital signal processing application", *Intern. Conf. on Signal Processing Applications and Technology*, Boston, USA, 1995, 145–149.
3. Kessal, L., Abel, N., Demigny, D.: "Real time image processing with dynamically reconfigurable architecture", *Real Time Imaging*, vol. 9, no. 5, Elsevier, 2003, 297–313.
4. Draper, B., Beveridge, J.R., Böhm, W., Ross, C., Chawathe, M.: "Accelerated image processing on FPGAs", *IEEE Trans. on Image Processing*, vol. 12, no. 12, 2003, 1543–1551.
5. Nagy, Z., Szolgay, P.: "Configurable multi-layer CNN-UM emulator on FPGA", *Proc. IEEE Int. Workshop on Cellular Neural Networks and Their Applications*, Frankfurt, Germany, 2002, 164–171.
6. Malki, S., Spaanenburg, L.: "CNN image processing on a Xilinx Virtex-II 6000", *Proc. European Conference on Circuit Theory and Design*, 2003, Krakow, Poland, 261–264.
7. Martínez, J.J., Toledo, F.J., Ferrández, J.M.: "New emulated discrete model of CNN architecture for FPGA and DSP applications", *7th Int. Work-Conference on Artificial and Natural Neural Networks*, LNCS 2687, Menorca, Spain, 2003, 33–40.
8. "Head mounted display survey", *Real Time Graphics*, vol. 10, no. 2, 2001, 6–10.
9. Azuma, R.T.: "A survey of augmented reality", *Presence: Teleoperators and virtual environments*, vol. 6, no. 4, 1997, 355–385.
10. Vargas Martín, F., Peli, E.: "Augmented view for tunnel vision: device testing by patients in real environments", *Digest of Technical Papers, Society for Information Display International Symposium*, San Jose, USA, 2001, 602–605.
11. Vargas Martín, F., Peli, E.: "Augmented-view for restricted visual field: multiple device implementations", *Optometry and Vision Science*, vol. 79, no. 11, 2002, 715–723.
12. Toledo, F.J., Martínez, J.J., Garrigós, F.J., Ferrández, J.M.: "Augmented reality system for visually impaired people based on FPGA", *Proc. 4th IASTED Int. Conf. on Visualization, Imaging and Image Processing*, Marbella, Spain, 2004, 715–723.
13. Martínez, J.J., Toledo, J., Ferrández, J.M.: "Implementation of a discrete cellular neuron model (DT-CNN) architecture on FPGA", *2nd SPIE Int. Symposium on Microtechnologies for the New Millennium, Conf. on bioengineered and bioinspired systems (2005) (to appear)*.

# A Gradient Descent MRI Illumination Correction Algorithm

M. Garcia, E. Fernandez, M. Graña, and F.J. Torrealdea\*

Dept. CCIA, UPV/EHU, Apdo. 649, 20080 San Sebastian, Spain  
ccpgrrom@si.ehu.es

**Abstract.** Magnetic Resonance Images(MRI) are piecewise constant functions that can be corrupted by an inhomogeneous illumination field. We propose a gradient descent parametric illumination correction algorithm for MRI. The illumination bias is modelled as a linear combination of 2D products of Legendre polynomials. The error function is related to the classification error in the bias corrected image. In this work the intensity classes are given beforehand, so the adaptive algorithm is used only to estimate the bias field. We test our algorithm against Maximum A Posteriori algorithms over some images from the ISBR public domain database.

## 1 Introduction

The MRI is different from the other medical visualization techniques because it is not the result of a tomography transformation. MRI allows to visualize with great contrast the soft tissues and has revolutionized the capacity to diagnose the pathologies that affect them. The visualization of magnetic resonance is based on the phenomenon known as nuclear magnetic resonance (NMR). The MRI results from the aggregated measurements of the tissues composition at the molecular level. The MRI has a high space resolution and provides much information on the anatomical structure, allowing quantitative pathological or clinical studies, the derivation of digitized anatomical atlases and also the guide before and during the therapeutic intervention. Many of the image analysis tasks of the radiologists are repetitive, for this reason the introduction of automated methods of image process is desired. The automated image processing range from the impulsive noise elimination, applying linear filters and others like the anisotropic filters, automated image to segmentation and registration. The image registration is the alignment of different images to obtain a more complete visualization. It is applied in the fusion of images from multiple modalities and the comparison of the patients data with anatomical atlases. The image segmentation consists of its decomposition in regions. The regions identification criteria vary depending on the application. The image segmentation is critical in applications like the

---

\* The Spanish Ministerio de Educacion y Ciencia supports this work through grants DPI2003-06972 and VIMS-2003-20088-c04-04.



diagnosis of schizophrenia, the detection of tumors and the surgery based on augmented reality. The segmentation methods must be reliable and reproducible. For example, in studies of degenerative cerebral diseases like the schizophrenia, Alzheimer’s disease [6], multiple sclerosis, it is necessary the precise measurement of the amount of gray matter, white matter, damages in the white matter, CSF and their space distributions and temporary changes. Given the piece wise constant nature of MRI the automatic image segmentation is trivial if the tissue intensity classes are known. There are two problems affecting the image segmentation: the additive noise and the illumination inhomogeneity. Usually we can cope with the additive noise by conventional clustering algorithms however the illumination inhomogeneity induced distortion need more sofisticate methods.

In this paper we deal with the correction of illumination inhomogeneities proposing gradient descent algorithm. We compare this algorithm with Bayesian state of the art algorithms. The method is described in Section 2, and experimental results of illumination field correction algorithms in MRI are presented in Section 3. Finally, some conclusions are draw in Section 4.

## 2 Description of the Algorithms

We will denote the observed and classification images as  $\mathbf{y} = (y_i; i \in I)$ ;  $\mathbf{x} = (x_i; i \in I, x_i \in \Omega)$  where  $i \in I \subset \mathbb{N}^2$  is the pixel site in the discrete lattice of the image support. The assumed image formation model is the following one:

$$y_i = \beta_i \cdot r_i + \eta_i, \tag{1}$$

where  $\beta_i$  is the bias due to the illumination,  $r_i$  is the reflectance and  $\eta_i$  is the additive noise. In the MRI we have the additional restriction of the reflectance values belonging to a discrete set,  $\Gamma = \{\mu_1, \dots, \mu_c\}$  so  $r_i = \mu_{x_i}$ . The illumination correction problem is the problem of estimating the image segmentation  $\mathbf{x}$  and the illumination bias field  $\beta$  from  $\mathbf{y}$ . The algorithms that we describe in the following, perform this estimation either as a Maximum *A Posteriori* (MAP) probability estimation process, or as an error minimization process.

### 2.1 BMAP

The BMAP [5] is a bayesian image processing algorithm [2, 3] which consists on the MAP estimation of the image classification for a given illumination bias, whose maximum likelihood estimation in its turn is performed given an image classification. The algorithm, thus, iterates both conditional estimations. for its presentation we will begin with the form that assume the conditional probability and *a priori* distribution. The conditional probability distribution is:

$$p(y_i | x_i) = \frac{1}{\sqrt{2\pi\sigma_{x_i}^2}} \exp \left\{ -\frac{1}{2} \left( \frac{y_i - (1 + \beta_i)\mu_{x_i}}{\sigma_{x_i}} \right)^2 \right\}, \tag{2}$$

It incorporates the hypothesis that the additive noise is a gaussian distributed, that is, the difference of the observed image intensity with respect to the illumination corrected tissue class intensity average follows a gaussian distribution.

The *a priori* probability distribution is modelled by a Markovian field:

$$p(\mathbf{x}) = \frac{1}{Z} \exp \left\{ - \sum_{c \in C} V_c(\mathbf{x}) \right\}, \quad (3)$$

where  $Z$  is the normalization constant and the sum of the potentials  $V_c(\mathbf{x})$  extends to the set of cliques  $C$  defined on the lattice of pixel sites. The potentials model the spatial relationships between pixel intensities and allow to introduce smoothness restrictions or other global restrictions. Considering only the cliques formed by one and two pixels, we have that the potentials can be written like:  $\sum_c V_c(\mathbf{x}) = \sum_i \{V_1(x_i) + V_2(x_i)\}$ . The local potentials can be expressed like:

$$V_1(x_i) = \sum_{\omega \in \Omega} -\alpha_{0\omega} \delta(x_i = \omega); \quad V_2(x_i) = \sum_{i,j \in c_l(i)} [\alpha_l (-\delta(x_i = x_j) + \delta(x_i \neq x_j))], \quad (4)$$

In these expressions  $\delta(c) = \begin{cases} 1 & \text{if } c \\ 0 & \text{otherwise} \end{cases}$ . We denote by  $c_l(i)$  the cliques of order two within a neighborhood of radius  $l$  around the reference pixel  $i$ :  $c_l(i) = \{j : j \in I, |i - j| < \sqrt{l}, i \neq j\}$ . The potentials  $V_1(x_i)$  correspond to the *a priori* probabilities of pixel classes. The potentials  $V_2(x_i)$  specify a classification smoothness restriction, penalizing the classifications of neighboring pixels different from the central pixel classification and rewarding those that are equal.

For computational reasons, in [5] the neighborhood radius considered was 2. Therefore the *a priori* probability density parameters to be set are:  $\alpha = (\alpha_{o\omega}, \omega \in \Omega; \alpha_1, \alpha_2)$ . The parameters  $\alpha_{o\omega}$  are estimated from the *a priori* probabilities of the pixel classes. We will not follow the iterative estimation procedure described in [5] for the order two parameters because we have verified that it adds little to the algorithm accuracy.

The problem of the MAP estimation of the classification image  $\mathbf{x}^* = \arg \max_{\mathbf{x}} p(\mathbf{x}|\mathbf{y})$  becomes an minimization problem on the energy  $u(\mathbf{x}|\mathbf{y}) \propto -\log(p(\mathbf{x}|\mathbf{y}))$  .:

$$\mathbf{x}^* = \arg \min_{\mathbf{x}} u(\mathbf{y}|\mathbf{x}, \beta^*, \theta^*, \alpha^*). \quad (5)$$

The energy minimization can be made of diverse ways, applying methods global random search like the simulated annealing [1]. In [5] it is performed by the application of ICM, which is a greedy local search algorithm . ICM does not guarantee optimal global but it gives good suboptimal solutions in a reasonable computation time. ICM assumes that pixels are independent, that is:

$$p(\mathbf{x}|\mathbf{y}) = \prod_i p(x_i|y_i), \quad (6)$$

so that the energy function can be expressed as the sum of the individual pixel energies

$$u(\mathbf{x}|\mathbf{y}) = \sum_i u(x_i|y_i). \tag{7}$$

In terms of the *a priori* and conditional probabilities, the local energy of each pixel can be written:

$$u(x_i|y_i) = -\frac{1}{2} \left( \frac{y_i - (1 + \beta_i)\mu_{x_i}}{\sigma_{x_i}} \right)^2 + \log(\sigma_{x_i}) + V_1(x_i) + V_2(x_i) \tag{8}$$

ICM starts from an initial arbitrary classification  $\mathbf{x}(0)$  and iterates the classification of the image through the optimization of the local pixel energy, until classification changes fall below a threshold. The complete BMAP algorithm iterates the MAP estimation of the classification by the ICM, the maximum likelihood estimation of the bias, the class intensity averages and their variances

$$\hat{\beta}_i = \frac{y_i}{\mu_{x_i}} - 1, \hat{\mu}_\omega = \frac{\sum_{i \in R_\omega} y_i(1 + \beta_i)}{\sum_{i \in R_\omega} (1 + \beta_i)^2}, \hat{\sigma}_\omega^2 = \frac{\sum_{i \in R_\omega} (y_i - (1 + \beta_i)\mu_\omega)^2}{|R_\omega|},$$

### 2.2 Wells

The Wells algorithm [8] tries to make the MAP estimation of the illumination bias

$$\hat{\beta} = \arg \max_\beta p(\beta | \mathbf{y}). \tag{9}$$

As it's well-know  $p(\beta|\mathbf{y}) \propto p(\mathbf{y}|\beta)p(\beta)$ . Assuming the pixel intensity statistical independence, the *a priori* probability density of the complete image can be written like

$$p(\mathbf{y}|\beta) = \prod_i p(y_i|\beta_i). \tag{10}$$

And the conditional probability of the intensity is obtained computing the sum over tissue class marginals

$$p(y_i|\beta_i) = \sum_{x_i} p(y_i, x_i|\beta_i) = \sum_{x_i} p(y_i|x_i, \beta_i)p(x_i). \tag{11}$$

The conditional probability of the observed image  $p(y_i|x_i, \beta_i)$  follows the same distribution as in equation 2. The *a priori* probability of the classes is assumed uniform in [8]. The *a priori* illumination bias probability is modeled by a random multidimensional variable with Gaussian distribution  $p(\beta) = G_{\psi_\beta}(\beta)$ , where  $\psi_\beta$  is the  $n \times n$  covariance matrix for the entire bias field. The approach taken in [8] to search the MAP solution was to solve the root of the posterior probability derivative. This leads to the following Expectation Maximization algorithm.

A necessary condition for a maximum of the *a posteriori* probability of  $\beta$  is that its gradient with respect to  $\beta$  is zero. After some formal derivations, this condition leads to the following equality  $\hat{\beta} = H\bar{R}$  where  $H$  is in general intractable because it depends on the covariance matrix of the illumination bias.

It is assumed to be well approximated by a low pass filter. The weighted residuals  $\bar{R}$  are given by  $\bar{R}_i \equiv \sum_j W_{ij} \sigma_j^{-1} (y_i - \mu_j)$  where  $W_{ij}$  is equivalent to the *a posteriori* probability of class  $j$  in pixel site  $i$ . The expectation maximization algorithm consists of the iteration of the following steps

$$W_{ij} \leftarrow \frac{[p(x_i) G_{\sigma_{x_i}}(y_i - \mu(x_i) - \beta_i)]_{x_i=tejido-clase-j}}{\sum_{x_i} p(x_i) G_{\sigma_{x_i}}(y_i - \mu(x_i) - \beta_i)}. \tag{12}$$

$$\hat{\beta} \leftarrow H\bar{R} \tag{13}$$

### 2.3 GradClassLeg

GradClassLeg (GCL) is our own proposition of an illumination correction algorithm. In this algorithm we have modeled the illumination bias as a linear combination of 2D products of Legendre polynomials given by

$$b(i, \mathbf{p}) = \sum_{k=0}^m \sum_{l=0}^{m-k} p_{ij} P_k(i_x) P_l(i_y); \tag{14}$$

where  $i = (i_x, i_y)$  and  $P_k(\cdot)$  is the Legendre polynomial of  $k$  order. The number of parameters that compose  $\mathbf{p}$  is  $n = (m + 1) \frac{m+2}{2}$ . Given an illumination field, the image correction error that we consider as the objective function is the following one:

$$e(\mathbf{p}) = \sum_i \prod_{k=1}^c \left( \frac{y_i}{\hat{b}(i, \mathbf{p})} - \mu_k \right)^2 \tag{15}$$

GCL is a gradient descent algorithm of this function of error:

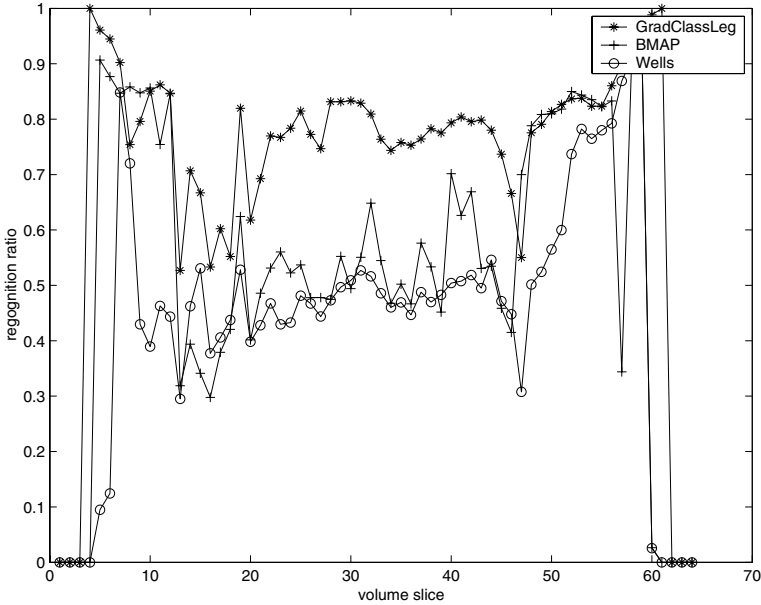
$$\mathbf{p}_{t+1} = \mathbf{p}_t + \alpha_t \nabla e(\mathbf{p}), \tag{16}$$

The iterative application of equation 16 starts from a random initial bias. The gradient vector is  $\nabla_{\mathbf{p}} e(\mathbf{p}) = \left\{ \frac{\partial}{\partial p_{ij}} e(\mathbf{p}) \right\}$ , where

$$\frac{\partial}{\partial p_{ij}} e(\mathbf{p}, I) = \sum_{\mathbf{x}} \sum_m \prod_{k \neq m} \left( \frac{f(\mathbf{x})}{\hat{b}(\mathbf{x}, \mathbf{p})} - \mu_k \right)^2 \left( \frac{f(\mathbf{x})}{\hat{b}(\mathbf{x}, \mathbf{p})} - \mu_j \right) \frac{-f(\mathbf{x}) P_i(x) P_j(y)}{\hat{b}^2(\mathbf{x}, \mathbf{p})}. \tag{17}$$

## 3 Results on Brain MRI

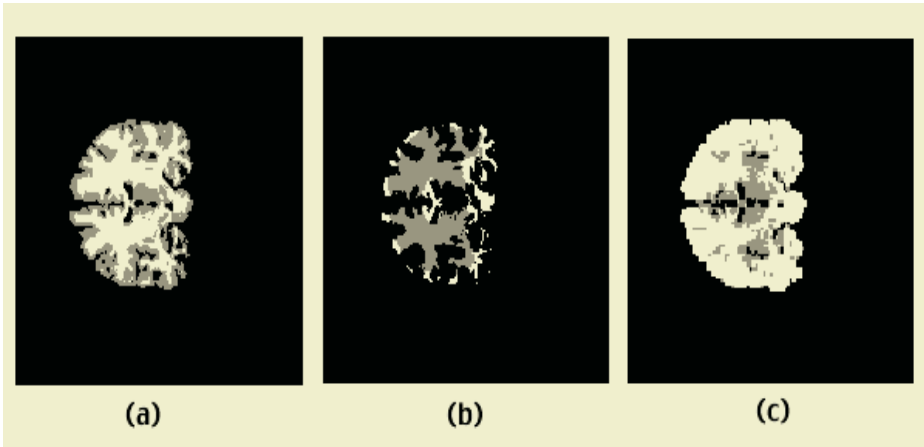
In this section we present the experimental results of MRI illumination inhomogeneity correction algorithms over images downloaded from *International Brain Segmentation Repository (IBSR)* [4] maintained by the Massachusetts General Hospital, Center for Morphometric Analysis. The images are T1-weighted images of normal subjects. There are manual segmentations of the regions corresponding to the gray matter, white matter and CSF. The skull and empty space have



**Fig. 1.** Accuracy classification results of the pixels into gray matter, white matter and CSF obtained by GCL, Wells and BMAP on each slice

been removed from both the MRI and classification images. The images consist of slices of  $256 \times 256$  pixels, 8 bits/pixel corresponding to coronal cuts that are grouped in volumes. The number of slices varies between images. The manual segmentations are provided like volumes in which the values of the pixels are:  $no\ class = 0, csf = 128, gray = 192, white = 254$ . The illumination inhomogeneities, if they exist, are unknown. The GradClassLeg, Wells and BMAP algorithms have been tested on these images. They are applied independently to each slice, without downsizing. The algorithms have not performed the estimation of the intensity class means, they have assumed the nominal values of classification images as the true values of the intensity classes means.

Strictly speaking, we do not realize an illumination correction process on the 3D MRI data, we perform a succession of independent 2D processes. Extension of GCL to 3D data is still in the stage of fine tuning of the gradient descent parameters. The algorithms have been used to estimate the illumination bias. The image is corrected with this bias and the pixel classification according to the intensity values assigned to each class in the manual segmentation images. Each foreground pixel is assigned to the class with the closest intensity mean value. The classification accuracy at each slice is computed as the average percentage of correct classification for each class, weighted by the percentage of pixels of each class in the slice (the *a priori* probabilities of the class in the slice). The operational parameter values used on each algorithm are the following ones:



**Fig. 2.** The slice #32 classification images obtained by the algorithms assuming the intensity nominal values corresponding to each classe: (a) GCL (b) BMAP (c) Wells

- GradClassLeg:  $\alpha = 0.01$ , 25 steps of iteration, minimum order 1, maximum order 5.
- BMAP: neighbourhood 2, MRF potentials that modeled the a priori probability  $\alpha_1 = \alpha_2 = 0.01$ , diameter of the smoothing mask of slant 20 ,
- Wells algorithm: smoothing mask with diameter 25.

The figure 1 plots the classification accuracy of each algorithm over an image shows the algorithms results over one image from ISBR. The plots present the classification accuracy of each algorithm over each image slice. It stands out that the GradClassLeg improves in almost all slices the other two algorithms. The figure 2 displays for the specific case of slice #32 the classification images of the corrected image using the nominal values of the intensity classes given in the manual segmentation images. It is apparent that the classification obtained from the GradClassLeg is most consistent with the manual segmentation. The Wells algorithm classifies most of the image as white matter, while the BMAP algorithm classifies most of the image as gray matter.

## 4 Conclusions

We propose a parametric gradient descent illumination inhomogeneity correction algorithm GradClassLeg, which assumes as the illumination bias model a linear combination of 2D products of Legendre polynomials. The GradClassLeg has been tested and compared exhaustively with two other algorithms found in the literature, BMAP and the Wells' algorithm. The GCL systematically improves the classification results on a set of MRI images downloaded from the IBSR. We are currently tuning the 3D implementation of the algorithm in order to

apply it as a competitive 3D analysis tool. This approach taken to define the algorithm is not restricted to the processing of MRI data. It can be extended to the illumination correction of other images, like faces, as a preprocessing for recognition processes.

## References

1. Aarts E., J. Korst (1989) Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing, John Wiley & Sons
2. Geman, D.; Geman, S.; Graffigne, C.; Dong, P. (1990) Boundary detection by constrained optimization, *IEEE Trans. Patt. Anal. Mach. Int.* 12(7):609 - 628
3. Geman, D.; Reynolds, G. (1992) Constrained restoration and the recovery of discontinuities, *IEEE Trans. Patt. Anal. Mach. Int.* 14(3) 367 - 383
4. Internet Brain Segmentation Repository <http://www.cma.mgh.harvard.edu/ibsr/>
5. Rajapakse J. C., F. Kruggel, (1998) Segmentation of MR images with intensity inhomogeneities, *Image and Vision Comp.*, 16(3):165-180.
6. H. Rusinek et al. (1991) Alzheimer disease: measuring loss of cerebral gray matter with MR imaging, *Radiology* 178:109-114
7. Styner M., G. Gerig, C. Brechbühler, G. Szekely, (2000) Parametric estimate of intensity inhomogeneities applied to MRI. *IEEE Trans. Med. Imag.* 19(3):153-165.
8. Wells III W.M., W.E.L. Grimson, R. Kikinis, F.A. Jolez, (1996) Adaptive Segmentation of MRI Data *IEEE Trans. Med. Imag.*, 15:429-442 .

# Mutifractal Analysis of Electroencephalogram Time Series in Humans

In-Ho Song<sup>1,2</sup>, Sang-Min Lee<sup>3</sup>, In-Young Kim<sup>2</sup>, Doo-Soo Lee<sup>1</sup>, and Sun I. Kim<sup>2</sup>

<sup>1</sup> Department of Electrical and Computer Engineering, Hanyang University

<sup>2</sup> Department of Biomedical Engineering, College of Medicine, Hanyang University Haengdang-dong, Seongdong-ku, Seoul, 133-791, Korea

<sup>3</sup> Division of Bionics & Bioinformatics, College of Engineering, Chonbuk National University, Deokjin-dong, Jenju, 561-756, Korea

sunkim@hanyang.ac.kr

<http://bme.hanyang.ac.kr>

**Abstract.** By analyzing electroencephalograms taken from healthy subjects and epilepsy patients, we investigated whether the complexity of the electroencephalogram (EEG) could be characterized by a multifractal. Our results showed that the EEGs from the two sets exhibit higher complexity than monofractal  $1/f$  scaling. A significant finding was the observation that the dynamics of the epileptic EEGs exhibited anticorrelated, correlated, and uncorrelated behaviors. In conclusion, multifractal formalism based on the wavelet transform modulus maxima (WTMM) may be a good tool to characterize the various dynamics of the two sets.

## 1 Introduction

In recent years, neuroscientists have reported that the brain has multiple feedback loops and that numerous components of the brain interact mutually [1]. Researchers have also begun to recognize that electroencephalogram (EEG) signals stem from a highly nonlinear system [1] [2] [3]. Two different approaches for EEG analysis have used. One is a linear approach, in which researchers view an EEG's behavior as linearly correlated noise. The other is a nonlinear approach, in which researchers have suggested nonlinear mechanisms in the EEG. However, much disputation between the two approaches is still occurring. Recent research has suggested that although the EEG exhibits short-term information loss such as random walk processes, the EEG could be characterized by long-range power law correlations that indicate time scale invariance and fractal structure. Monofractal signals have linear properties and homogeneity [4]. However, in general, many physiological time series fluctuate in a complex manner and have inhomogeneity, suggesting that different parts of the signal have different scaling properties [5]. The aim of the present study is (i) to elucidate whether the EEG exhibits higher complexity than fractal  $1/f$  scaling-characterized by a multifractal spectrum, (ii) to assess whether the EEG has nonlinear deterministic structures and (iii) to investigate whether pathological brain states affect the phenomenon of multifractality in the EEG dynamics. The scaling features of the EEG time series were



investigated by means of the wavelet transform modulus maxima (WTMM). We compared the multifractal properties of EEG signals obtained during epileptic seizures and in healthy subjects in a relaxed state with their eyes open.

## 2 Methods and Materials

### 2.1 WTMM-Based Multifractal Formalism

The wavelet transform of a signal  $f(t)$  is defined as

$$T_\psi[f](x_0, a) = \frac{1}{a} \int_{-\infty}^{\infty} f(x) \psi\left(\frac{x - x_0}{a}\right) dx \quad (1)$$

where  $x_0$  is the position parameter,  $a$  is the scale parameter, and  $\psi(t)$  is the mother wavelet function [6]. We used a third derivative Gaussian function as the analyzing wavelet. A partition function  $Z_q(a)$  was defined as the sum of the power of order  $q$  of the local maxima of  $|T_\psi[f](x_0, a)|$  at scale  $a$  [6]. For small scales, it was expressed as

$$Z_q(a) \sim a^{\tau(q)} \quad (2)$$

where  $\tau(q)$  is the scaling exponent. For monofractal signals,  $\tau(q)$  is a linear function:  $\tau(q) = qh(q) - 1$ , where  $h(q) = d\tau(q)/dq = \text{constant}$ , and is the global Hurst exponent [4] [5]. For multifractal signals,  $\tau(q)$  is a nonlinear function and  $h(q) = d\tau(q)/dq = \text{not constant}$  [4] [5]. For positive values of  $q$ ,  $Z_q(a)$  characterizes the scaling of the large fluctuations and strong singularities, whereas for negative values of  $q$ ,  $Z_q(a)$  characterizes the scaling of the small fluctuations and weak singularities. The singularity spectrum  $D(h)$  can be expressed using a Legendre transform [4] [5] [6].

$$D(h) = q \frac{d\tau(q)}{dq} - \tau(q) \quad (3)$$

$D(h)$  can quantify the statistical properties of the different subsets characterized by different exponents,  $h$ . Nonzero  $D(h)$  and  $h = 0.5$  imply that the fluctuations in signal exhibit uncorrelated behavior. Values of  $h$  in the range  $0 < h < 0.5$  imply anticorrelated behavior, while  $h > 0.5$  corresponds to correlated behavior [5].

### 2.2 Surrogate Time Series and Statistical Analysis

To assess the presence of nonlinearity in the time series, an appropriate null hypothesis is that the original time series arise from a Gaussian linear process measured through a static monotonic, possibly nonlinear function [7]. In this study, the iterative amplitude adjusted Fourier transform (iAAFT) was used for each original time series. The surrogate time series generated by the iAAFT preserve the Fourier amplitudes of the original time series but randomize the phase [7]. A paired-sample t-test was used to compare the mean of each set between the original data and the surrogate data. The significant level was 0.05.

### 2.3 Data

We used two sets of EEG data files [8]. One set A consists of 100 segments taken from five healthy subjects' scalps using a standardized electrode location scheme. The subjects were relaxed, awake, and with their eyes open. The other set E consists of 100 segments obtained from five patients during epileptic seizures. Set E was recorded intracranially. Each segment of 23.6 s duration was selected and cut out from continuous multichannel EEG recordings after visual inspection for artifacts and was chosen to satisfy a criterion of weak stationarity. All EEG signals were sampled at 173.61 Hz, 12 bits/sample. Band-pass filter settings were 0.53-40 Hz (12 dB/octave). All recording parameters were fixed except different recording electrodes were used for extracranial and intracranial EEG registration [8]. In this study, 30 segments from set A and 30 segments from set E were randomly selected. The set of 30 segments selected from set A was denoted as AA. Similarly, the set of 30 segments selected from set E was denoted as EE.

## 3 Results

Singularity spectra were computed using the WTMM for the two sets and the surrogate data. The shape of  $D(h)$  for the set AA was broad. The range of local Hurst exponents with  $D(h)$  greater than 0.8 was  $0 < h < 0.5$ . The shape of  $D(h)$  for the surrogate data was also broad. For the surrogate data, local Hurst exponents in the range  $0 < h < 0.5$  corresponded to  $D(h)$  greater than 0.8. The average values of local Hurst exponents with maximum  $D(h)$  for the set AA and for the surrogate data were 0.232 and 0.222, respectively. Fig. 1 shows the multifractal spectra for the sets AA and the surrogate data.

The shape of  $D(h)$  for the set EE was also broad. The range of local Hurst exponents with  $D(h)$  greater than 0.6 was  $0.25 < h < 1.0$ . For the surrogate

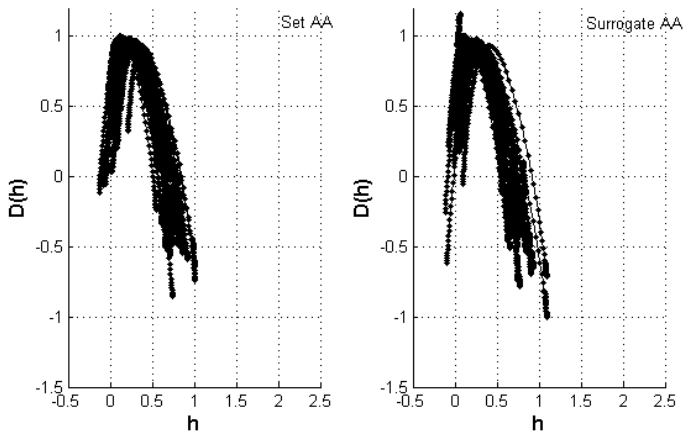
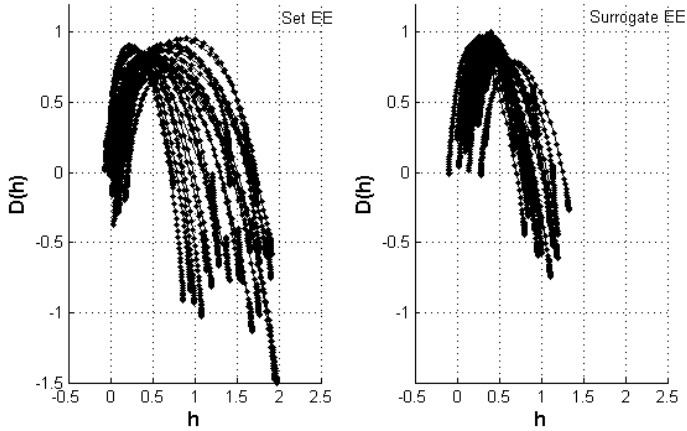
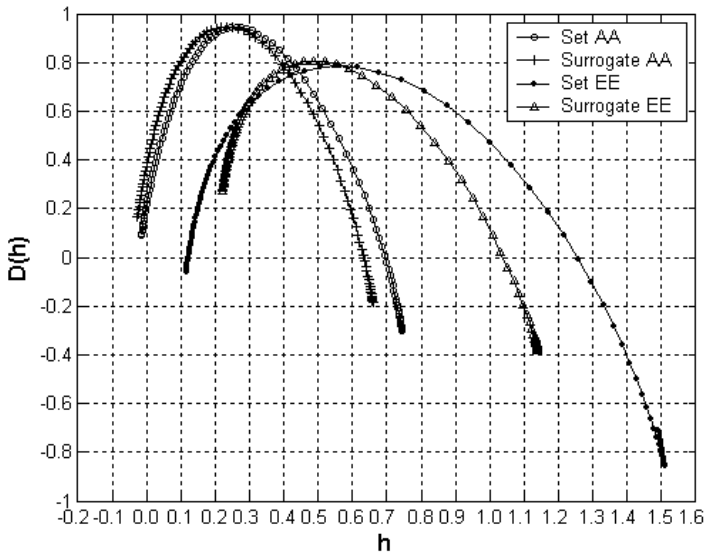


Fig. 1. Singularity spectra for the set AA and the surrogate data



**Fig. 2.** Singularity spectra for the set EE and the surrogate data.  $D(h)$  is obtained as the Legendre transform of  $\tau(q)$



**Fig. 3.** Comparison of singularity spectra for the two sets and the surrogate data

data of the set EE, the shape of  $D(h)$  was broad but was diminished in size. The range of local Hurst exponents with  $D(h)$  greater than 0.6 was reduced to  $0.25 < h < 0.75$  for the surrogate data. The average values of local Hurst exponents with maximum  $D(h)$  for the set EE and for the surrogate data were 0.552 and 0.414, respectively. Fig. 2 shows the singularity spectra for the set EE and the surrogate data. Fig. 3 shows that the singularity spectra for the two sets and the surrogate data.

## 4 Discussion and Conclusion

As shown in Fig. 3, the shapes of  $D(h)$  for the two sets were broad, indicating that the EEG could not be characterized as monofractal, but was multifractal. We suggest that the EEGs from the two sets have higher complexity and inhomogeneity. We found that the subsets characterized by local Hurst exponents of  $0 < h < 0.5$  were statistically dominant for the set AA. Therefore, the dynamics of the EEGs taken at rest with eyes open exhibit anticorrelated behavior. This means that large values are more likely to be followed by small values and vice versa [9]. The results observed after the surrogate test showed no significant change of the shape of  $D(h)$  and no significant change of the value of the local Hurst exponents with maximum  $D(h)$ . This means that the EEGs taken during rest with eyes open have a quasilinear structure. These results also are consistent with previous studies [8] [10]. For the set EE, statistically dominant subsets were characterized by local Hurst exponents of  $0.25 < h < 1.0$ . We found that the dynamics of the epileptic EEGs exhibited anticorrelated, correlated, and uncorrelated behaviors. This finding indicates that the epileptic EEG has more inhomogeneity than EEGs taken at rest with eyes open. Moreover, the average value of local Hurst exponents with maximum  $D(h)$  for the sets EE and AA were 0.552 and 0.232, respectively. From the dominant local Hurst exponents and the local Hurst exponents with maximum  $D(h)$ , we found that multifractality could quantify the complex dynamics from different physiological and pathological brain states. From the results of the surrogate data for the set EE, we found that the local Hurst exponents for the dominant subsets were significantly changed to  $0.25 < h < 0.75$ . In addition, we found that the average value of local Hurst exponents with maximum  $D(h)$  was significantly changed from 0.552 to 0.414 ( $p < 0.001$ ). The surrogate data, however, remained a multifractal process. These findings indicate that epileptic EEGs have significant nonlinear deterministic structures. This finding is in accord with other studies [8] [10]. In summary, we investigated the possibility that the EEGs from the two sets exhibit higher complexity than  $1/f$  scaling. Our results showed the existence of inhomogeneity in the two sets. We found that the EEGs taken during rest with the eyes open exhibited anticorrelated behavior and had linear structures. Of significance was the finding that the epileptic EEGs exhibited anticorrelated, correlated, and uncorrelated behaviors. Our finding indicated that epileptic EEGs had nonlinear deterministic structures rather than linear structures. Multifractality based on the WTMM allowed the various dynamics of the two sets to be revealed and characterized.

*Acknowledgements.* This study was supported by a grant of the project “Development of the Core Technology of Medical Devices for Elderly”, Ministry of Science and Technology (MOST), Republic of Korea.

## References

1. Fell, J., Röschke, J., Mann, K., Schäffner, C.: Discrimination of sleep stages: a comparison between spectral and nonlinear EEG measures. *Electroencephalogr. Clin. Neurophysio.* **98** (1996) 401–410
2. Thomasson, N., Hoepfner, T.J., Webber Jr, C.L., Zbilut, J.P.: Recurrence quantification in epileptic EEGs. *Phys. Lett. A.* **279** (2001) 94–101
3. Song, I.H., Lee, D.S., Kim, Sun.I.: Recurrence quantification analysis of sleep electroencephalogram in sleep apnea syndrome in humans. *Neurosci. Lett.* **366** (2004) 148–153
4. Ivanov, P.Ch., Amaral, Luís A.N., Goldberger, A.L., Havlin, S., Rosenblum, M.G., Struzik, Z.R., Stanley, H.E.: Multifractality in human heartbeat dynamics. *Nature.* **399** (1999) 461–465
5. Ivanov, P.Ch., Amaral, Luís A.N., Goldberger, A.L., Havlin, S., Rosenblum, M.G., Stanley, H.E., Struzik, Z.R.: From  $1/f$  noise to multifractal cascades in heartbeat dynamics. *Chaos*, **11** (2001) 641–652
6. Muzy, J.F., Bacry, E., Arneodo, A.: The multifractal formalism revisited with wavelet. *Int. J. Bifurc. chaos.* **4** (1994) 245–302
7. Schreiber, T., Schmitz, A.: Surrogate time series. *Physica D* **142** (2000) 346–382
8. Andrzejak, R.G., Lehnertz, K., Mormann, F., Reike, C., David, P., Elger, C.E.: Indications of nonlinear deterministic and finite-dimensional structures in time series of the electrical activity: Dependence on recording region and brain state. *Phys. Rev. E.* **64** (2001) 061907-1–061907-8
9. Beran, J.: *Statistics for Long-Memory Processes*. New York: Chapman & Hall. (1994)
10. Pereda, E., Gamundi, A., Rial, R., González, J.: Non-linear behaviour of human EEG: fractal exponent versus correlation dimension in awake and sleep stages. *Neurosci. Lett.* **250** (1998) 91–94

# Face Recognition with Improved Pairwise Coupling Support Vector Machines<sup>\*</sup>

Huaqing Li, Feihu Qi, and Shaoyu Wang

Department of Computer Science and Engineering,  
Shanghai Jiao Tong University, Shanghai 200030, P.R. China  
waking\_lee@cs.sjtu.edu.cn

**Abstract.** When dealing with multi-class classification tasks, a popular and applicable way is to decompose the original problem into a set of binary subproblems. The most well-known decomposition strategy is *one-against-one* and the corresponding widely-used method to recombine the outputs of all binary classifiers is *pairwise coupling* (PWC). However PWC has an intrinsic shortcoming; many meaningless partial classification results contribute to the global prediction result. In this paper, this problem is tackled by the use of *correcting classifiers*. A novel algorithm is proposed which works in two steps: First the original *pairwise probabilities* are converted into a new set of *pairwise probabilities*, then *pairwise coupling* is employed to construct the global posterior probabilities. This algorithm is applied to face recognition on the ORL face database, experimental results show that it is effective and efficient.

## 1 Introduction

Multi-class classification is a common task in many real life problems such as face recognition and speech recognition. Two kinds of algorithms are involved; algorithms of the first kind take all classes into consideration at once [1], while algorithms of the second kind decompose the original problem into a set of binary subproblems and construct a multi-class classifier by combing all corresponding binary classifiers [2, 3]. Due to their good scalability and less computational complexity, algorithms of the second kind are preferred by most researchers [3].

Among various decomposition strategies proposed, *one-against-one* is the most popular one. It trains one classifier for each pair of classes, ignoring the remaining ones. Hence for a  $k$ -class problem,  $\frac{k(k-1)}{2}$  binary classifiers need to be trained. In prediction, the outputs of all binary classifiers must be recombined to construct the global result. A simple combining algorithm is *Max-Voting*, which assigns a test example to the class with the most winning two-class decisions [5]. For binary classifiers with probabilistic outputs, a more sophisticated algorithm, called *pairwise coupling* (PWC), can be used [6]. PWC couples all *pairwise*

---

<sup>\*</sup> This work is supported by the National Natural Science Foundation of China under grant No.60072029 and No.60271033.

*probabilities*, i.e. the probabilistic outputs of all binary classifiers, into a set of posterior probabilities. Then the test example is assigned to the class with the max posterior probability. However PWC has a shortcoming: When a test example does not belong to either class related to a binary classifier, the prediction result of this classifier is meaningless and can damage the global result.

In [4], Moreira and Mayoraz proposed an algorithm, called PWC-CC, to solve this problem, where CC stands for *correcting classifier*. A *correcting classifier* is the binary classifier trained to distinguish a pair of classes from the remaining ones. Experimental results showed that PWC-CC has much better performance than PWC. However the original PWC-CC is quite simple and has some disadvantages. In this paper, we propose a novel PWC-CC algorithm to overcome these disadvantages. The new algorithm works in two steps: First the original *pairwise probabilities* are converted into a new set of *pairwise probabilities*, then *pairwise coupling* is employed to form the global posterior probabilities. Experimental results show that our algorithm is effective and can achieve even better performance. The rest of the paper is organized as follows: In Section 2, we briefly review some related works. Section 3 describes the proposed algorithm. Experimental results and corresponding analysis are presented in Section 4. Finally, Section 5 concludes the paper.

## 2 Related Works

### 2.1 Pairwise Coupling Methods

Take for example we are dealing with a classification task involving  $k$  classes  $w_i$ ,  $1 \leq i \leq k$ ,  $k \geq 3$ . Then there are  $\frac{k(k-1)}{2}$  binary classifiers. Suppose classifier  $C_{ij}$ ,  $i < j$ , is trained to separate class  $w_i$  from class  $w_j$ . Given a test example  $x$ , the output of  $C_{ij}$  is a probability  $r_{ij} = \text{Prob}(x \in w_i | x, x \in w_i \text{ or } w_j)$ , called *pairwise probability*. Obviously  $r_{ji} = 1 - r_{ij}$  holds. To couple all the *pairwise probabilities* into a common set of posterior probabilities  $p_i = \text{Prob}(x \in w_i | x)$ , several PWC methods have been proposed.

**Method by Hastie and Tibshirani.** In [6], Hastie and Tibshirani introduced a new set of auxiliary variables  $\mu_{ij}$ :

$$\mu_{ij} = \frac{p_i}{p_i + p_j} \tag{1}$$

and aimed at finding  $p_i$  so that the corresponding  $\mu_{ij}$  are in some sense “close” to the observed  $r_{ij}$ . In their work, the Kullback-Leibler distance between  $r_{ij}$  and  $\mu_{ij}$

$$l(\mathbf{P}) = \sum_{i < j} n_{ij} \left[ r_{ij} \log \frac{r_{ij}}{\mu_{ij}} + (1 - r_{ij}) \log \frac{1 - r_{ij}}{1 - \mu_{ij}} \right] \tag{2}$$

is selected as the closeness measurement, where  $n_{ij}$  are the weights. They pointed out that if  $n_{ij}$  are considered equal, which is reasonable when the multi-class data are balanced,  $\mathbf{P}$  satisfies

$$p_i > p_j \iff (\tilde{p}_i = \frac{2 \sum_{s:s \neq i} r_{is}}{k(k-1)}) > (\tilde{p}_j = \frac{2 \sum_{s:s \neq j} r_{js}}{k(k-1)}) . \tag{3}$$

Therefore  $\tilde{\mathbf{P}}$  is sufficient if one only requires a classification rule such as

$$\arg \max_{1 \leq i \leq k} \tilde{p}_i . \tag{4}$$

Furthermore,  $\tilde{\mathbf{P}}$  can be derived as an approximation to the identity

$$p_i = \sum_{j:j \neq i} \left( \frac{p_i + p_j}{k-1} \right) \left( \frac{p_i}{p_i + p_j} \right) = \sum_{j:j \neq i} \left( \frac{p_i + p_j}{k-1} \right) \mu_{ij} , \tag{5}$$

by replacing  $p_i + p_j$  with  $\frac{k}{2}$  and  $\mu_{ij}$  with  $r_{ij}$  in (3). Thereby the differences between  $p_i$  are underestimated, which causes the method to be instable when dealing with unbalanced probabilities.

**Method by Wu, Lin, and Weng.** In [7], Wu et al. proposed another PWC method. They found the optimal  $\mathbf{P}$  through solving the following optimization problem:

$$\begin{aligned} \min \quad & \sum_{i=1}^k \sum_{j:j \neq i} (r_{ji}p_i - r_{ij}p_j)^2 , \\ \text{s.t.} \quad & \sum_{i=1}^k p_i = 1 , \quad p_i \geq 0 \quad \forall i . \end{aligned} \tag{6}$$

Note that (6) can be reformulated as

$$\min \quad 2\mathbf{P}^T \mathbf{Q} \mathbf{P} \quad \equiv \quad \min \frac{1}{2} \mathbf{P}^T \mathbf{Q} \mathbf{P} , \tag{7}$$

where

$$Q_{ij} = \begin{cases} \sum_{s:s \neq i} r_{si}^2 & \text{if } i = j , \\ -r_{ji}r_{ij} & \text{if } i \neq j . \end{cases} \tag{8}$$

Then  $\mathbf{P}$  can be obtained by solving the following linear system:

$$\begin{bmatrix} \mathbf{Q} & \mathbf{e} \\ \mathbf{e}^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{P} \\ b \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ 1 \end{bmatrix} . \tag{9}$$

This method is easy to implement and has a more stable performance.

### 2.2 Improved Pairwise Coupling with Correcting Classifiers

If a test example  $x$  is classified by classifier  $C_{ij}$ , while  $x$  belongs to neither class  $w_i$  nor class  $w_j$ , the output of  $C_{ij}$ , i.e.  $r_{ij}$ , is meaningless. Consequently considering  $r_{ij}$  in forming the posterior probabilities  $\mathbf{P}$  ( $\tilde{\mathbf{P}}$ ) will bring in nonsense and can damage the quality of  $\mathbf{P}$  ( $\tilde{\mathbf{P}}$ ).



To tackle this problem, Moreira and Mayoraz proposed an algorithm called PWC-CC [4]. This algorithm trains  $\frac{k(k-1)}{2}$  additional binary classifiers  $CC_{ij}$ , called *correcting classifiers*, to distinguish classes  $w_i$  and  $w_j$  from the remaining ones. For a given example  $x$ , the probabilistic output of  $CC_{ij}$  is  $q_{ij} = \text{Prob}(x \in w_i \text{ or } w_j | x)$ . Obviously  $q_{ij} = q_{ji}$  holds. Then  $\tilde{\mathbf{P}}$  is computed using the following formula instead of (3):

$$\tilde{p}_i = \frac{2 \sum_{s:s \neq i} r_{is} \cdot q_{is}}{k(k-1)} . \tag{10}$$

If an example  $x$  belongs to neither class  $w_i$  nor class  $w_j$ , the output of classifier  $CC_{ij}$ , i.e.  $q_{ij}$ , is expected to be small (close to 0). Otherwise,  $q_{ij}$  is expected to be large (close to 1). Thus by using formula (10), the impact of those meaningless  $r_{ij}$  are largely weakened and the accuracy of the global prediction is improved.

### 3 Our Method

#### 3.1 Motivation

Let us analyze Moreira and Mayoraz’s algorithm in a more detailed way. First, we divide formula (10) into two formulas.

$$r_{ij}^i = r_{ij} \cdot q_{ij} , \tag{11}$$

$$\tilde{p}_i = \frac{2 \sum_{s:s \neq i} r_{is}^i}{k(k-1)} , \tag{12}$$

Note that (12) has exactly the same form as (3).

Then we immediately get

$$r_{ji}^i = q_{ij} - r_{ij}^i \neq 1 - r_{ij}^i . \tag{13}$$

and

$$\sum_{i=1}^k \tilde{p}_i \neq 1 . \tag{14}$$

Formula (13) indicates that  $r_{ij}^i$  obtained by (11) are not real *pairwise probabilities*. Thereby sophisticated PWC method, e.g. the one described by (6), can not be applied to them. Formula (14) indicates that  $\tilde{p}_i$  obtained by (12) are not posterior probabilities and the interpretation of  $\tilde{\mathbf{P}}$  becomes ambiguous. We consider these the disadvantages of the original PWC-CC method.

#### 3.2 Analysis

The purpose of using *correcting classifiers* is to reduce the impact those meaningless  $r_{ij}$  have on the global  $\mathbf{P}$  ( $\tilde{\mathbf{P}}$ ). The original PWC-CC method achieves this purpose by weighting  $r_{ij}$  with corresponding  $q_{ij}$ . Thus the *values* of the

$$\begin{aligned}
 q_{ij} &: \begin{bmatrix} - & 0.90 & 0.70 \\ 0.90 & - & 0.40 \\ 0.70 & 0.40 & - \end{bmatrix} & p_i &: \begin{bmatrix} 0.58 \\ 0.30 \\ 0.12 \end{bmatrix} \\
 \text{PWC} : r_{ij} &: \begin{bmatrix} - & 0.60 & 0.60 \\ 0.40 & - & 0.90 \\ 0.40 & 0.10 & - \end{bmatrix} & \implies \tilde{p}_i &: \begin{bmatrix} 0.40 \\ 0.43 \\ 0.17 \end{bmatrix} \\
 \text{PWC} - \text{CC} : r'_{ij} &: \begin{bmatrix} - & 0.54 & 0.42 \\ 0.36 & - & 0.36 \\ 0.28 & 0.04 & - \end{bmatrix} & \implies \tilde{p}_i^O &: \begin{bmatrix} 0.32 \\ 0.24 \\ 0.11 \end{bmatrix} \\
 \text{NPWC} - \text{CC} : t_{ij} &: \begin{bmatrix} - & 0.83 & 0.83 \\ 0.17 & - & 0.66 \\ 0.17 & 0.34 & - \end{bmatrix} & \implies \tilde{p}_i^N &: \begin{bmatrix} 0.55 \\ 0.28 \\ 0.17 \end{bmatrix}
 \end{aligned}$$

**Fig. 1.** Comparison of NPWC-CC, PWC-CC and PWC on a 3-class problem.  $p_i$  are the real probabilities of a test example  $x$ .  $\tilde{p}_i$ ,  $\tilde{p}_i^O$ , and  $\tilde{p}_i^N$  are estimated by formula (3). Both PWC-CC and NPWC-CC classifies  $x$  correctly, while plain PWC behaves wrong. Note that  $\tilde{p}_i^N$  is closest to  $p_i$

meaningless  $r_{ij}$  are likely to be decreased, while those meaningful ones are kept nearly unchanged.

There are other ways to achieve the same purpose. One approach is to reduce the *confidence* of meaningless  $r_{ij}$  and enhance the *confidence* of meaningful  $r_{ij}$ . This is based on the following observation: When making a global decision in real life, those who have more confidence in their opinions dominate those who have less confidence in their opinions. We believe the binary classifiers behave in a similar way. A classifier  $C_{ij}$  is considered to have much confidence in its output if the corresponding  $r_{ij}$  is very large (close to 1) or very small (close to 0). On the contrary, a  $r_{ij}$  around 0.5 indicates that  $C_{ij}$  is not that confident in determining which class wins.

### 3.3 Algorithm

Based on the previous analysis, a novel PWC-CC (NPWC-CC) algorithm is proposed. NPWC-CC works in two steps:

1.  $r_{ij}$  are converted into a new set of *pairwise probabilities*  $t_{ij}$ .

$$t_{ij} = \begin{cases} \frac{1-\Delta'}{2} & r_{ij} \leq 0.5 \\ \frac{1+\Delta'}{2} & r_{ij} > 0.5 \end{cases}, \tag{15}$$

where

$$\Delta' = \begin{cases} \tanh(4\Delta) & q_{ij} \geq 0.5 \\ q_{ij}\Delta & q_{ij} < 0.5 \end{cases}, \tag{16}$$

$$\Delta = |r_{ij} - r_{ji}| = |2r_{ij} - 1|. \tag{17}$$

2. Then a PWC method is employed to couple  $t_{ij}$  into a global  $\mathbf{P}$  ( $\tilde{\mathbf{P}}$ ).

In the first step, those meaningless  $r_{ij}$ , with expected small  $q_{ij}$ , are likely to be made more unconfident (corresponding  $t_{ij}$  are closer to 0.5). On the other hand, the confidences of the meaningful  $r_{ij}$  are strengthened (corresponding  $t_{ij}$  are farther from 0.5). An comparison of NPWC-CC, PWC-CC and PWC is illustrated in Figure 1.

From (15) (16) and (17), we immediately get  $t_{ij} = 1 - t_{ji}$ . This means that  $t_{ij}$  are real *pairwise probabilities*. Therefore any PWC method can be employed in NPWC-CC and the obtained  $p_i$  ( $\tilde{p}_i$ ) are meaningful posterior probabilities. Thus the disadvantages of PWC-CC are overcame.

### 4 Experimental Results

The proposed algorithm is applied to face recognition on the ORL face database. The database contains 10 different images of 40 distinct subjects. For some of the subjects, the images were taken at different times, varying lighting slightly, facial expressions (open/closed eyes, smiling/non-smiling) and facial details (glasses/no-glasses). All the images were taken against a dark homogeneous background and the subjects are in up-right, frontal position (with tolerance for some side movement). The original resolution of the images was  $92 \times 112$ , 8-bit grey levels. Some example faces are show in Figure 2. In our experiments, the ORL database is randomly divided into two sets with equal size. Five images of each subject are used for training, the rest five are used for testing. Feature extraction is done with the Eigenface algorithm.

Support vector machines (SVMs) are employed as binary classifiers to learn each binary subproblem. However standard SVMs do not produce probabilistic outputs. In [9], Platt suggested to map original SVM outputs to probabilities by fitting a sigmoid after the SVM:

$$P(y = 1|x) = \frac{1}{1 + \exp(Af(x) + B)} \quad (18)$$



**Fig. 2.** Example faces of four subjects from ORL, each row corresponds to one subject

Parameters  $A$  and  $B$  are found by minimizing the negative log likelihood of the training data:

$$\min - \sum_{i=1}^l t_i \log(p_i) + (1 - t_i) \log(1 - p_i) , \tag{19}$$

where

$$p_i = \frac{1}{1 + \exp(Af(x_i) + B)} , \quad t_i = \frac{y_i + 1}{2} . \tag{20}$$

$y_i$  is the target label of example  $x_i$ ,  $f(\cdot)$  is an SVM.

LIBSVM [10] is employed for SVMs training and testing. Experimental results are listed in Table 1. Where PWC1 is the PWC method described by formula (3), PWC2 is the method described by (6), NPWC-CC1 employs PWC1 in its second step, NPWC-CC2 employs PWC2 as its coupling method. Note that, NPWC-CC1 and PWC-CC differ only in the way they change the original *pairwise probabilities*.

**Table 1.** Face recognition performance of various algorithms

| #eigenfaces | PWC1  | PWC2  | PWC-CC | NPWC-CC1 | NPWC-CC2 |
|-------------|-------|-------|--------|----------|----------|
| 20          | 94.5% | 94.5% | 95.5%  | 95.5%    | 95.5%    |
| 40          | 94.5% | 95%   | 95.5%  | 95.5%    | 96%      |
| 60          | 95%   | 95%   | 96.5%  | 96.5%    | 97%      |
| 80          | 93.5% | 94%   | 94.5%  | 94.5%    | 95.5%    |
| 100         | 95%   | 95.5% | 96%    | 96%      | 96%      |

From Table 1, we can see that the use of *correcting classifiers* greatly improves the recognition accuracy. As expected, PWC-CC and NPWC-CC1 perform exactly the same, which verifies the analysis in Section 3.2 that the impact of meaningless  $r_{ij}$  can be reduced by making them more unconfident. NPWC-CC2 performs best on all cases. This highlights the virtue of NPWC-CC that more sophisticated coupling method can be employed, which is impossible for PWC-CC. In [7], it was concluded that PWC2 has a more stable performance than PWC1. Our experimental results give the same result.

## 5 Conclusion

*Pairwise coupling* (PWC) is a widely-used algorithm in multi-class classification tasks. But it has an important drawback, due to the nonsense introduced by those meaningless *pairwise probabilities*. PWC-CC tackles this problem by weighting the *pairwise probabilities* with the outputs of additional *correcting classifiers*. Though PWC-CC performs much better than PWC, it has its own

disadvantages. In this paper, a novel PWC-CC (NPWC-CC) method is proposed. NPWC-CC works in two steps: First the original *pairwise probabilities* are converted into a new set of *pairwise probabilities*, wherein those meaningless probabilities are made more unconfident, while the confidences of the meaningful ones are strengthened. Then a PWC method is employed to couple the new *pairwise probabilities* into global posterior probabilities. NPWC-CC overcomes the disadvantages of PWC-CC and can achieve even better performance.

## References

1. Weston, J., Watkins, C.: Support Vector Machines for Multi-Class Pattern Recognition. In: Proc. of 7th European Sympo. on Artificial Neural Networks (1999) 219-224
2. Platt, J., Cristianini, N., Shawe-Taylor, J.: Large Margin DAGs for Multiclass Classification. *Advance in Neural Information Processing Systems*. **12** (2000) 547-553
3. Hsu, C.-W., Lin, C.-J.: A comparison of methods for multi-class support vector machines. *IEEE Trans. on Neural Networks*. **13** (2002) 415-425
4. Moreira, M., Mayoraz, E.: Improved Pairwise Coupling Classification with Correcting Classifiers. In: Proc. of the 10th European Conf. on Machine Learning. (1998) 160-171
5. Friedman, J.: Another Approach to Polychotomous Classification. Technical report, Stanford University. (1996)
6. Hastie, T., Tibshirani, R.: Classification by Pairwise Coupling. *The Annals of Statistics*. **26** (1998) 451-471
7. Wu, T.-F., Lin, C.-J., Weng, R.C.: Probability Estimates for Multi-Class Classification by Pairwise Coupling. *Journal of Machine Learning Research*. **5** (2004) 975-1005
8. Michie, D., Spiegelhalter, D.J., Taylor, C.C.: *Machine Learning, Neural and Statistical Classification*. Ellis Horwood, London. (1994) Data available at <ftp://ftp.ncc.up.pt/pub/statlog>
9. Platt, J.: Probabilistic Outputs for Support Vector Machines and Comparison to Regularized Likelihood Methods. In: Smola, A.J., Bartlett, P.L., Schölkopf, B., Schuurmans, D. (eds.): *Advances in Large Margin Classifiers*. MIT Press (2000) 61-74
10. Chang, C.-C., Lin, C.-J.: LIBSVM: A Library for Support Vector Machines. (2002) Online at <http://www.csie.ntu.edu.tw/~cjlin/papers/libsvm.pdf>

# Face Recognition System Based on PCA and Feedforward Neural Networks

Alaa Eleyan and Hasan Demirel

Department of Electrical and Electronic Engineering,  
Eastern Mediterranean University,  
Gazimağusa, North Cyprus, Mersin 10, Turkey  
{alaa.eleyan, hasan.demirel}@emu.edu.tr

**Abstract.** Face recognition is one of the most important image processing research topics which is widely used in personal identification, verification and security applications. In this paper, a face recognition system, based on the principal component analysis (PCA) and the feedforward neural network is developed. The system consists of two phases which are the PCA preprocessing phase, and the neural network classification phase. PCA is applied to calculate the feature projection vector of a given face which is then used for face identification by the feedforward neural network. The proposed PCA and neural network based identification system provides improvement on the recognition rates, when compared with a face classifier based on the PCA and Euclidean Distance.

## 1 Introduction

Much of work done in computer recognition of faces has focused on detecting individual features such as the eyes, nose, mouth, and head outline, and defining a face model by the position, size, and relationships among these features. Such approaches have proven difficult to extend to multiple views and have often been quite fragile, requiring a good initial guess to guide them. Research in human strategies of face recognition, moreover, has shown that individual features and their immediate relationships comprise an insufficient representation to account for the performance of adult human face identification [1]. Bledsoe [2,3] was the first to attempt to use semi-automated face recognition with a hybrid human-computer system that classified faces on the basis of fiducially marks entered on photographs by hand. Parameters for the classification were normalized distances and ratios among points such as eye corners, mouth corners, nose tip, and chin point. Fischler and Elschlager [4] described a linear embedding algorithm that used local feature template matching and a global measure of fit to find and measure the facial features. Generally speaking, we can say that most of the previous work on automated face recognition [5, 6] has ignored the issue of just what aspects of the face stimulus are important for face recognition. This suggests the use of an information theory approach of coding and decoding of face images, emphasizing the significant local and global features. Such features may or may not be directly related to our intuitive notion of face features such as the eyes,

nose, lips, and hair. In mathematical terms, the principal components of the distribution of faces, or the eigenvectors of the covariance matrix of the set of face images, treating an image as point (or vector) in a very high dimensional space is sought. The eigenvectors are ordered, each one accounting for a different amount of the variation among the face images. These eigenvectors can be thought of as a set of features that together characterize the variation between face images. Each image location contributes more or less to each eigenvector, so that it is possible to display these eigenvectors as a sort of ghostly face image which is called an "eigenface". Each individual face can be represented exactly in terms of a linear combination of the eigenfaces. Each face can also be approximated using only the "best" eigenfaces, those that have the largest eigenvalues, and which therefore account for the most variance within the set of face images. The best  $M$  eigenfaces span an  $M$ -dimensional subspace which we call the "face space" of all possible images. The method proposed by M. Turk and A. Pentland [7] uses a PCA based face recognition system which is called the eigenfaces method. In this method, a given face image is transformed into the eigenspace to obtain a feature projection vector. The Euclidean Distance between the projection vector of a given face and the class projection vectors are used to determine a correct or false recognition. In this paper, the projection vectors obtained through the same PCA procedure are used as the input vectors for the feedforward neural network classifier. The proposed PCA and neural network based identification system provides improvement on the recognition rates, when compared with a face classifier based on the PCA and Euclidean Distance introduced by M. Turk and A. Pineland [7].

## 2 Calculating Eigenfaces

Let a face image  $I(x,y)$  be a two-dimensional  $N \times N$  array. An image may also be considered as a vector of dimension  $N^2$ , so that a typical image of size  $112 \times 92$  becomes a vector of dimension 10,304, or equivalently a point in a 10,304-dimensional space. An ensemble of images maps to a collection of points in this huge space. Images of faces, being similar in overall configuration, will not be randomly distributed in this huge image space and thus can be described by a relatively low dimensional subspace. The main idea of the principle component is to find the vectors that best account for the distribution of face images within the entire image space. These vectors define the subspace of face images, which we call "face space". Each vector is of length  $N^2$ , describes an  $N \times N$  image, and is a linear combination of the original face images. Because these vectors are the eigenvectors of the covariance matrix corresponding to the original face images, and because they are face-like in appearance, we refer to them as "eigenfaces".

Let the training set of face images be  $\Gamma_1, \Gamma_2, \dots, \Gamma_M$  then the average of the set is defined by

$$\Psi = \frac{1}{M} \sum_{n=1}^M \Gamma_n \quad (1)$$

Each face differs from the average by the vector

$$\Phi_i = \Gamma_i - \Psi \quad (2)$$

This set of very large vectors is then subject to principal component analysis, which seeks a set of  $M$  orthonormal vectors,  $U_n$ , which best describes the distribution of the data. The  $k$ th vector,  $U_k$ , is chosen such that

$$\lambda_k = \frac{1}{M} \sum_{n=1}^M (U_k^T \Phi_n)^2 \quad (3)$$

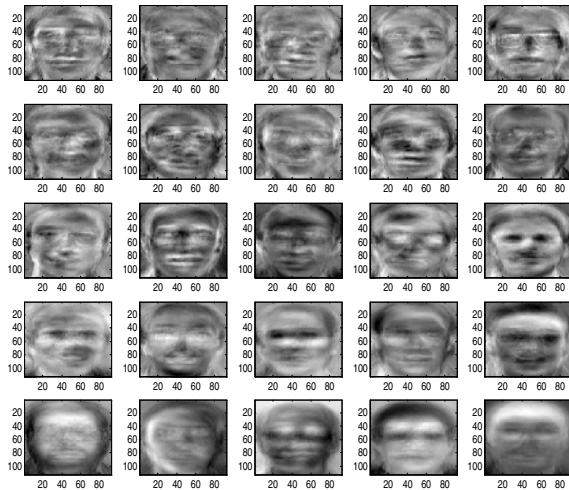
is a maximum, subject to

$$U_i^T U_k = \delta_{ik} = \begin{cases} 1, & \text{if } i = k \\ 0, & \text{otherwise} \end{cases} \quad (4)$$

The vectors  $U_k$  and scalars  $\lambda_k$  are the eigenvectors and eigenvalues, respectively of the covariance matrix

$$C = \frac{1}{M} \sum_{n=1}^M \Phi_n \Phi_n^T = AA^T \quad (5)$$

where the matrix  $A = [\Phi_1 \Phi_2 \dots \Phi_M]$ . The covariance matrix  $C$ , however is  $N^2 \times N^2$  real symmetric matrix, and determining the  $N^2$  eigenvectors and eigenvalues is an intractable task for typical image sizes. We need a computationally feasible method to find these eigenvectors.



**Fig. 1.** First 25 eigenfaces with highest eigenvalues



Consider the eigenvectors  $v_i$  of  $A^T A$  such that

$$A^T A v_i = \mu_i v_i \tag{6}$$

Premultiplying both sides by  $A$ , we have

$$A A^T A v_i = \mu_i A v_i \tag{7}$$

where we see that  $A v_i$  are the eigenvectors and  $\mu_i$  are the eigenvalues of  $C = A A^T$ .

Following these analysis, we construct the  $M \times M$  matrix  $L = A^T A$ , where  $L_{mn} = \Phi_m^T \Phi_n$ , and find the  $M$  eigenvectors,  $v_i$ , of  $L$ . These vectors determine linear combinations of the  $M$  training set face images to form the eigenfaces  $U_I$ .

$$U_I = \sum_{k=1}^M v_{Ik} \Phi_k, \quad I = 1, \dots, M \tag{8}$$

Examples of eigenfaces after applying the eigenfaces algorithm are shown in Fig.1.

With this analysis, the calculations are greatly reduced, from the order of the number of pixels in the images ( $N^2$ ) to the order of the number of images in the training set ( $M$ ). In practice, the training set of face images will be relatively small ( $M \ll N^2$ ), and the calculations become quite manageable. The associated eigenvalues allow us to rank the eigenvectors according to their usefulness in characterizing the variation among the images.

### 3 Using Eigenfaces to Classify a Face Image

The eigenface images calculated from the eigenvectors of  $L$  span a basis set with which to describe face images. Sirovich and Kirby [8, 9] evaluated a limited version of this framework on an ensemble of 115 images ( $M = 115$ ) images of Caucasian males digitized in a controlled manner, and found that *40 eigenfaces* were sufficient for a very good description of face images. In practice, a smaller  $M'$  can be sufficient for identification, since accurate reconstruction of the image is not a requirement. Based on this idea, the proposed face recognition system lets the user specify the number of eigenfaces ( $M'$ ) that is going to be used in the recognition. For maximum accuracy, the number of eigenfaces should be equal to the number of images in the training set. But, it was observed that, for a training set of fourteen face images, seven eigenfaces were enough for a sufficient description of the training set members. In this framework, identification becomes a pattern recognition task. The eigenfaces span an  $M'$  dimensional subspace of the original  $N^2$  image space. The  $M'$  significant eigenvectors of the  $L$  matrix are chosen as those with the largest associated eigenvalues.

A new face image ( $\Gamma$ ) is transformed into its eigenface components (projected onto "face space") by a simple operation,

$$w_k = U_k^T (\Gamma - \Psi) \tag{9}$$

for  $k = 1, \dots, M$ . This describes a set of point by point image multiplications and summations, operations performed at approximately frame rate on current image processing hardware, with a computational complexity. The weights form a feature vector,

$$\Omega^T = [w_1 \ w_2 \ \dots \ w_M] \quad (10)$$

that describes the contribution of each eigenface in representing the input face image, treating the eigenfaces as a basis set for face images. The feature vector is then used in a standard pattern recognition algorithm to find which of a number of predefined face classes, if any, best describes the face. The face class  $\Omega_k$  can be calculated by averaging the results of the eigenface representation over a small number of face images of each individual. Classification is performed by comparing the feature vectors of the training face images with the feature vector of the input face image. This comparison is based on the *Euclidean Distance* between the faces classes and the input face image. This is given in Eq. (11). The idea is to find the face class  $k$  that minimizes the Euclidean Distance.

$$\varepsilon_k = \|(\Omega - \Omega_k)\| \quad (11)$$

Where  $\Omega_k$  is a vector describing the  $k^{\text{th}}$  faces class.

## 4 Neural Networks for Classification

Neural networks can be trained to perform complex functions in various fields of applications including pattern recognition, identification, classification, speech, vision, and control systems.

In [10] a hybrid neural-network solution is presented which is compared with other methods. The system combines local image sampling, a self-organizing map (SOM) neural network, and a convolutional neural network.

Zhujie and Y.L. Yu [11] implemented a system to face recognition with eigenfaces and Back propagation neural network using 15 person database from Media Laboratory of MIT. In order to improve their system, Gaussian smoothing was applied where the system performance reached to 77.6%. This performance is almost the same performance with the Euclidean Distance based approach that we used for ORL Face Database of 40 persons, where half of images are used for training and the other half are used for testing (see Table 1 ).

### 4.1 Feedforward Neural Networks (FFNN)

In FFNN the neurons are organized in the form of layers. The neurons in a layer get input from the previous layer and feed their output to the next layer. In this type of networks connections to the neurons in the same or previous layers are not permitted. Fig. 2 shows the architecture of the proposed system for face classification.

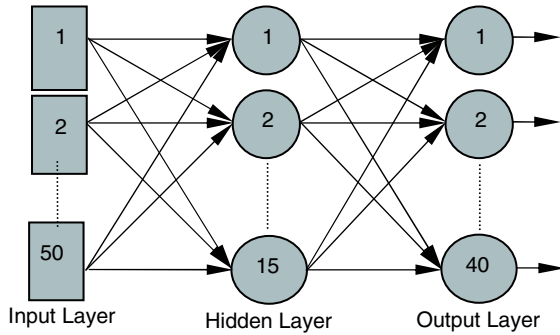


Fig. 2. Architecture of the proposed Neural Network

### 4.2 Training and Simulation of Neural Network

A large neural network for all people in the database of 40 persons was implemented. After calculating the eigenfaces, the feature projection vectors are calculated for the faces in the database. These feature projection vectors are used as inputs to train the neural network. Fig.3 shows the schematic diagram for the NN training phase.

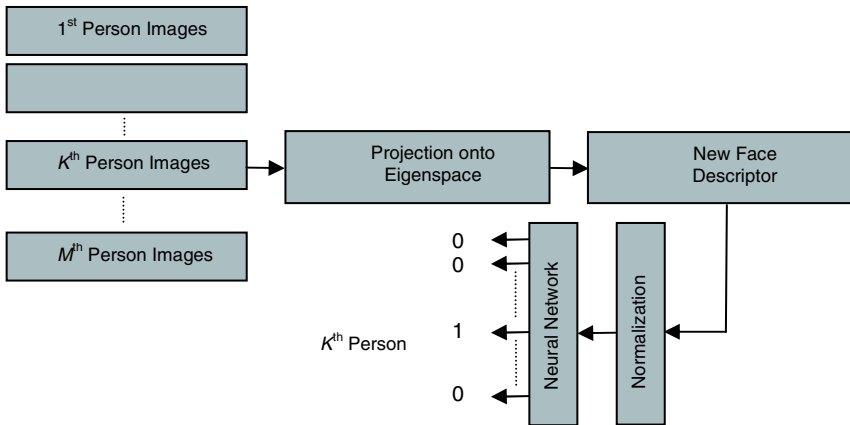


Fig. 3. Training phase of the Neural Network

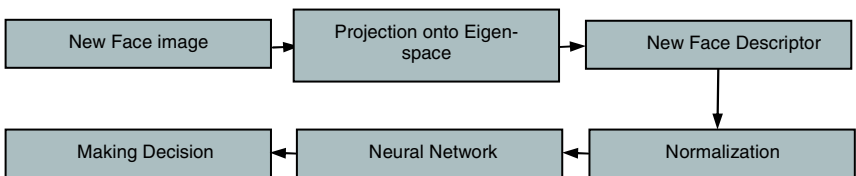


Fig. 4. Simulations of Neural Network for Classification

When a new image is considered for recognition, its feature projection vector is calculated from the eigenfaces, and this image gets its new descriptors. These descriptors are fed to the neural network and the network is simulated with these descriptors, where the network outputs are compared. By looking at the maximum output the new face is decided to belong to the class of person with this maximum output.

## 5 Experimental Results

In ORL database there are 10 different images for each 40 distinct persons. The neural network is trained with the first  $n$  poses of the 40 persons in the database. After the training the remaining  $10-n$  poses of the persons have been used for the testing. As expected the recognition performance increases with the increasing number of faces used to train the neural network (Table 1). However the real improvement is due to the use of the FFNN as the classifier instead of the Euclidean Distance based classifier. Table 1 clearly shows that, in all cases, the FFNN based face recognition system performs better than the Euclidean Distance based system.

**Table 1.** Performances of Euclidean Distance and Neural Networks

| Training Images | Testing Images | Euclidean Distance performance | Neural Networks performance |
|-----------------|----------------|--------------------------------|-----------------------------|
| 2               | 8              | 71                             | 75                          |
| 3               | 7              | 73                             | 76                          |
| 4               | 6              | 77                             | 80                          |
| 5               | 5              | 78                             | 85                          |
| 6               | 4              | 89                             | 90                          |
| 7               | 3              | 92                             | 94                          |
| 8               | 2              | 94                             | 95                          |

## 6 Conclusion

In this paper, a face recognition system, based on the PCA preprocessing followed by a FFNN based classifier is proposed. The feature projection vectors obtained through the PCA method are used as the input vectors for the training and testing of the FFNN architecture. The face recognition performance of the system using FFNN is better than the Euclidean Distance based classification system for changing number of training images. The performance of the FFNN based system with 5 training faces from each subject gives highest performance improvement which is 7% over the Euclidean Distance based system.

## References

1. Carey, S., and Diamond, R., "From Piecemeal to Configurational Representation of Faces", *Science* 195, pp. 312-313, 1977.
2. Bledsoe, W. W., "The Model Method in Facial Recognition", *Panoramic Research Inc.* Palo Alto, CA, Rep. PRI:15, August 1966.

3. Bledsoe, W. W., "Man-Machine Facial Recognition", *Panoramic Research Inc.* Palo Alto, CA, Rep. PRI:22, August 1966.
4. Fischler, M. A., and Elschlager, R. A., "The Representation and Matching of Pictorial Structures", *IEEE Trans. on Computers*, c-22.1, 1973.
5. Harmon, L. D., and Hunt, W. F., "Automatic Recognition of Human Face Profiles", *Computer Graphics and Image Processing*, Vol. 6, pp. 135-156, 1977.
6. Kaufman, G. J., and Breeding, K. J., "The Automatic Recognition of Human Faces From Profile Silhouettes", *IEEE Trans. Syst. Man Cybern.*, Vol. 6, pp. 113-120, 1976.
7. Turk, M., and Pentland, A., "Eigenfaces for Recognition", *Journal of Cognitive Neuroscience*, Vol. 3, pp. 71-86, 1991.
8. Kirby, M., and Sirovich, L., "Application of the Karhunen-Loeve Procedure for the Characterization of Human Faces", *IEEE PAMI*, Vol. 12, pp. 103-108, 1990.
9. Sirovich, L., and Kirby, M., "Low-Dimensional Procedure for the Characterization of Human Faces", *J. Opt. Soc. Am. A*, 4, 3, pp. 519-524, 1987.
10. Lawrence, S., Giles, C. L., Tsoi, A. C., Back, A. D., "Face Recognition: A Convolutional Neural-Network Approach", *IEEE Trans. Neural Networks*, Vol. 8, No. 1, January 1997.
11. Zhujie, Yu, Y. L., "Face Recognition with Eigenfaces", *Proc. of the IEEE Intl. Conf.*, pp.434 - 438, December, 1994.

# A New Fuzzy Approach for Edge Detection

Yasar Becerikli<sup>1</sup> and Tayfun M. Karan

Kocaeli University, Computer Engineering Department, Izmit, Turkey  
becer@kou.edu.tr, tmkaran@hotmail.com

**Abstract.** An edge detection is one of the most important tasks in image processing. Image segmentation, registration and identification are based on edge detection. In the literature, there is some techniques developed to achieve this task such as Sobel, Prewitt, Laplacian and Laplacian of Gaussian. In this paper, a novel knowledge-based approach which have been used to realize control techniques for past years is proposed for edge detection. Some of the classical techniques are used with certain parameters such as threshold and  $\sigma$  to implement edge detection process. The another restricts about classical approach, results generally have fixed edge thickness. The rule-based approach offers most advantages such as giving permission to adapt some parameters easily. The edges thickness can be changed easily by adding new rules or changing output parameters. That is to say rule-based approach has flexible structure which can be adapted any time or any where easily and new fuzzy approach produces nice result as well as classical techniques at least.

## 1 Introduction

An edge in an image is defined a boundary or contour where a abrupt change occurs in some physical aspect such as gray level value of an image[1]. Edge detection is one of the most important tasks in image procesing. Especially segmentation, registration, identification and recognition are based on edge detection algorithm. Many edge detection techniques are available in the literature. But most of them either have fixed result such as thickness of edges or some parameters must be selected certainly for good result such as threshold and  $\sigma$ . The fuzzy logic approach doesn't have this restricts. Simply, fixing some parameters changes the result of processing. Some of the other knowledge-based techniques that used for edge detection are based on classical techniques [4,9,11], some of them are based on training. The proposed techniques just use pure fuzzy *IF-THEN* rules and have simple structure to implement edge detection algorithm.

## 2 Detection of Image Edges

Edges in images are constituted significant gray level changing. In the other words, the edges are high frequency components of an image. That is to say a high-pass filter

---

<sup>1</sup> Corresponding Author. becer@kou.edu.tr, ybecer@ieee.org

can detect edges in an image. But working on frequency domain can load the task with additional complexity. So commonly edge detection techniques based on spatial domain working. On the spatial domain, an edge can be classically detected via first or second order derivative. Because of second order derivative has high sensitivity towards noises, first order derivative has been used for edge detection commonly and especially gradient which is based on first order derivative is popular. Before examining the first order derivative, we keep in mind our fuzzy system designed based on the relationship between each pixel and its eight closest neighbor pixels.

And so if we recall the gradient which column vector has x and y direction first order derivative, an edge can be found by the following formula:

$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \tag{1}$$

Solution of the Eq.(1) is

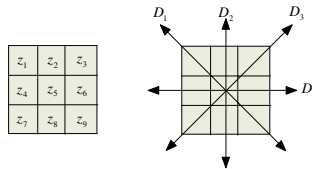
$$\nabla f = mag(\nabla f) = \left[ G_x^2 + G_y^2 \right]^{\frac{1}{2}} = \left[ \left( \frac{\partial f}{\partial x} \right)^2 + \left( \frac{\partial f}{\partial y} \right)^2 \right]^{\frac{1}{2}} \tag{2}$$

$\nabla f$  in Eq.(2) can be an edge, a line or a point at x and/or y direction. The first order numerical derivative calculated by the following formulas:

$$\frac{\partial f}{\partial x} = f(x+1) - f(x) = f(x) - f(x-1), \frac{\partial f}{\partial y} = f(y+1) - f(y) = f(y) - f(y-1) \tag{3}$$

If we assign center and its neighbor pixels in a sample matrix to z coefficients the following equations are possible.

$$\begin{aligned} z_1 = f(x-1, y-1) \quad z_2 = f(x, y-1) \quad z_3 = f(x+1, y-1) \\ z_4 = f(x-1, y) \quad z_5 = f(x, y) \quad z_6 = f(x+1, y) \\ z_7 = f(x-1, y+1) \quad z_8 = f(x, y+1) \quad z_9 = f(x+1, y+1) \end{aligned} \tag{4}$$



**Fig. 1.** Coefficients and possible edge directions for 3x3 sample matrix

We can calculate the first order derivative values by using Eq.(3) for all directions shown in Fig.1.

$$\begin{aligned} \nabla D_1 &= (z_5 - z_1)^2 + (z_9 - z_5)^2, \nabla D_2 = (z_5 - z_2)^2 + (z_8 - z_5)^2 \\ \nabla D_3 &= (z_5 - z_3)^2 + (z_7 - z_5)^2, \nabla D_4 = (z_5 - z_4)^2 + (z_6 - z_5)^2 \end{aligned} \quad (5)$$

and by using Eq.(2) possible edge value are calculated by following formula:

$$\nabla D = [\nabla D_1 + \nabla D_2 + \nabla D_3 + \nabla D_4]^{\frac{1}{2}} \quad (6)$$

The computational burden of implementing Eq.(5) and Eq.(6), it is common practice to approximate the magnitude of the gradient by using absolute values instead of squares and square roots. So the Eq.(5) and Eq.(6) are converted:

$$\begin{aligned} \nabla D_1 &= |z_5 - z_1| + |z_9 - z_5|, \nabla D_2 = |z_5 - z_2| + |z_8 - z_5|, \nabla D_3 = |z_5 - z_3| + |z_7 - z_5|, \\ \nabla D_4 &= |z_5 - z_4| + |z_6 - z_5| \end{aligned} \quad (7)$$

$$\nabla D = \nabla D_1 + \nabla D_2 + \nabla D_3 + \nabla D_4 \quad (8)$$

### 3 Fuzzy Approach

In order to implement a new fuzzy edge detection system  $\nabla D_1, \nabla D_2, \nabla D_3$  and  $\nabla D_4$  are calculated and put into place of the fuzzy system that uses membership functions like Fig.3. The fuzzy system uses singleton fuzzifier and *Zero-order Sugeno Inference System*. Membership functions type is triangle and shown in Fig.2.

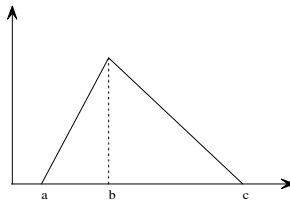


Fig. 2. Triangle membership function

The partial function shown in Eq.(9) for triangle membership function.

$$\mu_{trim}(x) = \begin{cases} \frac{x-a}{b-a}, & a \leq x < b \\ \frac{c-x}{c-b}, & b \leq x \leq c \end{cases} \quad (9)$$



The following rules are applied to the inputs and then a crisp value is found.

**Rule 1:**

- IF ( $\nabla D_1$  is VL) THEN ( $a_1$  is  $k_1$ )
- IF ( $\nabla D_1$  is LO) THEN ( $a_2$  is  $k_2$ )
- IF ( $\nabla D_1$  is MD) THEN ( $a_3$  is  $k_3$ )
- IF ( $\nabla D_1$  is HI) THEN ( $a_4$  is  $k_4$ )
- IF ( $\nabla D_1$  is VH) THEN ( $a_5$  is  $k_5$ )

**Rule 2:**

- IF ( $\nabla D_2$  is VL) THEN ( $b_1$  is  $k_1$ )
- IF ( $\nabla D_2$  is LO) THEN ( $b_2$  is  $k_2$ )
- IF ( $\nabla D_2$  is MD) THEN ( $b_3$  is  $k_3$ )
- IF ( $\nabla D_2$  is HI) THEN ( $b_4$  is  $k_4$ )
- IF ( $\nabla D_2$  is VH) THEN ( $b_5$  is  $k_5$ )

**Rule 3:**

- IF ( $\nabla D_3$  is VL) THEN ( $c_1$  is  $k_1$ )
- IF ( $\nabla D_3$  is LO) THEN ( $c_2$  is  $k_2$ )
- IF ( $\nabla D_3$  is MD) THEN ( $c_3$  is  $k_3$ )
- IF ( $\nabla D_3$  is HI) THEN ( $c_4$  is  $k_4$ )
- IF ( $\nabla D_3$  is VH) THEN ( $c_5$  is  $k_5$ )

**Rule 4:**

- IF ( $\nabla D_4$  is VL) THEN ( $d_1$  is  $k_1$ )
- IF ( $\nabla D_4$  is LO) THEN ( $d_2$  is  $k_2$ )
- IF ( $\nabla D_4$  is MD) THEN ( $d_3$  is  $k_3$ )
- IF ( $\nabla D_4$  is HI) THEN ( $d_4$  is  $k_4$ )
- IF ( $\nabla D_4$  is VH) THEN ( $d_5$  is  $k_5$ )

If  $K$  shows a row vector, the Sugeno constant values are  $K = [0,16,48,128,255]$  and four output values are found by using Eq.(9).

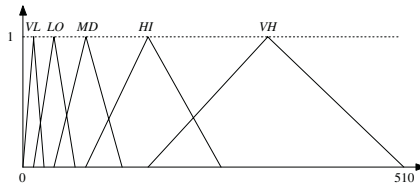


Fig. 3. Membership functions

$$\begin{aligned}
 \bar{y}_1 &= \frac{\mu_{VL}(\nabla D_1) \cdot a_1 + \mu_{LO}(\nabla D_1) \cdot a_2 + \mu_{MD}(\nabla D_1) \cdot a_3 + \mu_{HI}(\nabla D_1) \cdot a_4 + \mu_{VH}(\nabla D_1) \cdot a_5}{\mu_{VL}(\nabla D_1) + \mu_{LO}(\nabla D_1) + \mu_{MD}(\nabla D_1) + \mu_{HI}(\nabla D_1) + \mu_{VH}(\nabla D_1)} \\
 \bar{y}_2 &= \frac{\mu_{VL}(\nabla D_2) \cdot b_1 + \mu_{LO}(\nabla D_2) \cdot b_2 + \mu_{MD}(\nabla D_2) \cdot b_3 + \mu_{HI}(\nabla D_2) \cdot b_4 + \mu_{VH}(\nabla D_2) \cdot b_5}{\mu_{VL}(\nabla D_2) + \mu_{LO}(\nabla D_2) + \mu_{MD}(\nabla D_2) + \mu_{HI}(\nabla D_2) + \mu_{VH}(\nabla D_2)} \\
 \bar{y}_3 &= \frac{\mu_{VL}(\nabla D_3) \cdot c_1 + \mu_{LO}(\nabla D_3) \cdot c_2 + \mu_{MD}(\nabla D_3) \cdot c_3 + \mu_{HI}(\nabla D_3) \cdot c_4 + \mu_{VH}(\nabla D_3) \cdot c_5}{\mu_{VL}(\nabla D_3) + \mu_{LO}(\nabla D_3) + \mu_{MD}(\nabla D_3) + \mu_{HI}(\nabla D_3) + \mu_{VH}(\nabla D_3)} \\
 \bar{y}_4 &= \frac{\mu_{VL}(\nabla D_4) \cdot d_1 + \mu_{LO}(\nabla D_4) \cdot d_2 + \mu_{MD}(\nabla D_4) \cdot d_3 + \mu_{HI}(\nabla D_4) \cdot d_4 + \mu_{VH}(\nabla D_4) \cdot d_5}{\mu_{VL}(\nabla D_4) + \mu_{LO}(\nabla D_4) + \mu_{MD}(\nabla D_4) + \mu_{HI}(\nabla D_4) + \mu_{VH}(\nabla D_4)}
 \end{aligned}
 \tag{10}$$

The last process is addition of all output values calculated by using Eq.(10) to obtain all possible direction components as edge.

$$y = \bar{y}_1 + \bar{y}_2 + \bar{y}_3 + \bar{y}_4 \tag{11}$$

where y is the final result.

### 4 Experimental Results

In this section some experiments are made to verify the performance of the new fuzzy edge detector on different sample images. The image that is not corrupted by any noise are considered here. To perform the experiments some parameters must be determined first. The gray level value of the experimens is ranged [0,255]. Masks used by Sobel, Prewitt and Laplacian edge detector are shown below.



Fig. 4. (a) Sobel x-direction mask, (b) Sobel y-direction mask



Fig. 5. (a) Prewitt x-direction mask, (b) Prewitt y-direction mask

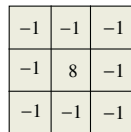


Fig. 6. Laplacian mask

The morphological gradient is calculated by using Eq.(12).

$$g(x, y) = [f(x, y) \oplus B] - [f(x, y) \otimes B] \tag{12}$$

In Eq.(12),  $\oplus$  symbol indicates morphological dilation and  $\otimes$  symbol indicates morphological erosion. The parameters that are used by membership functions are shown here.

$$\begin{aligned}
 I_{trim\_VL} &= [0,8,16] \\
 I_{trim\_LO} &= [8,24,40] \\
 I_{trim\_MD} &= [24,56,88] \\
 I_{trim\_HI} &= [56,120,184] \quad I_{trim\_VH} = [120,315,510]
 \end{aligned}
 \tag{13}$$

Experimental results realized for digitized images with size 256x256. In this experiments, one of the most important points are K values that are defined for output constant values. These parameters play a very important role to determine sharpness of the edges. Thickness of the edges depend on the K values. While  $k_1$  and  $k_2$  are bringing to light background details,  $k_3$ ,  $k_4$  and  $k_5$  bring to light foreground details. Accordingly, values of K selected differently for every inspection and appropriate values can be determined by human experiments that be basic of heuristic approach. Accordingly for the experimental results, these parameters was selected after some attending about this.

For  $K = [0,25,50,128,255]$ , the edges are become evident but are not black-white such as thresholded images. In Fig.12 and Fig.18 these K values are used. In order to has more less edge thickness these parameters defined as  $K = [0,16,32,64,128]$  in Fig.24. Accordingly unwanted edge thickness are prohibited.



Fig. 7. Tomatoes-peppers



Fig. 8. Laplacian applied to the image



Fig. 9. Sobel operator applied to the image



Fig. 10. Prewitt operator applied to the image



**Fig. 11.** Morphological gradient Applied to the image



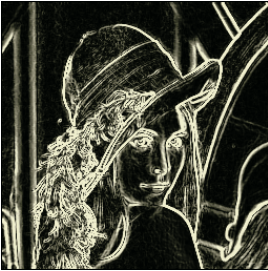
**Fig. 12.** Fuzzy approach applied to the image



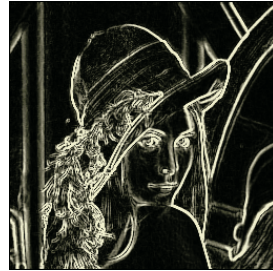
**Fig. 13.** Lena Image



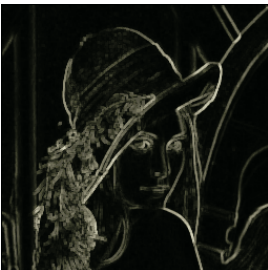
**Fig. 14.** Laplacian applied to the image



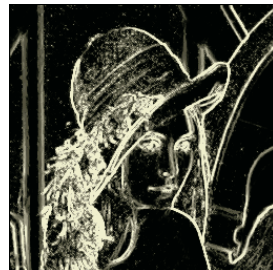
**Fig. 15.** Sobel operator applied to the image



**Fig. 16.** Prewitt operator applied to the image



**Fig. 17.** Morphological gradient applied to the image



**Fig. 18.** New Fuzzy approach applied to image



**Fig. 19.** *Cameraman Image*



**Fig. 20.** Laplacian applied to the image



**Fig. 21.** Sobel operator applied to the image



**Fig. 22.** Prewitt operator applied to image



**Fig. 23.** Morphological gradient applied to the image



**Fig. 24.** . New Fuzzy approach applied to the image

## 5 Conclusions and Future Works

In this work, we have used a new approach for edge detection that one of the most difficult tasks in image processing. In this approach, heurisc rules applied the system and results was observed for different images. Moreover we have seen that this system has dynamic structure adopted by changing rules easily. Accordingly the novel technique called fuzzy approach will be used fairly at the future for different tasks.

## References

- [1] Gonzalez,R.C.,"*Digital Image Processing*", Printice Hall, 2002
- [2] Kerre E.E., Nachtegal M.,"*Fuzzy Techniques in Image Processing (Studies in Fuzziness and Soft Computing, 52)*", Physica Verlag, 2000
- [3] Russo,F., "*Edge Detection in Noisy Images Using Fuzzy Reasoning*", IEEE Trans. on Inst. and Meas., vol.47, no.5, October 1998
- [4] Kuo,Y.H., Lee,C.S., Liu,C.C., "*A New Fuzzy Edge Detection Method for Image Enhancement*", IEEE 0-7803-3796-4/97
- [5] Lee,C.S., Kuo,Y.H., "*Adaptive Fuzzy Edge Detection for Image Enhancement*", IEEE 0-7803-4863-X/98
- [6] Tyan,C.Y., Wang,P.P., "*Image Processing – Enhancement, Filtering and Edge Detection Using the Fuzzy Logic Approach*", IEEE 0-7803-0614-7/93
- [7] Tizhoosh,H.R., "*Fast Fuzzy Edge Detection*", IEEE 0-7803-7461-4/02
- [8] El-Khamy,S.E., Ghaleb,I., El-Yamany,N.A., "*Fuzzy Edge Detection with Minimum Fuzzy Entropy Criterion*", IEEE 0-7803-7527-0/02
- [9] El-Khamy,S.E., Lotfy,M., El-Yamany,N.A.,"*A Modified Fuzzy Sobel Edge Detector*", 17. National Radio Science Conference, Feb.22-24 2000, Minufiya University, Egypt
- [10] Cai,J., Yang,J., Ding,R., "*Fuzzy Iteration Edge Detector*", IEEE 0-7803-6255-5/00
- [11] Miosso, C.J., Bauchspiess, A., "*Fuzzy Inference System Applied to Edge Detection in Digital Images*", Proceeding of the Brazilian Conference on Neural Networks, pp. 481-486, 2001
- [12] Liang, L.R., Looney, C.G., "*Competitive fuzzy edge detection*", Applied Soft Computing, vol. 3, pp. 132-137, 2003

# Fault Detection and Prediction of Clocks and Timers Based on Computer Audition and Probabilistic Neural Networks

S.Y. Chen<sup>1,2</sup>, C.Y. Yao<sup>1</sup>, G. Xiao<sup>1</sup>, Y.S. Ying<sup>1</sup>, and W.L. Wang<sup>1</sup>

<sup>1</sup> College of Information Engineering, Zhejiang University of Technology,  
Hangzhou 310014, China

<sup>2</sup> National Laboratory of Pattern Recognition of Automation Institute,  
Chinese Academy of Sciences, China  
sy@ieee.org  
<http://www.sychen.com/>

**Abstract.** This paper investigates the fault detection and prediction of rhythmically soniferous products, such as clocks, watches and timers. Such products with fault cannot work steadily or probably cause malfunction. The authors extend the concept of computer audition and establish an architectural model of product fault prediction system based on probabilistic neural networks. The system listens to the product sound by the multimedia technology and the sound features are extracted to detect and predict faults by the neural network. The paper analyzes the reasons of timer faults and the corresponding sound features. Experiments are made in the laboratory to demonstrate the proposed method. The technology is expected to apply in factories in coming years for realizing automatic product test and improving efficiency of product inspection.

## 1 Introduction

The preparation. When a timer or clock is manufactured, its functionality has to be inspected before it is sold to customers. The major functionality is the accuracy and faultless performance in a specified working period. Traditionally, people have to inspect these products manually. Since a clock or timer has to be inspected with a long period to wait until a fault caused, the process is with very low efficiency. Inspectors need to test the sample with several poses again and again. It is the common desirability to find an efficient and simple method to do so. This paper proposes to apply the computer audition technology to automate such inspection work.

Generally, computer audition means that computer processes and understands human voice. This paper extends this conception for computer to understand a product's sound features. Just like human themselves, people can listen to all kinds of nature sound and factitious sound, such as wind, pitter-patter, thunderclap, hooter, speaker, machining sound etc. This paper attempts to use computer to understand

soniferous product (with a regular rhythm) and inspect if the target is manufactured without faults. This is especially important to improve production efficiency to an enterprise that produces some mechanical clocks or timers.

Artificial neural network (ANN) is an extremely simplified model of the human brain. Thanks to its ability to learn and generalize, ANN is especially useful for classification and prediction. It is a neoteric field international, and it is developed promptly in the last few decades. Neural network surmounts the limit of traditional numerical computer that is on basis of linear processing. And it is a flag of human to begin to consider the nonlinear living world. Among the common sorts of neural networks, the probabilistic neural network (PNN) is known to have good generalization properties. A PNN is an implementation of a statistical algorithm called kernel discriminant analysis in which the operations are organized into a multilayered feedforward network with four layers, i.e. Input layer, Pattern layer, Summation layer, and Output layer. It can be trained much faster than the back propagation ANNs. The faster training is achieved at the cost of an increased complexity and higher computational and memory requirements. The inspection system presented in this paper works in real-time on common personal computers, once the training has been completed. This paper adopts a PNN for fault detection and prediction of mechanical clocks or timers with the sound features extracted by computer audition.

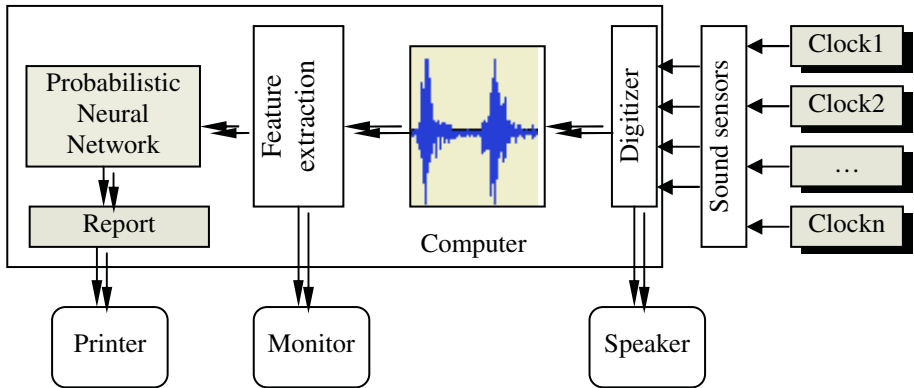


Fig. 1. The Inspection system for detection and prediction of clocks and timers

## 2 The Inspection System

### 2.1 The System Overview

Fig. 1 illustrates a fault detection and prediction system for common mechanical clocks and timers, which takes advantage of artificial neural network and computer audition. The system is mainly composed of some sound sensors (microphones) to



listen to the clocks, a sound digitizer (sound card) to collect the sound, a speaker for selective play, a controller, a feature extraction module to analyze the sound wave from a product, a monitor for selective display, a probabilistic neural network module to predict faults, and some output devices. The samples should be winded with the clockwork spring so that they play sound in regular pattern before inspection. The sound sensors and sound card can sample the sound sequence and carry them to computer memory for processing. Fig. 2 illustrates a segment of sound record of a certain clock. The artificial neural network piles up experience by self-study from known products previously judged by experienced workers, and makes fault prediction on the assembled products. Moreover, there are some auxiliary programs in the system to support database manager, transactions on fault prediction, printing, system on-line control, and other functions.

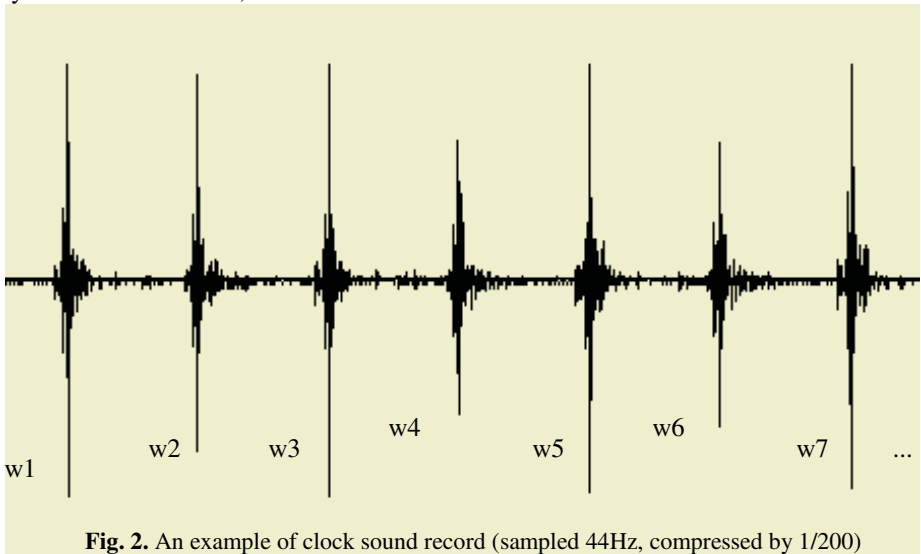


Fig. 2. An example of clock sound record (sampled 44Hz, compressed by 1/200)

## 2.2 The Probabilistic Neural Network for Fault Prediction

Neural networks are computing architectures that consist of massively parallel interconnections of simple neural processors. The study of such architectures has been inspired by the learning abilities and parallelism of biological nervous systems.

Information processing in network is realized by interaction of neurons, the storage of knowledge and information is performed as the fact relationship of neurons, and studying or recognition of network is decided by dynamical process of change of weight linked among neurons. Every neuron has some input values (vector  $X$ ), and every input value is corresponding with a weight or threshold ( $\omega_i$ ). Summing up all input data through weight or threshold, a single output value is  $y$ , where  $y = f(X \times \omega)$ . There are two main stages while the network works. One is learning stage. Given a set of input data and a set of known output data, the network can change the weights or thresholds by self-study. The other is working stage. In this stage, all weights and

thresholds are not changed anymore. The network is only to calculate the output data according the input data.

There are two main reasons to adopt neural network for fault prediction in this paper. (1) For clocks and timers, we are still not clear the precise model of fault expression. A neural network takes advantage not necessary to establish mathematical models. People need only give the present data to network and the network will select a model itself. Furthermore, it will solve the problem all-right. (2) Neural network can contain more noises than other methods, as horary sequence data are usually gone with lots of noises.

Among common neural network models, the probabilistic neural network is most suitable for fault prediction of this application. The architecture of the PNN designed in this paper contains four layers: the input layer, hidden layer, summarization layer, and decision output layer. It will be presented in next section in detail of this network.

### 2.3 Sound Record of a Clock

Taking account the traits of clocks and timers, we can find that its obvious characteristic is working with a regular pattern of sound wave. A mechanical clock sounds "ticktack, ticktack, ..." in certain frequency because it works with gears inside. For a desired product, it will work second by second well and truly.

Under perfect conditions, the sound wave that the clock utters should be quite regular and periodic. But every period may be not similar with each other in practice because many factors and noises exist. Figure 2 illustrates a practical sound wave of a clock, which is sampled in the frequency of 44Hz and is compressed in a ratio of 1/200. However, a clock with fault is obviously not the same as a normal clock somewhere. In this paper, that is studied and judged by the artificial neural network.

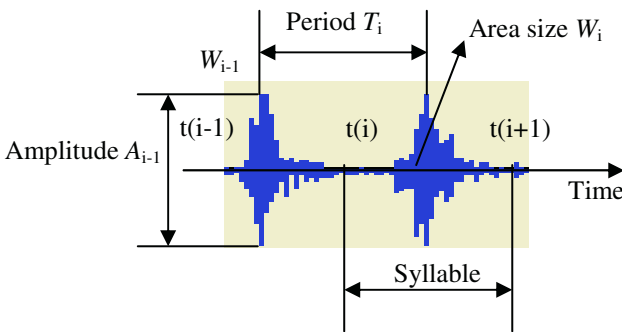


Fig. 3. Feature analysis of clock sound

## 3 Feature Extraction

According to our experiences, the reason why a clock causes malfunction is found is mainly from the part accuracy and assembly. If there is an imprecise part inside or the

parts are unsuitably assembled, the clock will probably stop after working a period of time. An imprecise part may have too rough teeth of gear, unsuitable gap between gears, imprecise dimensions of gear, or deflection assembled.

All abnormal behaviors can be reflected to the set of periods, amplitudes, and syllables of the sound wave although there are many factors that cause malfunction. Every syllable has its shape and size. The shape is not considered in this paper. The area size has the integration characteristic since it is corresponding to the energy of a syllable. That is, the syllable energy  $A=\alpha W$ , where  $W$  is area size. Here the size  $W$  is extracted from sound wave and used as the PNN input.

The inspection time is supposed to start at 0, and the sound wave is

$$z = f(t), \quad 0 \leq t \leq T.$$

After sampled under a certain frequency  $F(\text{Hz})$ , the magnitude is

$$y = g(n) = f(n/F), \quad 0 \leq n \leq FT.$$

If the frequency of object product is  $C$  (Hz), i.e. it strikes  $C$  times every second, the span of a syllable is  $s = F/C$ . Then the syllable No. $i$  is located in  $t(i) < n < t(i+1)$ , or

$$X: n_0 + (i-1)*s < n < n_0 + i*s$$

The starting point of the first syllable is determined as following,

Integral field  $X_0$ :

$$m/F < x < m/F + 1/CL, \quad L = 1/(5 \sim 8)s,$$

Repeat:

$$s_0 = \int |z| dx \quad (x \text{ in } X_0) \\ = \sum |y|, \quad m < n < m + s/L$$

If  $s_0 < s*$  then  $n_0 = m$ , end

End

The aim of this algorithm is to find the interval between two syllables. The letter  $L$  stands for calculated span of a cycle which is usually valued as  $1/5$  to  $1/8$  time of a syllable span  $s$ . If the energy value between the span is no more than a threshold  $S_t$ , it is considered that the starting point of the syllable is located here. And the threshold  $S_t$  is usually valued as 3 to 5 times of the sonority when the testing is silent (but background noise exists), which namely  $S_t = (3 \sim 5) S$ .

Therefore, the syllable No. $i$  has the energy of

$$W_i = \int |z| dx$$

where the integral field  $X$ :

$$t(i) < x < t(i+1), \quad t(i+1) = t(i) + 1/C \\ = \int |f(x)| dx \\ = \sum |g(n)|, \quad \text{where } n_0 + (i-1)*s < n < n_0 + i*s.$$

The sound period  $T_i$  and amplitude  $A_i$  can be easily determined after the syllable is located. Then during the inspection, a certain period (observation window) with a defined number of syllables is chosen to extract sound features as the input of the

PNN. For example, we may always choose the most recent 30 syllables from the wave record and extract the following features:

- $M_W$  - the mean energy of syllables in the defined period (window)
- $\sigma_W$  - the variance of the syllable energy in the defined period
- $M_{DW}$  - the mean differential energy of syllables
- $\sigma_{DW}$  - the variance of the differential syllable energy
- $M_T$  - the mean period length of syllables
- $\sigma_T$  - the variance of the syllable length
- $M_{DT}$  - the mean of differential period length of syllables
- $\sigma_{DT}$  - the variance of the differential syllable length
- $M_A$  - the mean amplitude of syllables
- $\sigma_A$  - the variance of the amplitudes
- $M_{DA}$  - the mean of differential amplitudes of syllables
- $\sigma_{DA}$  - the variance of the differential amplitudes

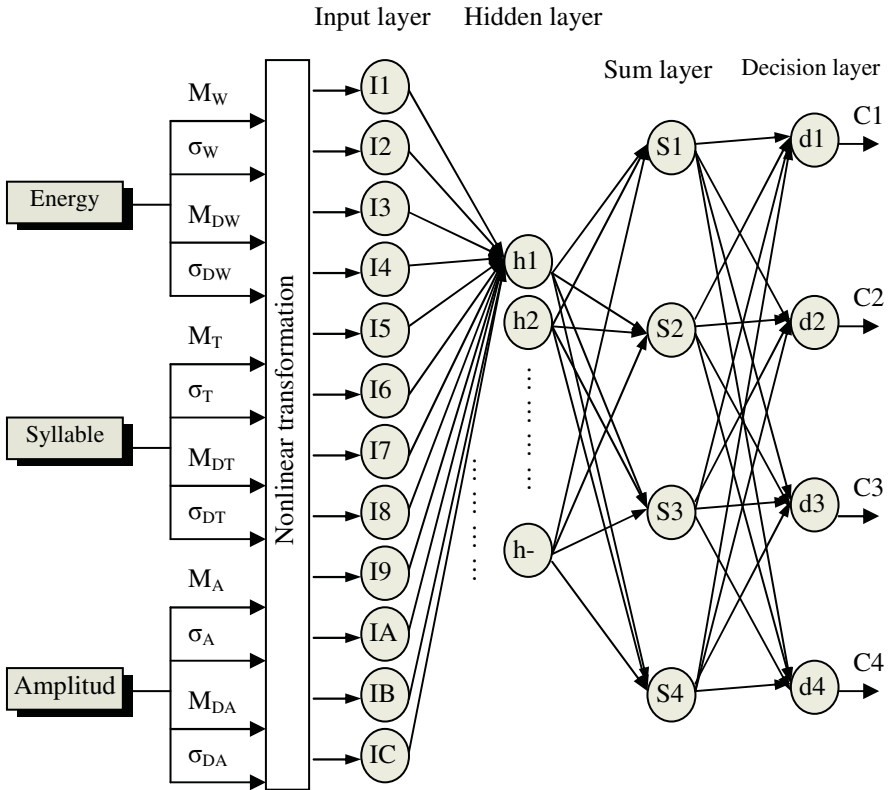


Fig. 4. The PNN for clock fault prediction

## 4 PNN-Based Prediction

A PNN is typically easy to set up and eliminates lengthy and arduous network training iterations but still possesses good generalization characteristics. For example, it takes advantages that fast training process (several orders of magnitude faster than backpropagation), an inherently parallel structure, guaranteed to converge to an optimal classifier as the size of the representative training set increases (no local minima issues), and training samples can be added or removed without extensive retraining.

Fig. 4 illustrates the architecture of the PNN designed in this paper. It contains four layers: the input layer, hidden layer, summarization layer, and decision output layer. In the hidden layer, an activation function is applied to the distance measure between the unknown input and the training example. For example, it can be designed to classify an input vector into several categories with category 1 for “severe fault”, category 2 for “general fault”, category 3 for “general OK”, and category 4 for “perfect”. Input vector is applied to 12 input nodes ( $I_1$  to  $I_C$ ). The hidden layer contains 25 nodes numbered from H1 to H25. Both the summarization layer and decision output layer contain 4 nodes, S1 to S4 and D1 to D4, respectively.

A PNN supplied with Mathworks Matlab and the Neural Network ToolBox is used to conduct the experiments. A sample set with a size of 28 clocks is used in the experiments, in which 20 are used for training and other 8 are used for testing. The result show that 100% of the 8 samples can be predicted correctly in a period of about 15 minute inspection for each sample by the computer audition. However, this good result maybe includes the reason of small set size. If the size increases, the network will probably not able to yield such percentage. Therefore, this result is a bit limited with the current available data in the experiments.

## 5 Conclusion

This paper proposed a computer audition method for fault detection and prediction of mechanical clocks and timers. An audition system is constructed to listen to the product sound by the multimedia technology and the sound features are extracted to detect and predict faults. The probabilistic neural network (PNN) is used for inspection of clocks and timers which will be one of the successful application in this industry.

The limitation of this work is that no enough training and testing samples are available during the experiments of this research at the current stage. This obviously affects the value of established network model for practical application since a PNN usually requires a representative training set (even more so than other types of neural network's).

## Acknowledgments

This work was supported by the National Natural Science Foundation of China (NSFC No. 60405009 and 60374056) and Zhejiang Natural Science Foundation of China (ZJNSF No.Y104185).

## References

1. Rutkowski, L.: Adaptive probabilistic neural networks for pattern classification in time-varying environment. *IEEE Trans. on Neural Networks*. Vol. 15, No. 4, (2004) 811 - 827
2. Lin W.-M., Lin C.-H., Sun Z.-C.: Adaptive multiple fault detection and alarm processing for loop system with probabilistic network. *IEEE Trans. on Power Delivery*. Vol. 19, No. 1, Jan. (2004) 64 – 69
3. Lennox B., Rutherford P.: Novel fault prediction technique using model degradation analysis. *Proc. of the American Control Conference*. Vol.5 (1995) 3274-3278
4. Ruan Q., Yuan B.: Status and trends of computer audio-visual information processing. *Telecommunications Science*, Vol.9, No.4 (1993) 23-30
5. Lau C., Widrow B.: Special Issue on Neural Networks II: Mathematical Analysis: Implementations and Applications. *Proc. IEEE*, Vol.78, No.10, 1990
6. Zhao J.S., Chen B.Z. and Shen J.Z.: A Neural Network Approach to the Dynamic Fault Diagnosis of Hydrocracking Process. *Proc. of the Second Chinese World Congress on Intelligent Control and Intelligent Automation*. Xian, China, June, (1997) 1892-1897
7. Ganchev, T. Fakotakis, N. Tasoulis, D.K. Vrahatis, M.N.: Generalized locally recurrent probabilistic neural networks for text-independent speaker verification. *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*. Vol. 1, May (2004) I-41-4
8. Bolat, B. Kucuk, O.: Speeh/music classification by using statistical neural networks. *IEEE 12th Signal Processing and Communications Applications Conference*. April (2004) 227 - 229
9. Comes, B.; Kelemen, A.: Probabilistic neural network classification for microarraydata. *Int. Joint Conf. on Neural Networks*. Vol. 3, July (2003) 1714 - 1717

# Long Term Prediction of Product Quality in a Glass Manufacturing Process Using a Kernel Based Approach

Tobias Jung<sup>1</sup>, Luis Herrera<sup>2</sup>, and Bernhard Schoelkopf<sup>3</sup>

<sup>1</sup> Johannes Gutenberg-Universitaet,

Fachbereich Mathematik & Informatik, 55099 Mainz, Germany

<sup>2</sup> University of Granada, Dpt. of Computer Architecture and Technology,  
18071 Granada, Spain

<sup>3</sup> M.P.I. for Biological Cybernetics, 72076 Tuebingen, Germany

**Abstract.** In this paper we report the results obtained using a kernel-based approach to predict the temporal development of four response signals in the process control of a glass melting tank with 16 input parameters. The data set is a revised version<sup>1</sup> from the modelling challenge in EUNITE-2003. The central difficulties are: large time-delays between changes in the inputs and the outputs, large number of data, and a general lack of knowledge about the relevant variables that intervene in the process. The methodology proposed here comprises Support Vector Machines (SVM) and Regularization Networks (RN). We use the idea of sparse approximation both as a means of regularization and as a means of reducing the computational complexity. Furthermore, we will use an incremental approach to add new training examples to the kernel-based method and efficiently update the current solution. This allows us to use a sophisticated learning scheme, where we iterate between prediction and training, with good computational efficiency and satisfactory results.

## 1 Introduction

In this paper we report on using kernel based methods to predict the temporal development of four response signals that are quality-related process variables occurring in the manufacturing of glass. The data, provided by Schott Glass (Mainz), consists of 16 input values and four output values (each recorded at 15-minute intervals) and is the rescaled operational data of a glass melt obtained over a period of forty weeks. For the last two weeks, only the 16 input values are known, i.e. the control targets of the process engineers and the sometimes unexpected, but measurable external influences (like e.g. outside temperature), and our goal was to predict the four output values. The real physical meaning of the data and the manner in which the melting reacts to these inputs was not given. The central difficulty in forecasting the outputs is the seemingly irregular

---

<sup>1</sup> To obtain the data set `cwt_2004`, please contact [katharina.lankers@schott.com](mailto:katharina.lankers@schott.com).

behavior with random spikes and the unknown delay time between variation of an input signal and the “response” from the glass melting tank.

Several paradigms and methodologies have been applied for time series prediction problems. Specifically, support vector machines (SVM) and kernel-based methods (KM) are receiving increasing attention [6, 8], due to its remarkable characteristics such as good generalization performance, the absence of spurious local minima, the possibility of sparse representation of the solution and the relative independence of the computational complexity on the number of input dimensions of the problem [4, 7]. Nevertheless, one disadvantage of KMs is that the computational complexity scales at least quadratically in the number of training samples. Thus a large amount of computation time will be involved when KMs are applied for solving large-size problems.

In this paper we present a kernel based approach that makes use of the idea of the sparse matrix approximation [3] in order to reduce the computational complexity of the problem and also as a means of regularization. Additionally, the performance of the model is improved using an iterative process that augments the original training sequence, by increasingly adding as new input variables noisy copies of the original data (obtained by learned 1-step predictions). The results obtained for the long term prediction problem showed that our approach obtained satisfactory results in comparison to those presented for the EUNITE 2003 competition [1].

The rest of the paper is organized as follows. Section 2 presents the kernel based learning methodology used in this paper. Section 3 presents the application of the methodology to the `cwt_2004` data set and the predictions obtained. Section 4 concludes the work.

## 2 Methodology, Kernel Based Learning

### 2.1 SVM and RN

Given a data set of  $\ell$  examples  $\{(\mathbf{x}_i, y_i)\}_{i=1}^{\ell}$  with  $\mathbf{x}_i \in \mathcal{X} \subset \mathbb{R}^d$  being the inputs and  $y_i \in \mathcal{Y} \subset \mathbb{R}$  being the outputs, the goal is to learn the underlying model. In this paper, we will consider as the space of candidate functions, the Reproducing Kernel Hilbert Space (RKHS)  $\mathcal{H}$  of functions  $f : \mathcal{X} \rightarrow \mathcal{Y}$  endowed with reproducing kernel  $k$ , where  $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathcal{Y}$  is a symmetric, positive definite function (e.g. think of Gaussian RBF). The underlying function can be thus reconstructed solving a Tikhonov functional of the following general form

$$\min_{f \in \mathcal{H}} H[f] = \frac{1}{\ell} \sum_{i=1}^{\ell} c(\mathbf{x}_i, y_i, f(\mathbf{x}_i)) + \lambda \|f\|_{\mathcal{H}}^2 \tag{1}$$

where the first term measures the error in the approximation and the second term measures the complexity (i.e. the smoothness) of the current candidate. The Representer Theorem tells us that any solution to (1) has a representation in the form:  $f(\cdot) = \sum_{i=1}^{\ell} \beta_i k(\mathbf{x}_i, \cdot)$  (i.e. as a sum of kernels centred on the data)



where  $\beta_i$  are the coefficients that need to be determined [4]. The choice of the cost function  $c$  in (1) leads to two different methodologies:

- Quadratic costs  $c(\mathbf{x}_i, y_i, f(\mathbf{x}_i)) = \frac{1}{2}(y_i - f(\mathbf{x}_i))^2$  lead to RN. The coefficients  $\beta_i$  are obtained by solving the linear system

$$\beta = (K^T K + \ell \Lambda K)^{-1} K^T \mathbf{y} = (K + \ell \Lambda I)^{-1} \mathbf{y}$$

with symmetric kernel matrix  $K$ . Note that we have to estimate one coefficient for each data point.

- $\varepsilon$ -insensitive costs  $c(\mathbf{x}_i, y_i, f(\mathbf{x}_i)) = \max(0, |y_i - f(\mathbf{x}_i)| - \varepsilon)$  lead to Support Vector Regression. The coefficients  $\beta_i$  are obtained using  $\beta_i = (\alpha_i^* - \alpha_i)$  and solving the constrained quadratic programming (QP)

$$\min_{\alpha, \alpha^* \in \mathbb{R}^\ell} -\frac{1}{2}(\alpha^* - \alpha)^T K(\alpha^* - \alpha) - \varepsilon(\alpha^* + \alpha)^T \mathbf{e} + (\alpha^* - \alpha)^T \mathbf{y} \quad (2)$$

subject to  $\mathbf{0} \leq \alpha^*, \alpha \leq C \mathbf{e}$ , where  $\alpha, \alpha^* \in \mathbb{R}^\ell$  denotes the unknowns,  $K \in \mathbb{R}^{\ell \times \ell}$  the kernel matrix,  $\mathbf{e} = (1, \dots, 1)^T$  and  $C \in \mathbb{R}_{\geq 0}$  corresponds to the regularization parameter. Note that we have to calculate as many coefficients as we have data.

From a practical point of view, in order to apply these methods we need to choose (e.g. via cross validation) the kernel  $k$ , the tolerance  $\varepsilon$  and the right amount of regularization  $C$  or  $\Lambda$ .

In both cases the overall computational demand increases (at least) quadratically in the number of data. Moreover, if we were to incrementally add new samples and simultaneously needed to predict using the currently available model, we would constantly have to retrain our solution from scratch. Clearly this is completely infeasible. The next subsection describes a possible solution to this problem.

## 2.2 Sparse Approximation and Incremental Learning

The technique of *sparse approximation* allows us to dramatically reduce the number of variables considered in the optimization problem. It is based on the observation that the kernel matrix  $K$  often has rapidly decaying eigenvalues, and thus that the data in the feature space spans a rather low-dimensional manifold. Instead of using all the data, we can restrict us to use only the data points that compose the basis of this manifold. Therefore, we can approximate the data set by choosing just a few samples (a suitable basis) and projecting the remaining ones onto their span [2, 3, 4, 5] (this technique is very common in the context of Gaussian process regression).

For example let the first  $m$  samples be our basis  $\{k(\mathbf{x}_i, \cdot)\}_{i=1}^m$  ( $m \ll \ell$ ). Then the remaining  $\ell - m$  ones can be approximated via linear combination

$$k(\mathbf{x}_i, \cdot) \approx \sum_{j=1}^m a_{ij} k(\mathbf{x}_j, \cdot) \quad , \quad i = m + 1 \dots \ell. \quad (3)$$

Since we are dealing with elements in RKHS  $\mathcal{H}$ , we have to determine coefficients  $a_{ij}$  such that the distance

$$d_i := \left\| k(\mathbf{x}_i, \cdot) - \sum_{j=1}^m a_{ij} k(\mathbf{x}_j, \cdot) \right\|_{\mathcal{H}}^2, \quad i = m + 1 \dots \ell \tag{4}$$

is minimized. Writing (4) in terms of the inner product (and recalling that in  $\mathcal{H}$  we have  $\langle k(\mathbf{x}_i, \cdot), k(\mathbf{x}_j, \cdot) \rangle_{\mathcal{H}} = k(\mathbf{x}_i, \mathbf{x}_j)$ ) and setting its derivate to zero leads to

$$\mathbf{a}_i = \tilde{K}^{-1} \tilde{\mathbf{k}}_i \tag{5}$$

where  $\tilde{K} \in \mathbb{R}^{m \times m}$  with  $[\tilde{K}]_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$  and  $\tilde{\mathbf{k}}_i \in \mathbb{R}^m$  with  $\tilde{\mathbf{k}}_i = (k(\mathbf{x}_1, \mathbf{x}_i), \dots, k(\mathbf{x}_m, \mathbf{x}_i))^T$ . Once we have determined  $\mathbf{a}_i$  corresponding to input  $\mathbf{x}_i$ , we obtain the corresponding approximation error as

$$d_i = k_{ii} - \tilde{\mathbf{k}}_i^T \mathbf{a}_i \tag{6}$$

This quantity tells us how well the basis  $\{k(\mathbf{x}_i, \cdot)\}_{i=1}^m$  is able to approximate a given sample. Now it is very straightforward to use this error as a guide to (greedily) build up the basis online [9], thus allowing incremental addition of data: every time a new sample arrives, we check whether it can be approximated by the current basis well enough. If the error is below some chosen threshold, we do not need to add the current sample to the basis and can hence discard it. Only samples that cannot be approximated well are added to the basis. This incremental approach for building the manifold basis usually reduces the total number of samples to a small fraction of it.

### 2.3 Solving a Reduced Problem

Thus, we can use the sparse greedy approximation to reduce the computational workload [3]: instead of solving a QP in  $\ell$  variables, we only need to consider the  $m$  variables corresponding to the selected basis (usually  $m \ll \ell$ ). Let  $A \in \mathbb{R}^{\ell \times m}$  be the matrix consisting of rows  $\mathbf{a}_i$  from (5). Then we approximate the full kernel matrix  $K \in \mathbb{R}^{\ell \times \ell}$  via  $K \approx A \tilde{K} A^T$ . We define *reduced* variables  $\tilde{\boldsymbol{\alpha}}, \tilde{\boldsymbol{\alpha}}^* \in \mathbb{R}^m$  by setting  $\tilde{\boldsymbol{\alpha}} = A^T \boldsymbol{\alpha}$  and  $\tilde{\boldsymbol{\alpha}}^* = A^T \boldsymbol{\alpha}^*$ . Now instead of solving the full QP (2) we can formulate a *reduced* QP: we replace  $K$  by  $A \tilde{K} A^T$  and  $\mathbf{y}$  by  $\tilde{\mathbf{y}} = A^\dagger \mathbf{y}$  and obtain

$$\min_{\tilde{\boldsymbol{\alpha}}, \tilde{\boldsymbol{\alpha}}^* \in \mathbb{R}^m} -\frac{1}{2} (\tilde{\boldsymbol{\alpha}}^* - \tilde{\boldsymbol{\alpha}})^T \tilde{K} (\tilde{\boldsymbol{\alpha}}^* - \tilde{\boldsymbol{\alpha}}) - \varepsilon (\tilde{\boldsymbol{\alpha}}^* + \tilde{\boldsymbol{\alpha}})^T A^\dagger \mathbf{e} + (\tilde{\boldsymbol{\alpha}}^* - \tilde{\boldsymbol{\alpha}})^T \tilde{\mathbf{y}} \tag{7}$$

Note that if we solve the *reduced* problem (7), we obtain the same result as if we had solved the full problem (2) using only the training samples  $\{\mathbf{x}_i, \tilde{\mathbf{y}}_i\}_{i=1}^m$  in the basis. However, solving the reduced problem only depends on  $m$  and is asymptotically independent of  $\ell$ . A similar reduction to an m-by-m problem is obtained in the case of regularization networks [5].

### 3 Predicting the Schott Data `cwt_2004`

The data set `cwt_2004` consists of 20 measurements obtained in regular intervals (once every 15 minutes) from a glass melting process. Each row contains 20 observations from one time step. The first 16 observations (denoted by `in1–in16`) are measurements that *could* have an influence on the targets (like e.g. room temperature). The remaining 4 observations `out1–out4` are the targets that we want to predict. The task in the EUNITE-2003 challenge was the following: use the given values for time steps 1-16 000 to learn a model that predicts `out1–out4` for the next 1568 steps.

#### 3.1 Preprocessing

The given data set `cwt_2004` consisted of raw measurements. We used the following steps during preprocessing of the data: replace missing values by an estimate (linear interpolation), remove outliers, apply data smoothing (rolling mean), normalize inputs and de-trend (remove increasing trend in first output, that often can not be implicitly considered by traditional learning methodologies).

#### 3.2 Timeseries Prediction as Supervised Learning

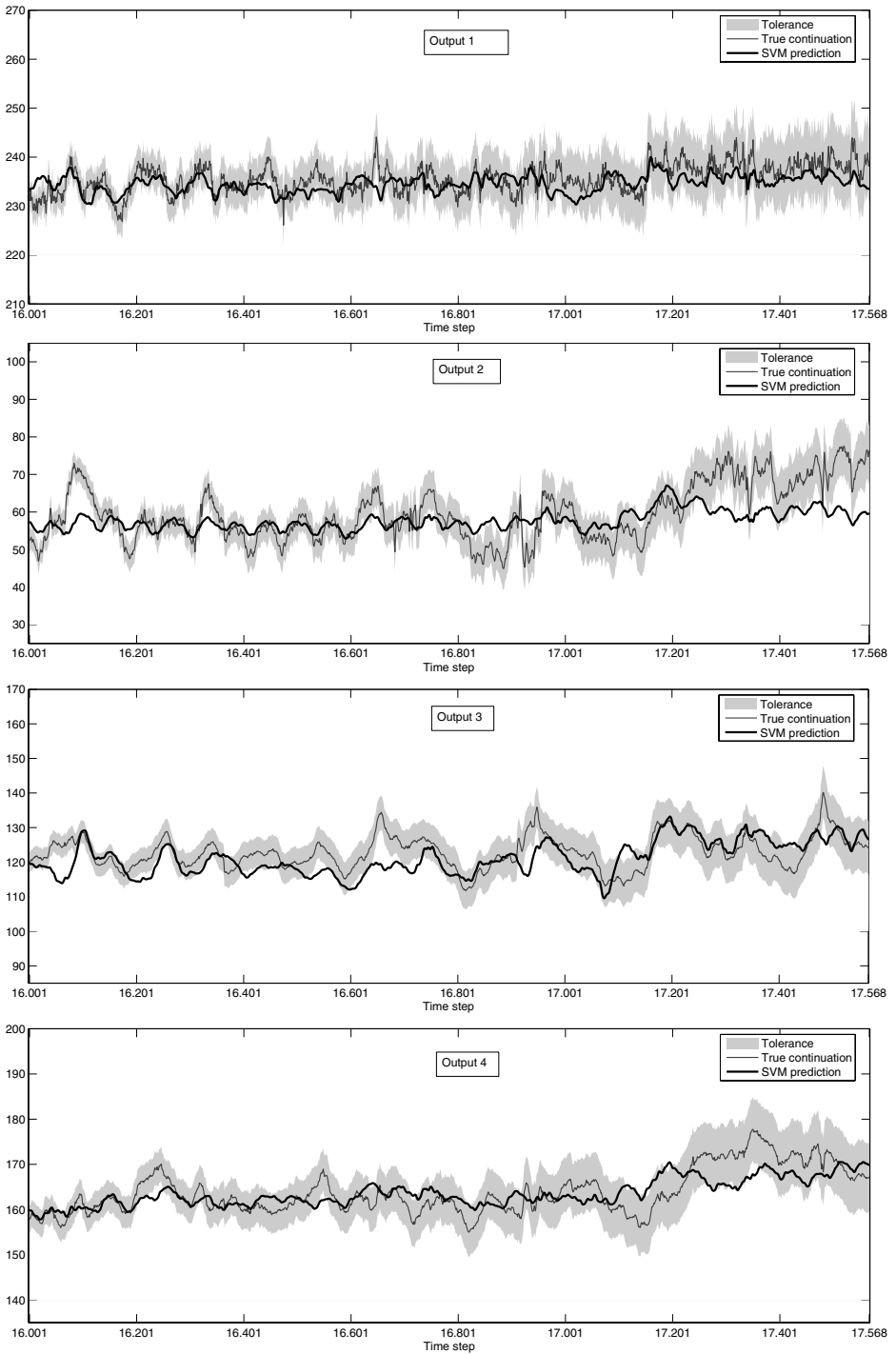
To obtain a model suitable for future predictions we use the supervised learning framework with 1-step predictions as targets and  $\lambda$  past observations as inputs. Thus our training samples  $\{(\mathbf{x}_t, y_t)\}_t$  were given by

$$\begin{aligned} \mathbf{x}_t &= (\text{out}_{t-1}, \dots, \text{out}_{t-\lambda}, \text{inp1}_t, \dots, \text{inp1}_{t-\lambda}, \dots, \text{inp16}_t, \dots, \text{inp16}_{t-\lambda})^T \\ y_t &= \text{out}_t \end{aligned}$$

We used lag  $\lambda = 40$ . Each of the 4 outputs is treated independently, that is we train 4 different models. Instead of using the usual approach of feeding the

**Table 1.** Comparison of the prediction errors for time step 16 001-17 568

|                 |                  | SVM     | RN              |
|-----------------|------------------|---------|-----------------|
| <b>output 1</b> | MSE              | 7.44    | 9.44            |
|                 | TOL=0.01         | TubeERR | 115.54 149.87   |
|                 | $\sigma = 0.015$ | #Basis  | 591             |
| <b>output 2</b> | MSE              | 41.83   | 47.41           |
|                 | TOL=0.011        | TubeERR | 2082.12 2382.44 |
|                 | $\sigma = 0.005$ | #Basis  | 195             |
| <b>output 3</b> | MSE              | 24.07   | 22.28           |
|                 | TOL=0.01         | TubeERR | 1214.25 1069.10 |
|                 | $\sigma = 0.013$ | #Basis  | 476             |
| <b>output 4</b> | MSE              | 20.84   | 23.29           |
|                 | TOL=0.009        | TubeERR | 781.02 1115.24  |
|                 | $\sigma = 0.013$ | #Basis  | 486             |



**Fig. 1.** Results using SVMs. We show the original outputs and the prediction obtained using SVMs. The gray curves mark the boundaries of the tolerated error TubeERR

training examples *once* to the function approximator, we used a more complicated, iterated training procedure adapted from [2] (dropping the `inp` to avoid cluttered notation):

1. Iteration: using  $\{(\mathbf{x}_i, y_i)\}_{i=\lambda}^\ell$  we obtain the model  $f^1$ . With  $f^1$  we can compute the 1-step predictions for the *known* time steps  $i = \lambda + 1 \dots \ell$  to obtain  $\hat{y}_i^1 = f^1(\mathbf{x}_i)$ . Now we assemble the modified training samples  $\{(\mathbf{x}_i^1, y_i)\}_{i=\lambda}^\ell$  with

$$\mathbf{x}_i^1 = (\hat{y}_{i-1}^1, y_{i-2}, y_{i-3}, \dots, y_{i-\lambda}, \dots)^T$$

and add them to the current training set.

2. Iteration: using  $\{(\mathbf{x}_i, y_i)\}_{i=\lambda}^\ell \cup \{(\mathbf{x}_i^1, y_i)\}_{i=\lambda}^\ell$  we obtain model  $f^2$ . With  $f^2$  we can compute the 1-step predictions for the *known* time steps  $i = \lambda + 1 \dots \ell$  to obtain  $\hat{y}_i^2 = f^2(\mathbf{x}_i)$ . Now we assemble the modified training samples  $\{(\mathbf{x}_i^2, y_i)\}_{i=\lambda}^\ell$  with

$$\mathbf{x}_i^2 = (\hat{y}_{i-1}^2, \hat{y}_{i-2}^2, y_{i-3}, \dots, y_{i-\lambda}, \dots)^T$$

and add them to the current training set.

3. Iteration etc.

Thus, in the  $k$ th iteration we build our model using the training sequence

$$\{(\mathbf{x}_i, y_i)\}_{i=\lambda}^\ell \cup \{(\mathbf{x}_i^1, y_i)\}_{i=\lambda}^\ell \cup \dots \cup \{(\mathbf{x}_i^k, y_i)\}_{i=\lambda}^\ell .$$

Note that this iterated training procedure greatly benefits from *sparse approximation* which drastically reduces the computational complexity. In the simulations performed, using repeated training resulted in much better predictions when compared with the results obtained using just one iteration. Thus we could observe that augmenting the original training sequence by increasingly "noisy" copies of the original data (where the "noise" is obtained by the learned 1-step predictions) is very helpful in getting good forecasts.

### 3.3 Results

In this section we compare the resulting predictions obtained from RN and SVM using *sparse approximation* in both cases and Gaussian RBF kernels. To determine the parameters, we used the following four validation sets: (1) train 1-13 500 predict 13 501-14 500 (2) train 1-14 000 predict 14 001-15 000 (3) train 1-14 500 predict 14 501-15 500 (4) train 1-15 000 predict 15 001-16 000. The optimal parameters that led to the best (averaged) prediction error during validation were then used to obtain the final model. These parameters are: (output1) TOL=0.01,  $\sigma = 0.015$  (output2) TOL=0.011,  $\sigma = 0.005$  (output3) TOL=0.01,  $\sigma = 0.013$  (output3) TOL=0.009,  $\sigma = 0.013$ . The parameters governing regularization ( $C = 500, \Lambda = 0.1$ ) and tolerance ( $\varepsilon = 0.1$ ) were determined outside the validation loop.

Forecasting the global trend of the outputs is more important than the modelling of high-frequency variations. To this end a custom error measure TubeERR

was devised, that does not penalize small deviations. Also an emphasize is done in that forecasts into the far future are less reliable. Thus the TubeErr measure is defined as

$$\text{TubeERR}(i) = \max(0, |y_i - f(\mathbf{x}_i)| - (3 + 0.003 * i)) \quad , i = 0, 1, \dots, 1568 - 1$$

Thus, to evaluate the quality of the predictions, we use MSE and the TubeERR error measure provided by Schott. Using these two measures, the optimal number of iterations in the iterated learning scheme was calculated using pre-tests on the validation sets. The achieved error is summarized in Table 1. Figure 1 shows the resulting predictions using SVM; see how for the four outputs, the prediction obtained follows the main variations and is inside the TubeERR region most of the 1568 time steps.

## 4 Conclusions

In this paper we have reported the use of a kernel-based method to predict the long term time series used in the data modelling competition EUNITE-2003. We have used the idea of sparse approximation as a means of reducing the computational complexity and an incremental learning approach to add new training samples to the kernel-based method and efficiently update the current solution. The results obtained support the suitability of the proposed methodology for this long term time series prediction problem with high training complexity.

## References

1. EUNITE Competition 2003: Prediction of product quality in glass manufacturing, [www.eunite.org](http://www.eunite.org) (2003)
2. Engel, Y., Mannor, S., Meir, R.: The Kernel Recursive Least Square Algorithm. *IEEE Transactions on Signal Processing*, **52(8)** (2004) 2275–2285
3. Engel, Y., Mannor, S., Meir, R.: Sparse online greedy support vector regression. *Proc. of 13th European Conference on Machine Learning*. Springer (2002)
4. Schoelkopf, B., Smola, A.: *Learning with Kernels*. Cambridge, MA: MIT Press (2002)
5. Smola, A., Schoelkopf, B.: Sparse greedy matrix approximation for machine learning. *Proc. of 17th International Conference on Machine Learning*. Morgan Kaufmann (2000)
6. Cao, L.J., Tay, F.E.H.: Support Vector Machine With Adaptive Parameters in Financial Time Series Forecasting. *IEEE Transactions on Neural Networks*. **14(6)** (2003) 1506–1518
7. Cristianini, N, Shawe-Taylor, J.: *An Introduction to Support Vector Machines*. Cambridge University Press. Cambridge, England (2000)
8. Chang, M.-W., Chen, B.-J., Lin, C.-J.: EUNITE Network Competition: Electricity Load Forecasting , November 2001. Winner of EUNITE world wide competition on electricity load prediction.
9. Csato, L., Opper, M.: Sparse on-line Gaussian processes. *Neural Computation* **14(3)** (2002) 641–669

# Time Series Forecast with Anticipation Using Genetic Programming

Daniel Rivero, Juan R. Rabuñal, Julián Dorado, and Alejandro Pazos

Univ. A Coruña, Fac. Informática, Campus Elviña, 15071, A Coruña, Spain  
danielrc@mail2.udc.es, {juanra, julian, apazos}@udc.es

**Abstract.** This paper presents and application of Genetic Programming (GP) for time series forecast. Although this kind of application has been carried out with a wide range of techniques and with very good results, this paper presents a different approach. In most of the experiments done in time series forecasting the objective is, from a consecutive set of samples or time interval, to obtain the value of the sample in the next time step. The aim of this paper is to study the forecasting not only on the next sample, but in general several samples forward. This will allow the building of more complete prediction systems. With this objective, one of the most widely used series for this kind of application has been used, the Mackey-Glass series.

## 1 Introduction

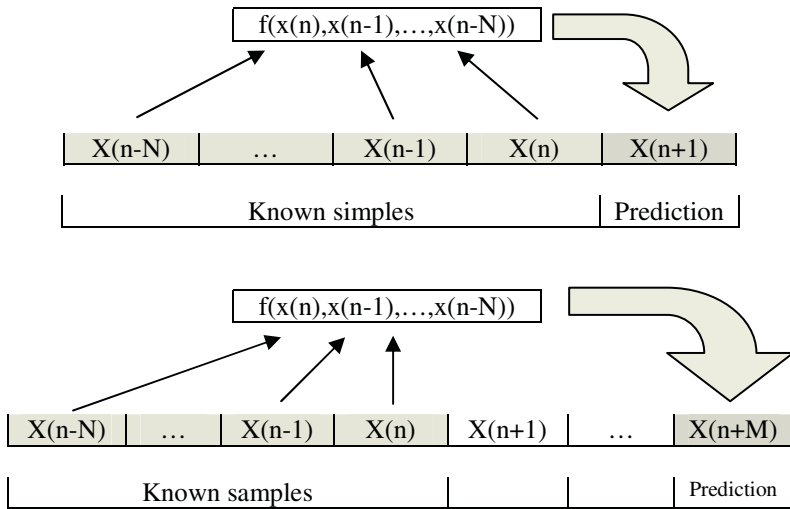
Time series forecasting is a classical application in the world of Artificial Intelligence, and in general in the world of computation [1][2][3], and it has many practical applications. It has been successfully applied to a wide range of different environments and problems, from financial series forecasting [4] to the modelling of complex hydrological processes [5][6]. Time series forecasting has been done using very different techniques, such as Artificial Neural Networks (ANNs) [7][8][9], Genetic Algorithms (GAs) [10] or Genetic Programming (GP) [11][12].

GP [11][13] is the technique used in this paper. This technique has already been extensively used in the field of signal processing, with practical applications such as signal modelling [6] or filter design [12][14]. The ability of GP to develop mathematical equations, and in general any kind of expressions, has made possible that it may be successfully applied on signal processing, and its application in time series forecast is obvious [14].

## 2 Description of the Problem

The classical approach for forecasting time series involves taking a time interval with the objective of predicting the value of the series in the following moment of time. This consists on building a system that takes a consecutive set of samples from the signal and returns a prediction of the next sample.

There are several ways to build this system with GP. The most usual one is to use GP to generate a mathematical expression that, taking as variables the consecutive group of samples, returns a prediction of the value of the following sample. However, in this paper the prediction will not be made only on the next value of the signal, but in general on the M next values of the signal. To achieve this, different systems will be created, one for each value of the anticipation with which the prediction is desired. This may be seen on Fig. 1.



**Fig. 1.** Prediction systems with one forward sample and with M forward samples

This paper will focus on the study of 2 important parameters. These parameters are the number of consecutive samples taken for prediction (N), and the number of forward samples with which the prediction is desired (M).

To do this, a well-known time series has been taken. This series is the Mackey-Glass time series [15], which has been extensively used for time series forecast in the bibliography. For this study, 200 samples of this series were taken. The Mackey-Glass equation is an ordinary differential delay equation:

$$\frac{dx}{dt} = \frac{\alpha x(t-\tau)}{1+x^c(t-\tau)} - bx(t) \tag{1}$$

Choosing  $\tau = 30$ , the equation becomes chaotic, and only short-term predictions are feasible. Integrating the equation (1) in the rank  $[t, t + \delta t]$ , the following expression is obtained:

$$x(t + \Delta t) = \frac{2-b\Delta t}{2+b\Delta t} x(t) + \frac{\alpha\Delta t}{2+b\Delta t} \left[ \frac{x(t+\Delta t-\tau)}{1+x^c(t+\Delta t-\tau)} + \frac{x(t-\tau)}{1+x^c(t-\tau)} \right] \tag{2}$$



### 3 GP Configuration

As mentioned beforehand, the way in which GP will be used to make predictions is through the creation of mathematical expressions. These equations will return the predictions of the desired values from a consecutive set of samples. For this objective, GP has been configured with the most common mathematical expressions, and using as inputs the values of the samples of the interval. These inputs are introduced into GP as variables. Therefore, the configuration of the GP algorithm is as shown on Table 1. The operator “%” stands for the protected division operator, which returns a value of 1 when a number is divided by 0 and the result of the division in other case.

**Table 1.** Terminal and function sets

|                     |                  |                           |
|---------------------|------------------|---------------------------|
| <b>Function set</b> |                  | +, -, *, %                |
| <b>Terminal set</b> | <b>Constants</b> | [-1, 1]                   |
|                     | <b>Variables</b> | X(n-N), ..., X(n-1), X(n) |

Each expression created by GP will be evaluated through the whole signal, and, as result, the fitness function will return the mean difference between the forecasted values and the actual values. This fitness function will be, for a signal with T samples and taking N variables (size of the time interval) and for a prediction of M forward samples, the following expression:

$$\frac{1}{T - (M + N - 1)} \sum_{n=N}^{T-M} |x(n+M) - f(x(n), x(n-1), \dots, x(n-N))| \tag{3}$$

where  $f(x(n), x(n-1), \dots, x(n-N))$  is the expression generated by GP that is being evaluated. This expression cannot be evaluated on the whole signal, as an initial set of samples are required to make the first prediction. This initial set of samples is N. The number of samples in which this expression can be evaluated on the signal is  $T - (N+M-1)$ , taken from N (first prediction) to T-M (last prediction), as may be graphically seen on Fig. 1.

Prediction will be done separately for each of the values on M and N. Therefore, a different system for each anticipation value of the prediction will be generated.

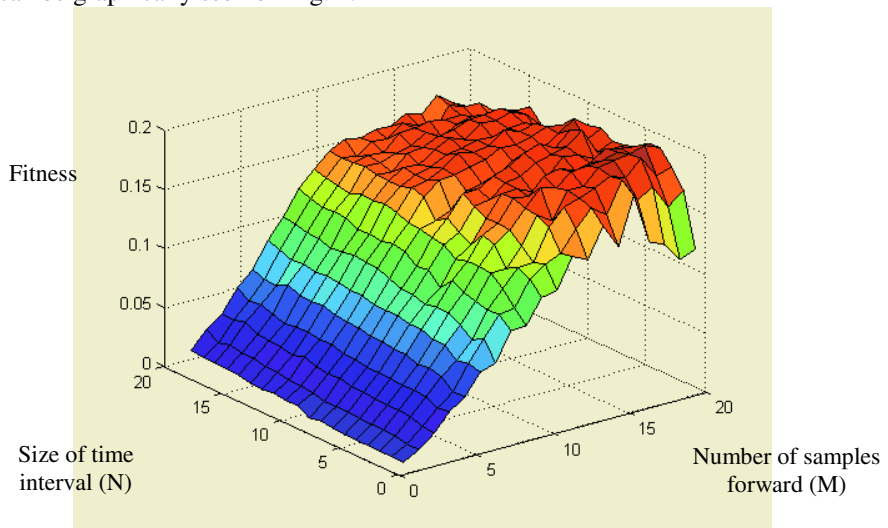
## 4 Results

### 4.1 Training

The main interest of this paper is to study the results with different values of M and N. In this sense, many different trials have been done with different values of M and N, but, in order to obtain coherent results, the same GP configuration had to be kept. With this purpose, several trials were done with different values of M and N in order to obtain a good GP configuration. This GP configuration is the following:

- Selection Algorithm: Roulette.
- Mutation rate: 5 %.
- Crossover rate: 95 %
- Population: 1000 individuals.

This configuration was kept in all of the trials with different values of M and N. These two parameters have taken values until they reached a maximum value of 20. This means that to make a prediction, an interval with a size between 2 and 20 samples from the signal was taken, and this prediction was made with an anticipation from 1 to 20 samples. For each combination of the parameters M and N, 30 independent runs were done, so the total number of runs carried out is  $19 \times 20 \times 30 = 11400$ . After these trials, the mean values of the error for each combination of N and M, can be graphically seen on Fig. 2.



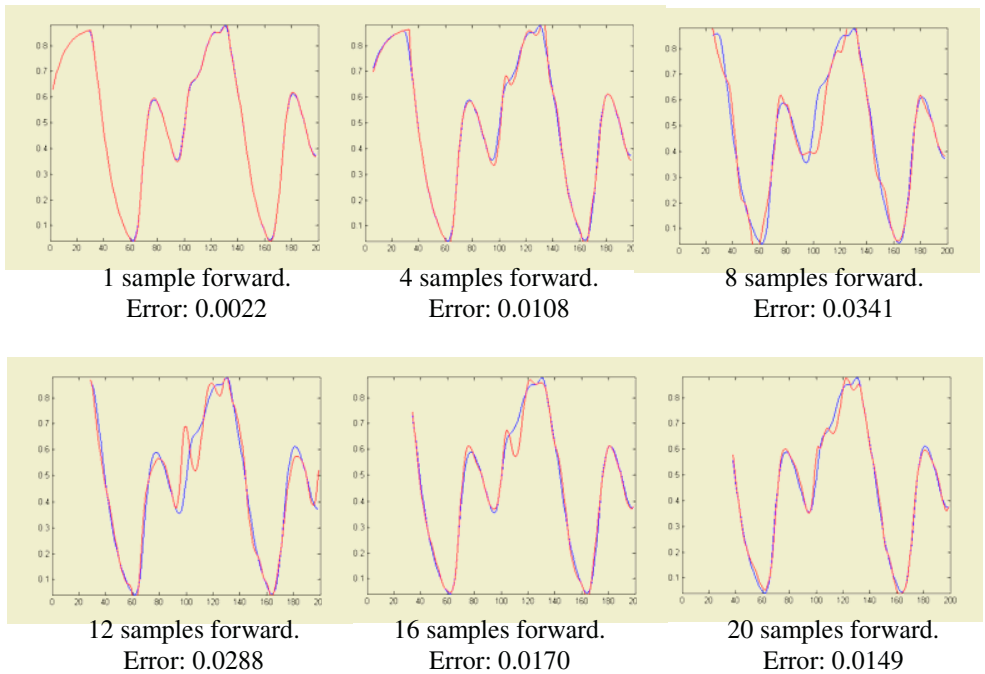
**Fig. 2.** Mean error values obtained for each combination of M and N

Logically, the error increases as the desired anticipation for the prediction increases too (the value of M). But the system reaches a point in which the difference in the value on M is not significant and the values of the error are very similar, except in those cases in which its value is very low.

Before looking at Fig 2, one could think that the system would obtain better results with higher values of N because the GP algorithm would be provided with more information from the signal in order to make better predictions. This is not true. Fig. 2 shows that the value of the error obtained is, in general, lower for lower values of N. This means that a lower error is obtained by taking less information from the signal. This occurs because as the number of samples from the signal taken for making the prediction increases, the search area also increases, because the system will have more variables. Therefore, the system will takes longer to evolve and return a good expression and it will tend more to fall on a local optima.

However, it is important to realize that the values shown on Fig. 2 are just mean values. The best error values were obtained, for low  $M$  values, with few information from the signal ( $N = 2, 3$ ), and for medium and high  $M$  values, with high  $N$  values ( $N = 15, \dots, 20$ ). This means that it is necessary to take a great deal of information from the signal in order to make a prediction significantly in advance. This cannot be seen on Fig. 2, because they show mean error values, and to obtain these values the same GP parameters had to be used. For high values of  $N$ , it would have been suitable to use a bigger population size in order to escape from falling into local optima, but it was necessary to keep the population size constant to obtain coherent results.

Fig 3. shows the signals obtained with the best expressions returned by GP with values of advances of  $M = 1, 4, 8, 12, 16$  and  $20$  samples forward. These expressions were obtained taking  $2, 3, 18, 18, 19$  and  $19$  samples from the signal respectively. So, it can be seen that it is necessary to take more information from the signal in order to make a satisfactory prediction with high values of  $M$ .



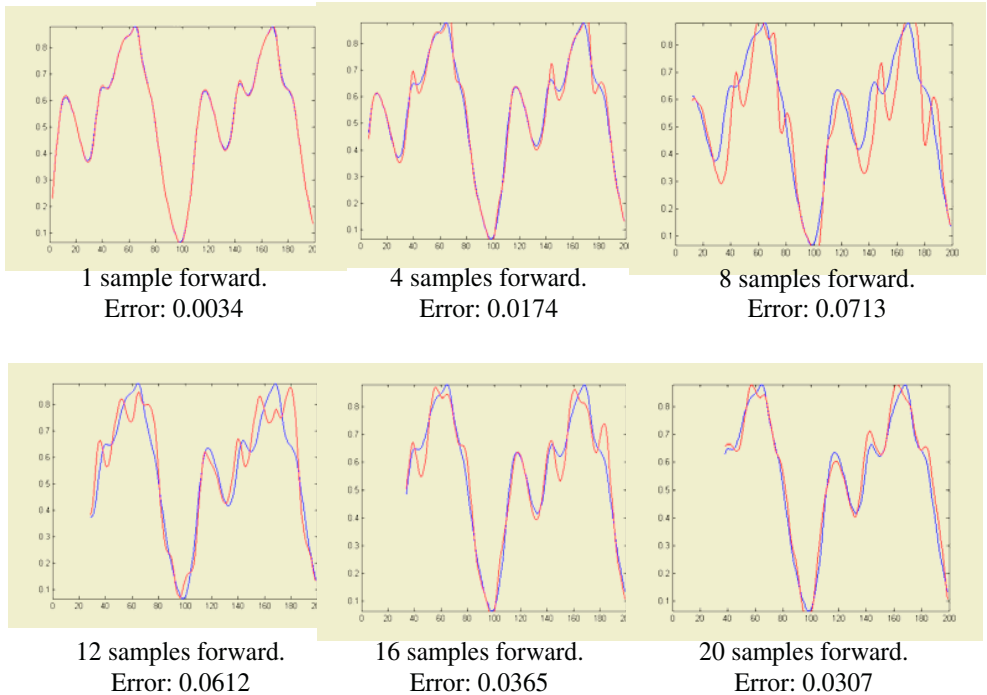
**Fig. 3.** Best output signals for advances of 1, 4, 8, 12, 16 and 20 samples

### 4.2 Test

These results have been tested with a file containing other 200 samples of the Mackey-Glass time series. Fig. 4 shows the best signals obtained in the test phase for values of  $M = 1, 4, 8, 12, 16$  and  $20$  samples forward. These signals were built with

the expressions obtained in the training phase with values of  $N = 2, 3, 5, 18, 19$  and  $19$  respectively. These were the values that gave best results in the test phase for those values of  $M$ .

Fig. 4 shows that the expressions obtained can return good results, although the values of the mean error are slightly higher than in the training phase, with parts of the signal where they were not trained.



**Fig. 4.** Best output signals for advances of 1, 4, 8, 12, 16 and 20 samples in the test

## 5 Conclusions

This paper presents an application of GP in the field of signal processing. Signal forecasting is a research field that has already been widely studied with a large number of different techniques. However, this paper aims at offering a step forward, making a more generic application in which the prediction is not made on the following sample from a consecutive group of samples, but instead that now it is possible to advance the prediction by a given number of samples.

If one sample forward forecasting has many undeniable real-world applications, the possibility of forecasting several samples forward allows the creation of more complete systems, because this anticipation will allow the implementation and improvement of prevention systems, alarms, etc.

As shown in the experiments carried out, it is difficult to obtain good results, because in those cases in which much anticipation in the forecast is desired, it will be necessary to take a large amount of information from the signal. This leads to the problem that the search space increases significantly, and the system becomes more likely to fall on local optima. The solution can be to increase the population size as the number of the samples taken from the signal increases. This could not be done in the experiments, because it was necessary to keep the GP parameters constant in order to be able to make coherent comparisons between the results obtained.

## 6 Future Works

From this study, the research work goes on different directions.

A possible continuation of this work is a more detailed study of the number of variables necessary to obtain a good prediction for a concrete anticipation. In this sense, it could be possible to make a regression process that returns a relationship between the anticipation desired for a prediction and the number of variables necessary to obtain it.

The use of other time series with different mathematical properties, as well as other kind of series (financial series, etc.) is interesting to make a comparison of the results and to validate the use of this technique on real-world series.

## Acknowledgments

This work was supported in part by the Spanish Department of Science and Technology (MCYT) (Ref TIC2000-0120-P4-03 and TIC2003-07593), by the Dirección Xeral de Investigación - Xunta de Galicia (Ref PGIDT04PXIC10503PN and PGIDT04PXIC10504PN) and by the Instituto de la Salud Carlos III (Red n°: G03/160).

## References

1. Danilov D.L.: Principal Components in Time Series Forecast. *Journal of Computational and Graphical Statistics*. Vol. 6, pp: 112-121. (1997)
2. Jenkins G.M., Box G.E.P.: *Time Series Analysis Forecasting and Control*. Holden Day Ed., Oakland. (1976)
3. Casdagli M.: Nonlinear Prediction of Chaotic Time Series. *Physica*. Vol. 35d, pp. 335-356. (1989)
4. Kaboudan M.: Forecasting Demand for Natural Gas Using GP-Econometric Integrated Systems. *Computing in Economics and Finance 2003*. Society for Computational Economics. (2003).
5. Dorado, J., Rabuñal, J. R., Puertas, J., Santos, A., Rivero, D.: Prediction and Modelling of the Flow of a Typical Urban Basin Through Genetic Programming. *Applications of Evolutionary Computing, Proceedings of EvoWorkshops 2002: EvoCOP, AvoIASP, EvoSTIM/EvoPLAN*. (2002)

6. Rabuñal J.R., Dorado J., Puertas J., Pazos A., Santos A., Rivero D.: Prediction and Modelling of the Rainfall-Runoff Transformation of a Typical Urban Basin using ANN and GP. *Applied Artificial Intelligence*. (2003)
7. Svarer C.: Designer Networks for Time Series Processing. *Neural Networks for Signal Processing III*. (1993)
8. Dorado J., Rabuñal J.R., Santos A., Pazos A., Rivero D.: Automatic Recurrent and Feed-Forward ANN Rule and Expression Extraction with Genetic Programming. *Parallel Problem Solving from Nature. Proceedings of PPSN VII*. (2002)
9. Wan E.A.: Finite Impulse Response Neural Networks with Applications in Time Series Prediction. PhD thesis. University of Stanford. (1993)
10. Packard N.H.: A Genetic Learning Algorithm for the Analysis of Complex Data. *Complex Systems*. 4:543. (1990)
11. Koza J.: *Genetic Programming: on the programming of computers by means of natural selection*. Cambridge, MA: MIT Press. (1992)
12. Bennet H., Koza J.R., Andre D., Keane M.A.: Evolution of a 60 Decibel op amp using Genetic Programming. *Proceedings of International Conference on Evolvable Systems: From Biology to Hardware ({ICES}-96)*. (1996)
13. Luke S. Spector L.: A Revised Comparison of Crossover and Mutation in Genetic Programming. *3rd Annual Conference on Genetic Programming, Morgan-Kauffman*, (1998)
14. Dracopoulos D.C., Kent S.: Genetic Programming for Prediction and Control. *Neural Computing and Applications*. Vol. 6, pp: 214-228. (1997)
15. Mackey, M., Glass, L.: Oscillation and chaos in physiological control systems. *Science*, pp. 197-287. (1977)

# Multi-modeling: A Different Way to Design Intelligent Predictors

Kurosh Madani<sup>1</sup>, Lamine Thiaw<sup>1,3</sup>, Rachid Malti<sup>2</sup>, and Gustave Sow<sup>3</sup>

<sup>1</sup> Intelligence in Instrumentation and Systems Laboratory (I<sup>2</sup>S),  
Senart Institute of Technology, University PARIS XII,  
Av. Pierre Point, F-77127 Lieusaint, France  
madani@univ-paris12.fr, thiawlamine@yahoo.fr

<sup>2</sup> Automation, Productic, Signal and Image Laboratory (LAPS / UMR5131),  
BORDEAUX I University, 351 Cours de la Liberation,  
F-33405 Talence, France  
rachid.malti@laps.u-bordeaux1.fr

<sup>3</sup> Renewable Energies Laboratory (Laboratoire d'Energies Renouvelables LER),  
Dakar Polytechnic University, Cheikh Anta Diop University,  
BP 5085, Dakar Fan, Senegal  
gsow@ucad.sn

**Abstract.** Recently, multiple works proposed multi-model based approaches to model nonlinear systems. Such approaches could also be seen as some “specific” approach, inspired from ANN operation mode, where each neuron, represented by one of the local models, realizes some higher level transfer function. We are involved in nonlinear dynamic systems identification and nonlinear dynamic behavior prediction, which are key steps in several areas of industrial applications. In this paper, two identifiers architectures issued from the multi-model concept are presented, in the frame of nonlinear system’s behavior prediction context. The first one, based on “equation error” identifier, performs a prediction based on system’s inputs and outputs. However, if the system’s inputs are often accessible, its outputs are not always available in prediction phase. The second one, called “output error” based identifier/predictor needs only the system’s inputs to achieve the prediction task. Experimental results validating presented multi-model based structures have been reported and discussed.

## 1 Introduction

Identification of nonlinear systems is an important task for model based control, system design, simulation, prediction and fault diagnosis. The identification task involves two essential steps: structure selection and parameter estimation. These two steps are linked and generally have to be performed in order to achieve the best compromise between error minimization and the total number of parameters in the final model. In real world applications (real world situations), strong nonlinearity and large number of related parameters make the realization of those steps challenging, and so, the iden-

tification task difficult. A large variety of structures to take into account systems non-linearity have already been proposed among which, Wiener and Hammerstein type models [1], Volterra series [2], Fuzzy logic based models [3], [4] and especially in last decades, neural network based approaches, which applications are numerous in dynamical system modeling [6]. However, a number of real world applications, especially those dealing with nonlinear systems behavior prediction, require refined generalization capabilities, which are not always ensured with simple artificial neural networks issued solutions.

Recently, a multiple works treat multi-model based approaches to avoid difficulties (modeling complexity) related to non-linearity ([7], [8] [9] and [12]). The use of multi-model for identifying complex systems is due to the fact that it is not always possible to find an analytical relation which describes the system's behavior in its whole operating range. The complexity of the system can be considerably reduced if the operating range of the system is divided into different regions in which its local behavior could be described with relatively simple functions. This approach is at the base of multi-model theory.

In a general way, in multi-models based approaches, a set of models, corresponding to a set of operating ranges contributes to identify the whole system. In other words, such approach could appear as some kind of weighted contribution of a set of models approximating the whole system's behavior. Thus, it could also be seen as some "specific" approach, inspired from ANN operation mode, where each neuron (here represented by one of the local models) realizes some higher level transfer function. Concerning the learning process (taking into account the analogy with learning in ANN), it could be performed on the level of contribution rate of each model to the global response, called also "activation degree" associated to each of participant models in such multi-model architecture. Several strategies could be exploited to determine "activation degree" (validation interval) of each model in such multi-model structures. All of them operate on the basis of the whole system's feature space (operation space) partitioning. What differs is the way of the partitioning. The most elementary partitioning strategy is the "grid partitioning", where a prearranged clustering chart maps the system's feature space dividing it into several partitions associating to each of obtained partitions a local model describing the system's behavior in that interval. However, if the implementation facility is the main advantage of this simple strategy, the appropriated choice of each interval reminds its principle drawback. Among promising strategies to overcome this difficulty are: "decision tree construction" (DTC - a deterministic partitioning approach) and "fuzzy clustering" partitioning (FC - a probabilistic approach). The clusters construction in both of these partitioning strategies is based on identification error minimization.

In this paper we present and compare two identifiers architectures issued from the multi-model concept, in the frame of nonlinear system's behavior prediction dilemma. In the first one, both identification (system's modeling) and prediction tasks are based on the mutual usage of system's inputs and outputs. However, in real-world applications, especially those concerning the behavior prediction area, it is not always possible to access system's output during the prediction phase. So, if this first identification/prediction architecture could be successfully applied in a



number of fields related to the controllers design, it is not always appropriated for other behavior prediction applications. In the second one, only the identification task takes partially advantage from the system’s outputs. In fact, during the prediction operation, the system’s outputs are supposed to be inaccessible. System’s feature space partitioning has been based on Decision Tree Construction (DTC) and Fuzzy Clustering (FC) strategies.

The paper is structured as follow: first, the principle of Multi-Model approach is presented. Then, the application of such multi-model based architecture to the non linear systems identification and their behavior prediction is presented. Next, the multi-model architectures are validated giving experimental results. Finally, perspectives of the presented work conclude this paper.

## 2 Multi-modeling Principle

A multi-model is composed of several models each of which is valid in a well defined interval which corresponds to a part of the operation range of the system or covers a part of the whole feature space of the problem to be solved. The local validity of a model in a well defined interval is specified by using functions with limited supports which tend to significantly increase the contribution of the local models in that zone and tend to decrease it elsewhere. The combination of all local models allows description of the whole system’s behavior. The local models participations in the multi-model’s output are quantified (determined) by “activation degree” associated to each local model. The action of “activation degrees” on multi-model’s response could be seen as some kind of local models responses weighting fashioning its response in order to approximate the modeled behavior.

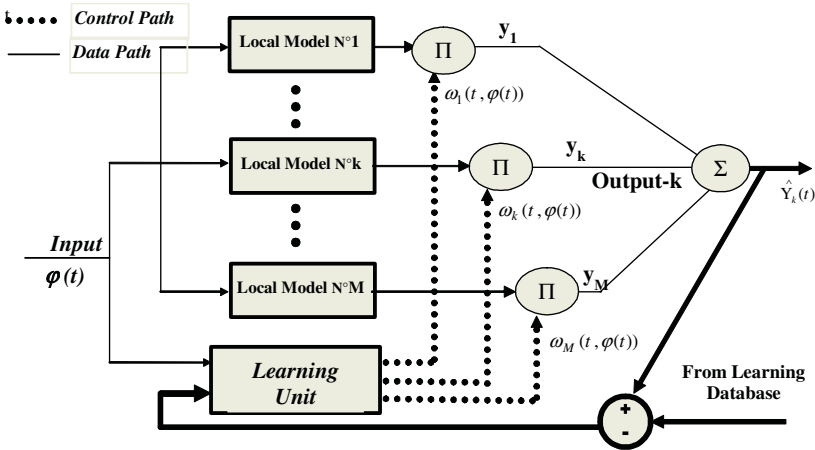


Fig. 1. General bloc diagram of a multi-model and the associated learning process

Consider a non-linear dynamical system described by the general equation, expressed by relation (1), where  $F(\cdot)$  represents a global unknown model (system to be identified) and  $\varphi(t)$  is a regression vector composed by a number of delayed system's inputs and outputs. The associated multi-model, composed by  $M$  local models and their weights  $\rho_i(\varphi(t), \beta_i)$ , with  $\rho_i(\varphi(t), \beta_i) \geq 0$  (for all  $i$ ) and  $\sum_{j=1}^M \rho_j(\varphi(t), \beta_j) > 0$  (for all  $\varphi(t)$ ), is defined by the weighted average expressed in the relation (2). In this relation  $f_i(\varphi(t))$  represents the  $i$ -th local model and  $\beta_i$  is a parameter related to the validity function  $\rho_i$ .

$$y(t) = F[\varphi(t)] \tag{1}$$

$$\hat{y}(t) = \frac{\sum_{j=1}^M \rho_j(\varphi(t), \beta_j) f_j(\varphi(t))}{\sum_{j=1}^M \rho_j(\varphi(t), \beta_j)} \tag{2}$$

Defining the “activation degree” of the  $i$ -th local model conformably to the relation (3), the general relation for multi-model architecture could then be written according to the relation (4), where  $\omega(\cdot)$  is the “activation degree” of the  $i$ -th local model.

$$\omega_i(\varphi(t), \beta) = \frac{\rho_i(\varphi(t), \beta_i)}{\sum_{j=1}^M \rho_j(\varphi(t), \beta_j)} \tag{3}$$

$$\hat{y}(t) = \sum_{j=1}^M \omega_j(\varphi(t), \beta_j) f_j(\varphi(t)) \tag{4}$$

The general concept of Multi-modeling could be seen as some kind of artificial neural network structure, where the neurons' activation functions are those corresponding to the local models' transfer functions. The appropriated choice of the “activation degree” parameters could be seen as some learning procedure. Generally, the proposed multi-model structures use conventional (polynomial) local models. However, local models could be any kind of models. Figure 1 shows the general bloc diagram of a multi-model as well as the general learning process in such structures.

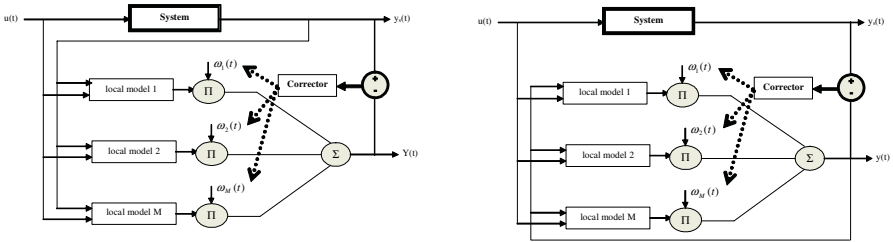
### 3 Multi-model Based Identifier/Predictor Structures

System identification with multi-models can be done in two steps: structural identification and parametric identification. Structural identification allows the specification of the operating zones and the structure of each local model. The specification of the operating zones consists of determining the characteristic space “ $Z$ ” and its decomposition into different zones, each of which have a validity function. In each step of the decomposition, a new zone is added so that the number of local models is increased by one. The decomposition

can be orthogonal [10]. The specification of the structure of each local model consists of determining the relation which describes the model in its validity domain.

Depending on the regression vector construction, one can have two types of multi-model: “equation error based” multi-model (EE), depicted in the left block diagram of figure 2 and “output error” based multi-models (OE), shown in the right bloc diagram of the same figure. The parameter estimations result from the minimization of the error between the system output and the multi-model output. One can plan to minimize the error between the system output (in a specific interval) and the local models outputs (local learning) or to minimize the error between the system output and the multi-model output (global learning). We used the global learning, which is performed according to the relation (5) criterion, where  $N$  is number of measurements.

$$J_g = \frac{1}{N} \cdot \sum_{t=1}^N \left( \hat{y}(t) - y_s(t) \right)^2 \tag{5}$$



**Fig. 2.** Learning mechanism bloc diagram for equation error (left) and for output error (right) based multi-models

According to the criterion nature  $J$ , one distinguishes two types of optimization:

- Linear optimization: if the criterion  $J$  is quadratic with respect to the vector parameter  $\theta$  of the local models, one can use the least square method to set the parameters. In general the regression vector contains some delayed outputs which make the criterion not quadratic. However by replacing these delayed outputs by the corresponding measurements, the criterion becomes quadratic.
- Non linear optimization: if the model is non-linear, the parameters can't be computed analytically and iterative techniques of optimisation are to be used. The estimation  $\hat{\theta}(k + 1)$  of  $\theta$  at the  $k + 1$ -th iteration is obtained by the relation (6), where  $\hat{\theta}(k)$  – estimation of  $\theta$  obtained at the  $k$ -th iteration,  $\eta_k$  – “relaxation” control parameter,  $G_k$  – estimation of the gradient of the criterion at the iteration  $k$ ,  $H_k$  – matrix which modifies the search direction. Its choice defines the non-linear optimization method. Levenberg – Marquardt method is used for which  $H_k$  is the regularised Hessian matrix expressed by relation (7), where  $Ha_k$  – the approximated Hessian,  $\lambda_k$  – coefficient of regularization,  $I$  – identity matrix.

$$\hat{\theta}(k+1) = \hat{\theta}(k) - \eta_k \cdot H_k^{-1} \cdot G_k \tag{6}$$

$$H_k = H a_k + \lambda_k \cdot I \tag{7}$$

### 4 Validation Results

Software tool was created under Matlab environment, using an object oriented programming, to validate the above-presented multi-model based identifier/predictor architectures. A benchmark based on a dynamic non linear system, described by relation (8) where  $u_i(t)$  are system's inputs and  $Y_s(t)$  its output  $u_i(t)$  take random value in interval  $[0, 1]$ :  $u_1 \in [0, 0.990]$ ,  $u_2 \in [0, 0.985]$  and  $y_s \in [-0.703, 0.294]$ , was created to validate the presented multi-models performances. The regression vector of this system is  $\varphi(t) = [u_1(t-1) \ u_2(t-1) \ Y_s(t-1)]$ .

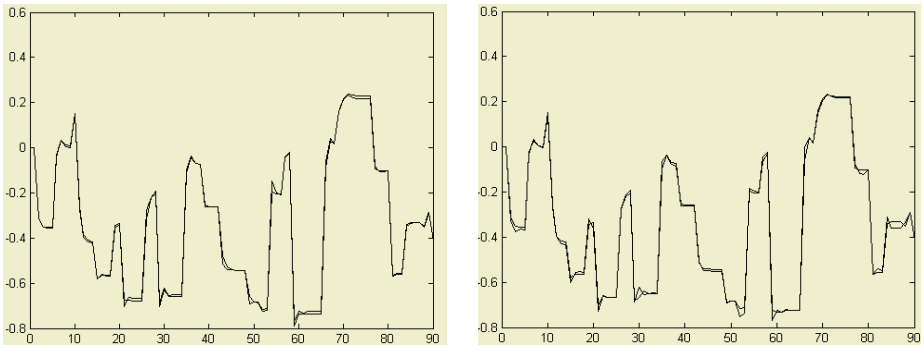
$$Y_s(t) = \frac{Y_s(t-1)[0.5u_1(t-1) - 0.3u_2(t-1)]}{1 + y^2(t-1)} + 0.3u_1^2(t-1) - 0.5u_2^2(t-1) \tag{8}$$

**Table 1.** Validation results relative to the local models parameters estimation, inputs and outputs validation intervals

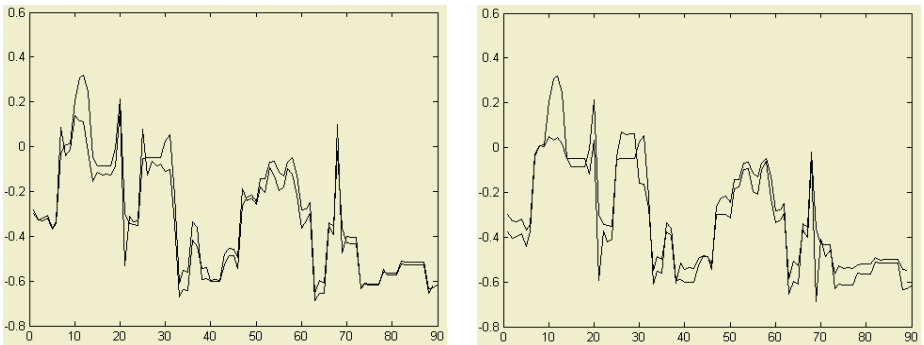
| Multi-model Structure + Partitioning | Num. of Local models | Local Models Parameters |        |        |        | Learning & Testing Errors |                     |
|--------------------------------------|----------------------|-------------------------|--------|--------|--------|---------------------------|---------------------|
|                                      |                      | a                       | b      | c      | d      | Learn.                    | Test                |
| EE + DTC                             | 5                    | 0.078                   | -1.387 | -0.217 | 0.511  | $9.4 \cdot 10^{-5}$       | $1.5 \cdot 10^{-4}$ |
|                                      |                      | 0.314                   | -0.088 | 0.423  | 0.016  |                           |                     |
|                                      |                      | 0.397                   | -0.578 | 0.355  | -0.238 |                           |                     |
|                                      |                      | -0.182                  | -0.900 | -0.146 | -0.090 |                           |                     |
|                                      |                      | 0.010                   | -0.569 | -0.152 | 0.032  |                           |                     |
| EE + FC                              | 6                    | 0.436                   | -0.827 | 0.320  | 0.001  | $1.2 \cdot 10^{-4}$       | $2.5 \cdot 10^{-4}$ |
|                                      |                      | 0.139                   | -0.265 | 0.284  | -0.024 |                           |                     |
|                                      |                      | 0.798                   | -0.707 | 0.585  | -0.394 |                           |                     |
|                                      |                      | 0.075                   | -1.630 | -0.473 | 0.503  |                           |                     |
|                                      |                      | -0.249                  | -0.830 | -0.122 | 0.186  |                           |                     |
| OE + DTC                             | 3                    | 0.405                   | -0.235 | 0.442  | -0.156 | $5.9 \cdot 10^{-3}$       | $5.3 \cdot 10^{-3}$ |
|                                      |                      | 0.267                   | -0.747 | -0.066 | -0.115 |                           |                     |
|                                      |                      | 0.302                   | -0.552 | -0.088 | 0.107  |                           |                     |
| OE + FC                              | 2                    | 0.178                   | -0.656 | -0.018 | -0.141 | $9.0 \cdot 10^{-3}$       | $1.0 \cdot 10^{-2}$ |
|                                      |                      | 0.128                   | -0.081 | -0.052 | 0.091  |                           |                     |
|                                      |                      | -0.143                  | -0.383 | 0.023  | 0.314  |                           |                     |

We have considered an EE and OE multi-models based architectures (figure 2) with linear local models described by  $y_i = a_i u_1 + b_i u_2 + c_i \hat{y} + d_i$  where  $a_i$ ,  $b_i$ ,  $c_i$  and  $d_i$

are local model's parameters (to be optimized) and with a global learning. The learning procedure operates as follows: an observation matrix, including inputs and outputs (system's outputs for EE based multi-model and estimated outputs for the OE based one) is updated progressively until the output's value stabilizes. For both cases, we have considered "decision tree construction" (DTC) and "fuzzy clustering" (FC) partitioning strategies. The FC partitioning has been based on FCM (Fuzzy Centers Mean) algorithm. The feature space partitioning is accepted if it reduces the global error. The learning process stops if the feature partitioning process doesn't lead to a new lower error between estimated and system's outputs. The EE based multi-model uses the linear criterion. The OE based multi-model uses the nonlinear criterion with "Levenberg-Marquardt" optimization algorithm. Table 1 gives validation results relative to each of two multi-models. For each space partitioning strategy, local models parameters, learning and testing errors are reported. Figures 3 and 4 compare estimated (predicted) output with the system's one for each multi-model based predictor.



**Fig. 3.** System's and Multi-model issued outputs using EE scheme: results obtained with decision tree partitioning strategy (left) and with fuzzy clustering partitioning strategy (right)



**Fig. 4.** System's and Multi-model issued outputs OE scheme: results obtained with decision tree partitioning strategy (left) and with fuzzy clustering based partitioning strategy (right)

One can remark that system's output prediction with EE based multi-model structure leads to better results. In fact, the usage of system's output makes the prediction more accurate. However, system's output availability in real world prediction context remains restrictive and often unrealistic. That's why, even if to operate correctly (with acceptable approximation and prediction performances) the OE based multi-model structures seem to require more complex learning and optimization mechanisms, they remain the most effective to solve prediction problems. The OE based prediction could be enhanced by using hybrid scheme where the learning (identification) is performed on the basis of an EE model and the prediction on the basis of an OE model.

## 5 Conclusion

The Multi-modeling general concept could be seen as some kind of artificial neural network structure, where the neurons activation functions are those corresponding to the local models transfer functions. Two multi-model based identifier/predictor structures have been presented. If prediction performances obtained for the first one (equation error based multi-model), which uses both system's inputs and outputs, seem to be better than those obtained for the second one (using only input of the system), its usage concerns a limited number of real world cases. In fact, it is not always possible to access system's output (during the prediction phase) in the case of real world behavior prediction applications. On the other hand, the performances of the "output error" based identifier/predictor could be enhanced on one hand by increasing local models complexity (for example, by using higher order local models) and on the other hand, by improving feature space partitioning strategy and learning mechanism. These features delineate future perspectives of the present work.

## References

1. N. Wiener, Non linear Problems in Random Theory, *Technology Press MIT*, and *John Wiley, New York* (1958).
2. M. Schetzen, The Volterra and Wiener Theories of Nonlinear Systems, *John Wiley, New York*, (1980).
3. L. Zadeh, Outline of a New Approach to the Analysis of Complex Systems and Decision Processes, *IEEE Trans. On Systems, Man and Cybernetics* 3, pp. 28-44.
4. T. Takagi and M. Sugeno, Fuzzy identification of systems and its application to modeling and control. *IEEE Trans. on Systems Man and Cybernetics*, Vol. 15, (1985). pp. 116-132.
5. Boukhris, A, Mourot G. and Ragot J. (2000). Nonlinear dynamic system identification: a multiple-model approach. *Int. J. of control*, Vol. 72, N°7/8, pp. 591-604.
6. K.S. Narendra, and K. Parthasarath., Identification and control of dynamical systems using neural networks, *IEEE Trans. Neural Networks*, Vol. 1, No. 1, (1990).
7. O. Nelles, On the identification with neural networks as series-parallel and parallel models, *Int. Conf. on Artificial Neural Networks (ICANN'95), Paris, France*, (1995).
8. *Multiple Model Approaches to Modeling and Control*, edited by R. Murray-Smith and T.A. Johansen, *Taylor & Francis Publishers*, (1997), ISBN 0-7484-0595-X.

9. M. Mayoubi, M. Schafer, S. Sinsel, Dynamic Neural Units for Non-linear Dynamic Systems Identification, *LNCS Vol. 930, Springer Verlag*, (1995), pp.1045-1051.
10. S. Ernst, Hinging hyper-plane trees for approximation and identification, *37<sup>th</sup> IEEE Conf. on Decision and Control, Tampa, Florida, USA*, (1998).
11. K. Madani, M. Rybnik, A. Chebira, Data Driven Multiple Neural Network Models Generator Based on a Tree-like Scheduler, *LNCS series, Edited by: J. Mira, A. Prieto - Springer Verlag* (2003), ISBN 3-540-40210-1, pp. 382-389
12. K. Madani, M. Rybnik, A. Chebira, Non Linear Process Identification Using a Neural Network Based Multiple Models Generator, *LNCS series, Edited by: J. Mira, A. Prieto - Springer Verlag*, (2003). ISBN 3-540-40211-X, pp. 647-654

# Input and Structure Selection for $k$ -NN Approximator

Antti Sorjamaa, Nima Reyhani, and Amaury Lendasse

Neural Network Research Centre,  
Helsinki University of Technology, P.O. Box 5400,  
02015 Espoo, Finland  
{asorjama, nreyhani, lendasse}@cis.hut.fi

**Abstract.** This paper presents  $k$ -NN as an approximator for time series prediction problems. The main advantage of this approximator is its simplicity. Despite the simplicity,  $k$ -NN can be used to perform input selection for nonlinear models and it also provides accurate approximations. Three model structure selection methods are presented: Leave-one-out, Bootstrap and Bootstrap 632. We will show that both Bootstraps provide a good estimate of the number of neighbors,  $k$ , where Leave-one-out fails. Results of the methods are presented with the Electric load from Poland data set.

**Keywords:**  $k$ -NN, Time Series Prediction, Bootstrap, Leave-one-out and Model Structure Selection.

## 1 Introduction

In any function approximation, system identification, classification or prediction task one usually wants to find the best possible model and the best possible parameters to have good performance. Selected model must be generalizing enough still preserving accuracy and reliability without unnecessary complexity, which increase computational load and thus calculation time. Optimal parameters must be determined for every model to be able to rank the models according to their performances.

Furthermore, in order to select the best model, or model class, one also needs to determine the best input set, to use in the determination of the best parameters for each structure. When too many inputs are selected, it is possible to get even worse results than with fewer inputs containing more accurate and valid information. Vice versa, with only few inputs the accuracy of the model might not be enough and results are poor and unreliable.

The problems mentioned above occur simultaneously, so it is difficult to find the right combination of correct attributes. It can be a very tedious and time-consuming procedure to go through every possible structure to select the best one.

In this paper, we focus in finding the optimal structure for  $k$ -Nearest Neighbors ( $k$ -NN) approximator. At the same time we consider the problem of selecting the most necessary and optimal inputs for the  $k$ -NN as well as selecting the structure of the approximator. The  $k$ -NN method is presented in Section 2; Section 3 describes the selection of the inputs with an exhaustive search and Section 4 the selection of the



structure using Leave-one-out (LOO) and Bootstraps. In Section 5 we show some experimental results with an electric load time series and then derived conclusions from the results in Section 6.

## 2 $k$ -Nearest Neighbors

$k$ -Nearest Neighbors approximation method is a very simple, but powerful method. It has been used in many different applications and particularly in classification tasks [1].

The key idea behind the  $k$ -NN is that similar input data vectors have similar output values. One has to look for a certain number of nearest neighbors, according to Euclidean distance [1], and their corresponding output values to get the output approximation. We can calculate the estimation of the outputs by using the average of the outputs of the neighbors in the neighborhood.

If the pairs  $(x_i, y_i)$  represent the data with  $x_i$  as an  $n$ -dimensional input and  $y_i$  as a scalar output value,  $k$ -NN approximation is

$$\hat{y}_i = \frac{\sum_{j=1}^k y_{P(j)}}{k}, \quad (1)$$

where  $\hat{y}_i$  represents the output estimation,  $P(j)$  is the index number of the  $j^{\text{th}}$  nearest neighbor of the input  $x_i$  and  $k$  is the number of neighbors that are used.

We use the same neighborhood size for every data point, so we use a global  $k$ , which must be determined.

## 3 Input Selection

In order to select the best set of inputs, all possible  $2^n$  ( $n$  is the maximum number of inputs) input sets are built and evaluated. For each input set the global optimum number of neighbors is determined and the generalization error estimate (defined in Section 4) of the set is calculated as a mean of errors of all data points. In this way it is possible to compare all different input sets and take the best one to be used in the final  $k$ -NN approximation.

In this case, adding one input doubles the needed calculation time. We have to make a compromise between the maximum input size to use and the calculation time available. This kind of exhaustive search for best inputs is usually not preferred, because of the huge computational load. However, with  $k$ -NN the computations can be performed in a reasonable time, thanks to the simplicity of the  $k$ -NN.

## 4 Model Structure Selection

We consider the problem of determining a model which approximates as accurately as possible an unknown function  $g(\cdot)$ . This approximation is chosen among a set of several possible models. Models in a set are denoted here by

$$h^q(x, \theta(q)) , \tag{2}$$

where  $q$  represents the  $q^{\text{th}}$  model in the set,  $\theta(q)$  are the parameters of the  $q^{\text{th}}$  model and  $x$  is a  $n$ -dimensional input vector. The parameters that define a set of possible models are called hyper-parameters; they are not estimated by the learning algorithm, but by some external procedure [4].

In a typical learning procedure, the  $\theta(q)$  parameters are optimized to minimize the approximation error on the learning set; the structure is determined as the minimization of the *generalization error* defined as

$$E_{gen}(q, \theta) = \lim_{M \rightarrow \infty} \frac{\sum_{i=1}^M (h^q(x_i, \theta(q)) - y_i)^2}{M} , \tag{3}$$

where  $x_i$  are  $n$ -dimensional input vectors to the model and  $y_i$  the corresponding scalar expected outputs.

According to the definition (3), the generalization error is the mean square error of the model, computed on an infinite sized test set. Such set is not available in practice, so we must approximate the generalization error. The best model structure  $q$  is the structure that minimizes the approximation of the generalization error.

In our case the model parameters consist in selecting the number of neighbors to use in the  $k$ -NN approximation. We have used two different methods to select the global optimal number of neighbors, Leave-one-out (LOO) and Bootstrap 632.

Leave-one-out is a common method used in many statistical evaluation purposes and we wanted to show that Bootstrap 632 is better than LOO in this case, even [2] claims that Bootstrap 632 doesn't work in selecting the number of neighbors.

### 4.1 Leave-One-Out

Leave-one-out [3] is a special case of  $k$ -fold cross-validation resampling method. In  $k$ -fold cross-validation the training data is divided into  $k$  approximately equal sized sets. Then model is trained by using all but one set and the leftover set is used in validation. The generalization error estimation of  $k$ -fold cross-validation is a mean of all  $k$  different validation results.

LOO procedure is the same as  $k$ -fold cross-validation with  $k$  equal to the size of the training set  $N$ . So, for each different neighborhood size, LOO procedure is used to calculate its generalization error estimate by removing each neighbor at a time from the training set, building a model with the rest of the training data and calculating the validation error with the one taken out. This procedure is done for every data point in the training set and the estimate of the generalization error is calculated as a mean of all  $k$ , or  $N$ , validation errors (4).

$$\hat{E}_{gen}(q) = \frac{\sum_{i=1}^N (h^q(x_i, \theta_i^*(q)) - y_i)^2}{N} , \tag{4}$$

where  $x_i$  is the  $i^{\text{th}}$  input vector from the training set,  $y_i$  is the corresponding output and  $\theta_i^*(q)$  includes the model parameters without using  $(x_i, y_i)$  in the training.

Because we want to use the global optimum size of the neighborhood, we have to calculate the LOO error for each data point and each size of the neighborhood. After that we can take the mean over all data points to find out, which is globally the optimum number of neighbors. We select the number of neighbors that gives us the smallest generalization error.

### 4.2 Bootstrap and Bootstrap 632

Bootstrap [4] is a resampling technique developed to estimate some statistical parameters (like the mean of a population, its variance, etc). In the case of a model structure selection, the parameter to be estimated is the generalization error.

When using bootstrap, the generalization error is not estimated directly. Rather the bootstrap estimates the difference between the generalization error and the training error, or apparent error according to Efron [2]. This difference is called *the optimism*. The estimation of the generalization error will be the sum of the training error and the estimated optimism.

The training error is computed using all available data on the training set.

$$E^{I,I}(q, \theta^*(q)) = \frac{\sum_{i=1}^N (h^q(x_i^I, \theta^*(q)) - y_i^I)^2}{N}, \tag{5}$$

where  $h^q$  is the  $q^{\text{th}}$  model that is used,  $I$  denotes the training set,  $\theta^*(q)$  includes the model parameters after learning,  $x_i^I$  is the  $i^{\text{th}}$  input vector from the training set,  $y_i$  is the corresponding output and  $N$  is the number of elements in the training set.

The optimism is estimated using a resampling technique, based on drawing with replacement within the training set. This *bootstrap set* is as large as the training set with its participants drawn randomly from the training set. Each model is trained using the bootstrap set and optimism is calculated as the difference between the learning error (6) and the validation error (7).

Learning error is calculated in the bootstrap set with model trained in the same bootstrap set.

$$E_j^{A_j, A_j}(q, \theta_j^*(q)) = \frac{\sum_{i=1}^N (h^q(x_i^{A_j}, \theta_j^*(q)) - y_i^{A_j})^2}{N}, \tag{6}$$

where  $A_j$  is the  $j^{\text{th}}$  bootstrap set,  $x_i^{A_j}$  is the  $i^{\text{th}}$  input vector from the bootstrap set and  $y_i^{A_j}$  is the corresponding output.

Validation error is calculated in the initial training set with model trained on the same bootstrap set than the learning error.

$$E_j^{A_j, I}(q, \theta_j^*(q)) = \frac{\sum_{i=1}^N (h^q(x_i^I, \theta_j^*(q)) - y_i^I)^2}{N}. \tag{7}$$

Above described optimism calculation procedure is repeated as many times, or rounds, as possible considering linearly increasing computation time. The optimism

of a model is then calculated, as a mean of the difference of the two error functions described above

$$opti \hat{m}ism(q) = \frac{\sum_{j=1}^J E_j^{A_j, I}(q, \theta_j^*(q)) - E_j^{A_j, A_j}(q, \theta_j^*(q))}{J}, \tag{8}$$

where  $J$  is the number of bootstrap rounds done.

The final generalization error estimate is the sum of the training error and the optimism

$$\hat{E}_{gen}(q) = opti \hat{m}ism(q) + E^{I, I}(q, \theta^*(q)). \tag{9}$$

Bootstrap 632 [5] is a modified version of the original Bootstrap. Where the original Bootstrap gives biased estimation of the generalization error (3), Bootstrap 632 is not biased [4] and thus is more comparable with other methods estimating the generalization error. Bootstrap 632 converges towards the correct generalization error in a reasonable amount of calculation time.

The main difference between standard Bootstrap and Bootstrap 632 is the estimation of optimism. In original Bootstrap the optimism is calculated as difference of the errors in two sets, in the initial training data set and in the randomly drawn bootstrap set (8). Bootstrap 632 estimates optimism using the data points not drawn into the bootstrap set (10). The model is trained using this set of unselected data points and its error is evaluated on the bootstrap set.

$$opti \hat{m}ism^{632}(q) = \frac{\sum_{j=1}^J E_j^{\bar{A}_j, A_j}(q, \theta_j^*(q))}{J}, \tag{10}$$

where the error function is the same as the learning error (6), except the model is trained using  $\bar{A}_j$ , the complement of the bootstrap set  $A_j$ .

The estimation of the generalization error of the Bootstrap 632 is calculated as weighted sum of the training error and the new optimism (10).

$$\hat{E}_{gen}(q) = .368 opti \hat{m}ism^{632}(q) + .632 E^{I, I}(q, \theta^*(q)). \tag{11}$$

From equation (11) it gets quite clear to see, that the name of the Bootstrap 632 comes from the weighting coefficient of the training error term. The value 0.632 is the probability of one sample to be drawn to the bootstrap set from the training set [2, 5].

## 5 Experimental Results

### 5.1 Time Series Prediction

Time series forecasting [6] is a challenge in many fields. In finance, one forecasts stock exchange courses or stock market indices; data processing specialists forecast the flow of information on their networks; producers of electricity forecast the load of

the following day. The common point to their problems is the following: how can one analyse and use the past to predict the future?

Next section will demonstrate how  $k$ -NN can be used to predict future values of a time series.

### 5.2 Results of Data Set 1: Electric Load

The dataset used in experiments is a benchmark in the field of time series prediction: The Poland Electricity Dataset. It represents the electric load of Poland during 2500 days in the 90's.

In our experiments we used the first half of the data set as a training set and the other half as a test set in order to evaluate the selected model's performance between different methods.

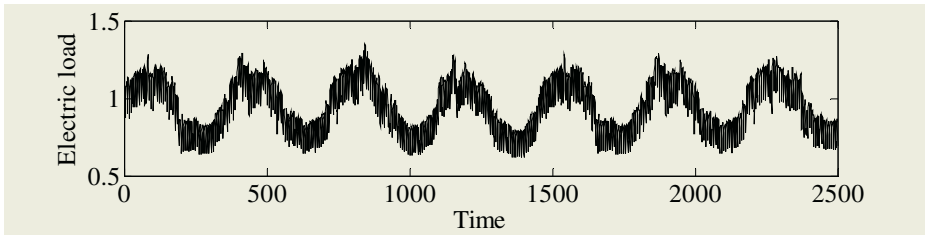


Fig. 1. Electric load time series from Poland

Fig. 2 and 3 show the generalization error estimates of all different methods according to the number of neighbors, when using the best selected input set.

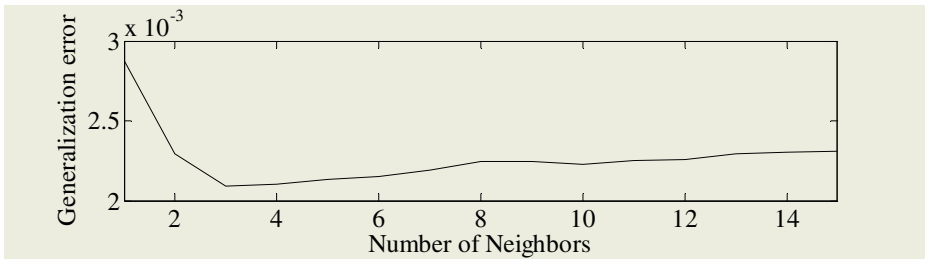
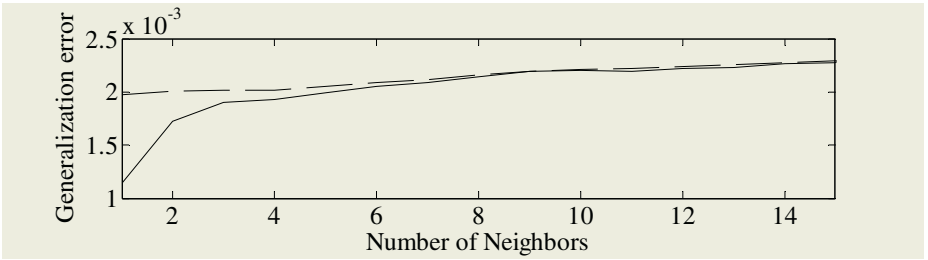


Fig. 2. The generalization error estimate using Leave-one-out according to the number of neighbors

The Table 1 shows the results of the experiments with three methods in selecting the inputs and number of neighbors. We have used  $n = 8$  as a maximum number of inputs and  $J = 100$  bootstrap rounds in calculations in both, Bootstrap and Bootstrap 632.



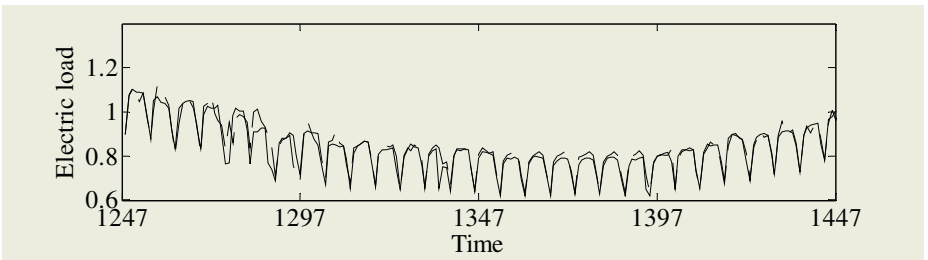
**Fig. 3.** The generalization error estimate using Bootstrap (solid line) and Bootstrap 632 (dashed line) according to the number of neighbors

**Table 1.** The results using Electric load data set and methods described in this paper

|               | Selected Inputs         | $k$ | $\hat{E}_{gen}$ | Test error |
|---------------|-------------------------|-----|-----------------|------------|
| LOO           | $t - \{1, 2, 5, 7, 8\}$ | 3   | 0.0021          | 0.0011     |
| Bootstrap     | $t - \{1, 2, 5, 7, 8\}$ | 1   | 0.0011          | 0.0007     |
| Bootstrap 632 | $t - \{1, 2, 5, 7, 8\}$ | 1   | 0.0020          | 0.0007     |

All three methods select the same inputs but different number of neighbors. According to the test error both, Bootstrap and Bootstrap 632, select the best  $k$ . On the other hand, Bootstrap and Bootstrap 632 use  $J$  times more time than LOO.

In Fig. 4 we have used the model selected by bootstrap to predict 300 first test set values.



**Fig. 4.** Test set from Electric load data. 200 first values predicted and plotted. Solid line represents the real values and dashed represents the prediction

## 6 Conclusion

We have shown, that all methods, Leave-one-out and Bootstraps, select the same inputs. But number of neighbors is selected more efficiently by Bootstraps, according to the test error; even some studies have proven otherwise [2].

It has also been shown, that  $k$ -NN is a good approximator for time series. We have also tested  $k$ -NN with a couple of other time series and acquired the same results. As a conclusion we suggest Leave-one-out to be used in input selection and Bootstrap or Bootstrap 632 in selection of  $k$ .

## Acknowledgements

Part the work of A. Sorjamaa, N. Reyhani and A. Lendasse is supported by the project of New Information Processing Principles, 44886, of the Academy of Finland.

## References

1. Bishop C.M.: Neural Networks for Pattern Recognition. Oxford University Press (1995).
2. Efron B., Tibshirani R. J.: An introduction to the bootstrap. Chapman & Hall (1993).
3. Kohavi R.: A study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. In: Proc. of the 14th Int. Joint Conf. on A.I., Montréal (1995) 2:1137-1143.
4. Lendasse A., Wertz V., Verleysen M.: Model selection with cross-validations and bootstraps – Application to time series prediction with RBFN models. In: Artificial Neural Networks and Neural Information Processing – ICANN/ICONIP (2003), Kaynak O., Alpaydin E., Oja E., Xu L. (eds): Springer-Verlag Lecture Notes in Computer Science 2714, Berlin (2003) 573-580.
5. Efron B., Tibshirani R. J.: Improvements on cross-validation: The .632+ bootstrap method. J. Amer. Statist. Assoc. (1997) 92:548–560.
6. Weigend A.S., Gershenfeld N.A.: Times Series Prediction: Forecasting the future and Understanding the Past. Addison-Wesley, Reading MA (1994).

# Nonlinear Robust Identification with $\epsilon$ -GA: FPS Under Several Norms Simultaneously

J.M. Herrero, X. Blasco, M. Martínez, and C. Ramos

Predictive Control and Heuristic Optimization Group,  
Department of Systems Engineering and Control,  
Polytechnic University of Valencia  
juaherdu@isa.upv.es  
<http://ctl-predictivo.upv.es>

**Abstract.** In nonlinear robust identification context, a process model is represented by a nominal model and possible deviations. With parametric models this process model can be expressed as the so-called Feasible Parameter Set (*FPS*), which derives from the minimization of identification error specific norms. In this work, several norms are used simultaneously to obtain the *FPS*. This fact improves the model quality but, as counterpart, it increases the optimization problem complexity resulting in a multimodal problem with an infinite number of minima with the same value which constitutes *FPS* contour. A special Evolutionary Algorithm ( $\epsilon$ -GA) has been developed to find this contour. Finally, an application to a thermal process identification is presented.

## 1 Introduction

In Robust Process Control all designs are based on a nominal process model and a reliable estimate of the uncertainty associated to this nominal model through robust identification. Uncertainty can be caused mainly by measurement noise and model error [8] (e.g. dynamics not captured by nominal model). Although uncertainty can have different sources, it always will appear as an error between model and process outputs (identification error for a specific experiment).

Different procedures for Robust Identification are based on error statistical assumptions [3] or rely on deterministic hypotheses, where the identification error, although unknown, will be bounded by a certain norm [2],[10]. That deterministic approach will be used in this work to estimate the model parameters set which keeps the identification error bounded. A parametric uncertainty related to model parameters also is assumed.

For linear in parameters models, the *FPS* (consistent with the data and the given a priori information), if it exists, is a convex polytope (in the parameter space) and it can be approximated by orthotopes, ellipsoids or parallelotopics [6]. In nonlinear models, the polytope can be non-convex even disjoint which both increase the computational complexity, above all, if the number of data increases. Also in this case, if the model is time differentiable respect to the unknown



parameters, interval computation [9] or deterministic search algorithms [7] can be used. Otherwise, heuristic optimization such as random search or Monte Carlo techniques are applied [4].

Because to find the *FPS* contour is a multimodal problem with infinite number of solutions with the same value, a specific Evolutionary Algorithm ( $\epsilon$ -GA), based on Evolutionary Algorithm for Multiobjective Optimization [1], [5], has been used. The flexibility of the algorithm permits to solve robust identification problems with several norms considered simultaneously.

## 2 *FPS* Identification

In this work the following structure is assumed for the nonlinear model:

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t), \theta), \quad \hat{\mathbf{y}}(t) = g(\mathbf{x}(t), \mathbf{u}(t), \theta) \tag{1}$$

where

- $f(\cdot), g(\cdot)$  are the nonlinear functions of the model,
- $\theta \in D \subset R^q$  is the vector of unknown model parameters,
- $\mathbf{x}(t) \in R^n$  is the vector of model states,
- $\mathbf{u}(t) \in R^m$  is the vector of model inputs,
- $\hat{\mathbf{y}}(t) \in R^l$  is the vector of model outputs.

Let  $\mathbf{E}(\theta) = \mathbf{Y} - \hat{\mathbf{Y}}(\theta)$ , where

- $\mathbf{E}(\theta)$  is the identification error,
- $\mathbf{Y}$  are the process output measurements,  $[\mathbf{y}(0), \mathbf{y}(T) \dots \mathbf{y}(NT)]$ , when the inputs  $\mathbf{U} = [\mathbf{u}(0), \mathbf{u}(T) \dots \mathbf{u}(NT)]$  are applied to the process,
- $\hat{\mathbf{Y}}$ : are the simulated<sup>1</sup> outputs  $[\hat{\mathbf{y}}(0), \hat{\mathbf{y}}(T) \dots \hat{\mathbf{y}}(NT)]$ , when the same inputs  $\mathbf{U}$  are applied to the model.

Denote  $\|\mathbf{E}(\theta)\|_{p_i}$  as the  $p_i$ -norm of the identification error. Defining  $FPS_i$  consistent with a certain norm  $i \in A := [1, 2 \dots s]$  as

$$FPS_i := \{\theta \in D : \|\mathbf{E}(\theta)\|_{p_i} \leq \delta_i, \delta_i > 0\}, \tag{2}$$

the *FPS* for all norms simultaneously is given by

$$FPS := \left\{ \bigcap_A FPS_i \right\}. \tag{3}$$

Therefore the *FPS* will be conditioned by each  $FPS_i$  and these ones by the bounds  $\delta_i$ . The selection of  $\delta_i$  and  $p_i$ -norms is based on a priori process knowledge (e.g. tolerated model error) and noise characteristics. High values for

---

<sup>1</sup> Model outputs are calculated by integrating equation (1).  $T$  is the sample time and  $N$  is the measurement number.

$\delta_i$  provide a more conservative  $FPS_i$  whereas low ones could achieve an empty  $FPS_i$ . The  $FPS$  can be characterized by its contour given by

$$C(FPS) := \{\theta \in FPS : \exists i \mid \|\mathbf{E}(\theta)\|_{p_i} = \delta_i\}. \tag{4}$$

The problem of obtaining  $C(FPS)$  is formulated as the minimization problem of cost function

$$J(\theta) := \begin{cases} \sum_B J_i \text{ if } B \neq \emptyset \\ \prod_A J_i \text{ if } B = \emptyset \end{cases} \tag{5}$$

where<sup>2</sup>

$$B := \{i \in A : \|\mathbf{E}(\theta)\|_{p_i} > \delta_i\}; \quad J_i(\theta) = \|\|\mathbf{E}(\theta)\|_{p_i} - \delta_i\|.$$

Since  $C(FPS)$  includes an infinite number of points, especially when  $q > 1$  for searching space  $D \in R^q$ , the exact  $C(FPS)$  calculation is impossible. Consequently a uniformly distributed sample of  $C(FPS)$ , called  $C^*(FPS)$  will be calculated.

### 3 Obtaining $C^*(FPS)$ with $\epsilon$ -GA

A set of some global minima of (5) sufficiently distributed to characterize  $C(FPS)$  is necessary to obtain  $C^*(FPS)$ . That problem is similar to multiobjective optimization problems and Pareto Front localization. So, a specific Evolutionary Algorithm ( $\epsilon$ -GA) based on these ideas has been developed.

The  $\epsilon$ -GA algorithm is composed of three populations (see Fig. 1).

1.  $P(t)$  is the main population with size  $Nind_P$ . It explores the searching space during the algorithm iterations ( $t$ ).
2.  $A(t)$  with size  $Nind_A$  is the archive where the solution of the optimization problem is stored.  $Nind_A$  can be variable and it will never be higher than  $Nind_{max\_A}$ .
3.  $GA(t)$  is an auxiliary population which is used in the algorithm evolution. Its population size,  $Nind_{GA}$ , is an even number.

In archive  $A(t)$ , searching space  $D$  is divided by a hypergrid of width  $\epsilon$ , in order to force the algorithm to produce just one solution at the same box (orthotope). This ensures the diversity in the archive, avoiding the convergence in one unique point or area inside  $D$ . The pseudocode of the  $\epsilon - GA$  algorithm is given by

1.  $t := 0$
2.  $A(t) := \emptyset$

---

<sup>2</sup> Notice that for all global minima  $\bar{\theta} \in C(FPS)$ ,  $J(\bar{\theta}) = 0$ . If  $FPS = \emptyset$ , minima get a  $J > 0$  and it will be necessary either to increase the value of some  $\delta_i$  or to modify the model structure to achieve an  $FPS \neq \emptyset$ .

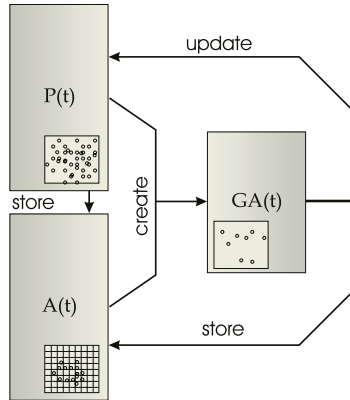


Fig. 1.  $\epsilon$  - GA algorithm structure

3.  $P(t) := ini\_random(D)$
4.  $eval(P(t))$
5.  $A(t) := store(P(t), A(t))$
6. *while*  $t < n\_iterations$  *do*
7.  $GA(t) := create(P(t), A(t))$
8.  $eval(GA(t))$
9.  $A(t + 1) := store(GA(t), A(t))$
10.  $P(t + 1) := update(GA(t), P(t))$
11.  $remove(A(t))$
12.  $t := t + 1$
13. *end while*

The main steps of the above algorithm are detailed:

- Step 3.** Population  $P(0)$  is initialized with  $Nind_P$  individuals, randomly created inside the searching space  $D$ .
- Step 4 and 8.** Function **eval** calculates the value of the cost function (5) for every individual from  $P(t)$  (step 4) and  $GA(t)$  (step 8).
- Step 5 and 9.** Function **store** analyzes whether every individual of  $P(t)$  (step 5) or  $A(t)$  (step 9) must be included in archive  $A(t)$ . An individual  $\theta_1$  is included in archive  $A(t)$  if it satisfies the following conditions:
1.  $\theta_1$  is in the same box that  $\theta_2$  (inside the hypergrid), then  $\theta_1$  replaces  $\theta_2$  iff  $J(\theta_1) < J(\theta_2)$ .
  2.  $\theta_1$  is in an empty box and  $Nind_A < Nind\_max\_A$ .
  3.  $\theta_1$  is in an empty box and  $Nind_A = Nind\_max\_A$  and  $\exists \theta_2 : J(\theta_1) < J(\theta_2)$ . In this case, when  $\theta_1$  is included in  $A(t)$ , the individual with the highest cost function value is removed. Thus, the number of individuals in  $A(t)$  is never higher than  $Nind\_max\_A$ .
- Step 7.** Function **create** creates  $GA(t)$  every iteration using the following procedure:

1. Two individuals are randomly selected,  $\theta_1$  from  $P(t)$ , and  $\theta_2$  from  $A(t)$ .
2.  $\theta_1$  and  $\theta_2$  are crossed over by the linear recombination technique and two new individuals  $\theta'_1$  and  $\theta'_2$  are created.
3.  $\theta'_1$  and  $\theta'_2$  are included in  $GA(t)$ .

This procedure is repeated  $N_{ind_{GA}}/2$  times until  $GA(t)$  population size will be reached.

**Step 10.** Function **update** updates  $P(t)$  with individuals from  $GA(t)$ . Every individual  $\theta_{GA}$  from  $GA(t)$  is randomly compared with an individual  $\theta_P$  from  $P(t)$ . If  $J(\theta_{GA}) \leq J(\theta'_P)$  then  $\theta_{GA}$  replaces  $\theta_P$ , on the other hand,  $\theta_P$  is maintained.

**Step 11.** Function **remove** deletes individuals from  $A(t)$ . Every individual  $\theta_A$  from  $A(t)$  is compared with the individual  $\theta'_P$  from  $P(t)$  with the highest cost function value . If  $J(\theta_A) > J(\theta'_P)$  then  $\theta_A$  is removed.

Finally, the solution  $C^*(FPS)$  are individuals from the final  $A(t)$ .

## 4 Robust Identification – Experimental Results

Consider a scale furnace with a resistance placed inside. A fan continuously introduces air from outside (air circulation) while energy is supplied by an actuator controlled by voltage. Using a data acquisition system, resistance temperature and air temperature are measured when voltage is applied to the process. Fig. 2 shows the input signal applied and the output signal measured for an experiment of length  $N = 6000$ . These signals will be used for the robust identification problem.

The dynamics of the resistance temperature can be modelled by

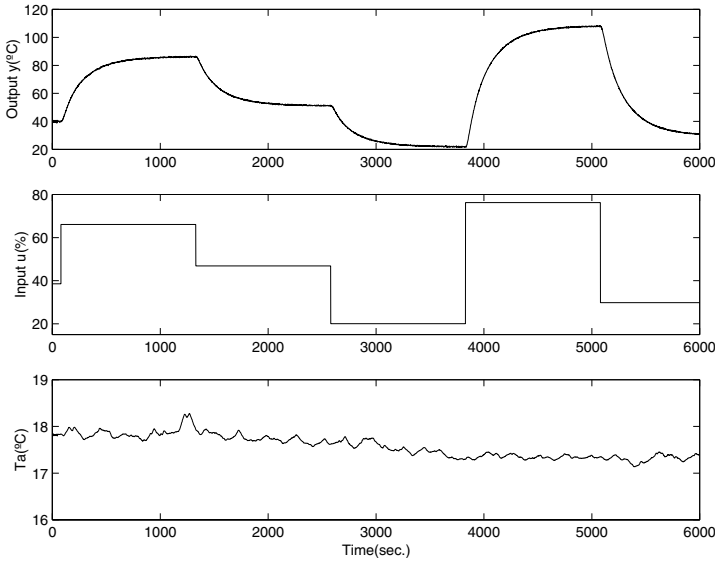
$$\begin{aligned} \dot{x}_1(t) &= \frac{1}{1000} (k_1 u(t)^2 - k_2 ((x_1(t) - T_a(t))) + O_{ffset} \\ \dot{x}_2(t) &= (1/k_3)(x_1(t) - x_2(t)) \\ \hat{y}(t) &= x_2(t) \end{aligned} \tag{6}$$

where

- $\dot{x}_1(t), \dot{x}_2(t)$  are the model states,
- $u(t)$  is the input voltage with rank 0 - 100 (%),
- $\hat{y}(t)$  is the resistance temperature ( $^{\circ}C$ ) (model output),
- $T_a(t)$  is the temperature inside furnace ( $^{\circ}C$ ),
- $k_1, k_2, k_3$  are the model parameters ( $\theta$ ) to be identified,
- $O_{ffset}$  is the correction to ensure zero steady-state error at a particular operating point.

Two norms,  $p_1 = \ell_{\infty}$  (maximum model error) and  $p_2 = \ell_1$  (average model error) have been selected. Therefore the cost function (5) is formed by using

$$J_1(\theta) = \left| \|\mathbf{E}(\theta)\|_{\infty} - \delta_1 \right|, \quad J_2(\theta) = \left| \|\mathbf{E}(\theta)\|_{\ell_1} - \delta_2 \right|. \tag{7}$$



**Fig. 2.** Output process  $y(t)$ , input process  $u(t)$  and disturbance  $T_a(t)$ .  $T = 1sec.$  (sample time),  $N = 6000$  (experiment length)

The bounds  $\delta_1$  and  $\delta_2$  are selected considering that <sup>3</sup>

$$\beta_1 = \min_{\theta \in D} \|\mathbf{E}(\theta)\|_\infty = 1.8 \text{ } ^\circ C, \quad \beta_2 = \min_{\theta \in D} \|\mathbf{E}(\theta)\|_{\ell_1} = 0.42N \text{ } ^\circ C,$$

thus,  $\delta_1 = 1.6 \times \beta_1 = 3 \text{ } ^\circ C$  and  $\delta_2 = 1.6 \times \beta_2 = 0.7N \text{ } ^\circ C$ .

The configuration parameters of the  $\epsilon - GA$  algorithm are the following:

- Searching space <sup>4</sup> $D$ :  $k_1 \in [0.05 \dots 0.12]$ ,  $k_2 \in [3.0 \dots 8.0]$  and  $k_3 \in [1.0 \dots 25.0]$ .
- $Nind_P = 8000$ ,  $Nind_{GA} = 8$ ,  $Nind_{max\_A} = 200$  and  $n_{iterations} = 3000$ .
- $\epsilon = [1.75 \cdot 10^{-3}, 0.125, 0.60]$  is the hypergrid width to have 40 divisions per dimension.

As a result, the solution  $C^*(FPS)$  obtained by  $\epsilon - GA$  is shown in Fig. 3 and compared with  $C(FPS)$  obtained by exhaustive search. The algorithm has achieved an appropriate distribution of the  $C^*(FPS)$  around  $C(FPS)$ <sup>5</sup>. The number of cost function  $J(\theta)$  evaluations results in

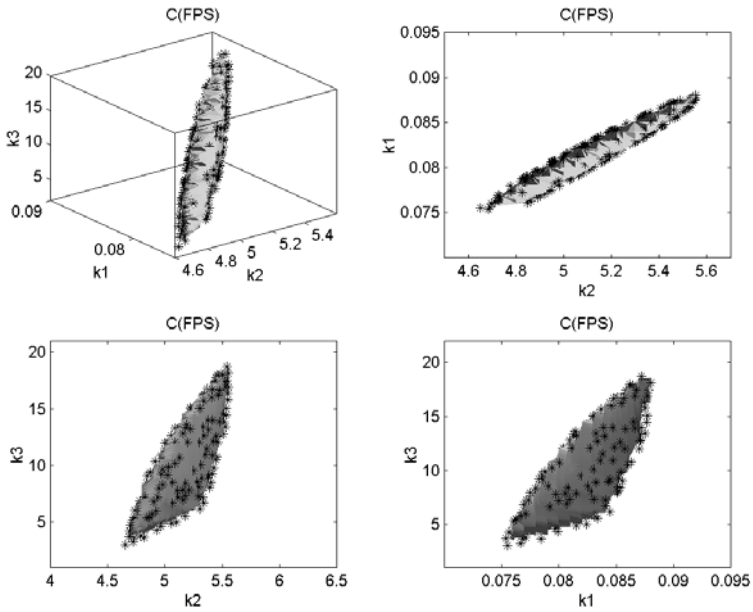
$$n_{eval\_J} = Nind\_P + n_{iteracions} \cdot Nind\_GA = 8000 + 3000 \cdot 8 = 32000,$$

<sup>3</sup> GAs has been applied to solve both optimization problems.

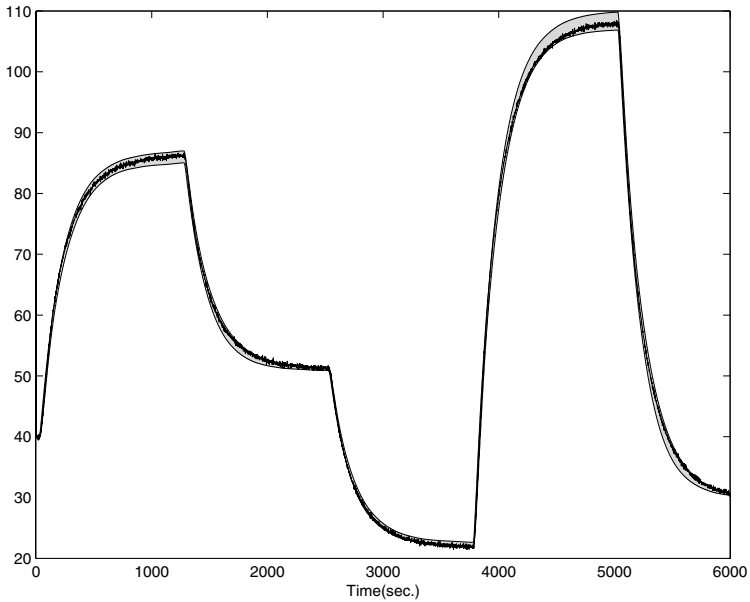
<sup>4</sup> Limits of  $D$  have been selected based on:

$\theta_\infty = arg \min_{\theta \in D} \|E(\theta)\|_\infty = [0.082, 5.15, 9.5]$ ,  $\theta_{\ell_1} = arg \min_{\theta \in D} \|E(\theta)\|_{\ell_1} = [0.079, 4.95, 5.75]$ .

<sup>5</sup> If a better characterizing of  $C(FPS)$  is required, the  $Nind_{max\_A}$  parameter should be increased.



**Fig. 3.**  $C(FPS)$  (solid surface),  $C^*(FPS)$  (stars)



**Fig. 4.** Real output  $y(t)$  (line).  $\hat{y}(t)$  simulated outputs envelope produced for all individual in  $C^*(FPS)$  (solid)

which is exactly half of the computational cost if the cost function would be evaluated in every grid box. Although the accuracy obtained with the  $\epsilon - GA$  algorithm is higher than the exhaustive search one. In addition, the mean value of the cost function for the archive individuals is 0.0031 and the maximum 0.007. Figure 4 shows that the real process output  $y(t)$  and the envelope of simulated model outputs produced by all identified parameters in  $C^*(FPS)$ .

## 5 Conclusions

A specific evolutionary algorithm  $\epsilon$ -GA, inspired by Multiobjective Evolutionary Algorithms, has been developed to find the Feasible Parameter Set ( $FPS$ ) of a nonlinear model under parametric uncertainty. That robust identification problem is formulated by assuming, simultaneously, the existence of several bounds in identification error. The solution of this problem is shown to be possible through an approximation of the contour of  $FPS$ ,  $C^*(FPS)$ . The algorithm presents the following features:

- By assuming parametric uncertainty, all kind of processes can be identified if its outputs can be calculated by model simulation. Differentiability respect to the unknown parameters is not necessary.
- Because more than one norm is taken into account at the same time, the computational cost is reduced since different  $FPS_i$  intersection is done implicitly.
- Non-convex even disjoint  $C(FPS)$  can be calculated.

## Acknowledgements

This work has been partially financed by AGL2002-04108-C02-01 and DPI2004-08383-C03-02 (MEC - FEDER).

## References

1. Coello, C., Veldhuizen, D., Lamont, G.: Evolutionary algorithms for solving multi-objective problems, Kluwer Academic Publishers, (2002)
2. Garulli, A., Reinelt, W.: Model error modeling in set membership identification. In Proc. of the System Identification Symposium, (2000)
3. Goodwin, G., Braslavsky, J., Seron, M.: Non-stationary stochastic embedding for transfer function estimation. In Proc. of the 14 th IFAC World Congress, (1999)
4. Keesman, K.J.: Membership-set estimation using random scanning and principal component analysis. Mathematics and Computers in Simulation, **32** (1990) 535-544
5. Laumanns, M., Thiele, L., Deb, K., Zitzler, E.: Combining convergence and diversity in evolutionary multi-objective optimization. Evolutionary computation, **10**(3) (2002)
6. Milanese, M., Vicino, A.: Optimal Estimation theory for Dynamic Systems with Set Membership Uncertainty: An Overview. Automatica, **27**(6) (1991) 997-1009

7. Norton, J., Veres, S.: Identification of nonlinear state-space models by deterministic search. In Proc. of the IFAC Symposium on Identification and system parameter estimation, **1** (1991) 363-368
8. Reinelt, W., Garulli, A., Ljung, L.: Comparing different approaches to model error modelling in robust identification. *Automatica*, **38**(5) (2002) 787-803
9. Walter, E., Kieffer, M.: Interval analysis for guaranteed nonlinear parameter estimation. In Proc. of the 13th IFAC Symposium on System Identification, (2003)
10. Walter, E., Piet-Lahanier, H.: Estimation of parameter bounds from bounded-error data: A survey. *Mathematics and computers in Simulation*, **32** (1990) 449-468



# Input Selection for Long-Term Prediction of Time Series

Jarkko Tikka, Jaakko Hollmén, and Amaury Lendasse

Helsinki University of Technology, Laboratory of Computer and  
Information Science, P.O. Box 5400, FI-02015 HUT, Finland

[tikka@mail.cis.hut.fi](mailto:tikka@mail.cis.hut.fi)

<http://www.cis.hut.fi/tikka>

**Abstract.** Prediction of time series is an important problem in many areas of science and engineering. Extending the horizon of predictions further to the future is the challenging and difficult task of long-term prediction. In this paper, we investigate the problem of selecting non-contiguous input variables for an autoregressive prediction model in order to improve the prediction ability. We present an algorithm in the spirit of backward selection which removes variables sequentially from the prediction models based on the significance of the individual regressors. We successfully test the algorithm with a non-linear system by selecting inputs with a linear model and finally train a non-linear predictor with the selected variables on Santa Fe laser data set.

## 1 Introduction

Time series prediction plays an important role in analysis of many problems encountered in science and engineering, for instance in climatology [1], ecology [2], electricity production [3], and economics [4].

Long-term prediction is difficult and time-consuming, since it extends the horizon of prediction further to the future, adding significant uncertainty in the prediction task. We address the problem of input selection for the purpose of long-term prediction. We present an algorithm for input variable selection in the spirit of backward selection, in which variables are progressively removed from the autoregressive prediction model. The removal of variables is based on a median and a standard deviation of parameter distributions sampled with a bootstrap resampling procedure [5]. These statistics reflect the uncertainty for a variable to play an important role in the prediction task. We apply the algorithm in a long-term prediction setting, where input selection is performed for different  $k$ -step-ahead prediction models. Whereas the input selection is accomplished with the linear autoregressive models, the final model is a non-linear predictor used in the prediction of a prominent non-linear benchmark (Santa Fe laser data set [6]).

## 2 Resampling Using Bootstrap

The bootstrap is a statistical resampling method represented by Efron *et al.* in [5]. The idea of bootstrap is to sample original data and to estimate some

statistics from these resampled sets. No assumptions are made about the forms of probability distributions in the bootstrap method. The statistic of interest and its distribution are computed by resampling the original data with replacement.

In this article the bootstrap is applied in context of the linear autoregressive process  $AR(l)$ . It can be represented by the following equation

$$y_t = \sum_{i=1}^l \alpha_i y_{t-i} + \epsilon_t . \tag{1}$$

It is assumed that  $\epsilon_t$  is a sequence of independently normally distributed random noise with zero mean and common finite variance. Let us assume that  $N + l$  observations are available. Equation (1) can be written in the matrix form as follows

$$\mathbf{y} = \mathbf{Y}\boldsymbol{\alpha} + \boldsymbol{\epsilon} , \tag{2}$$

where  $\mathbf{y} = (y_{l+1}, \dots, y_{l+N})^T$ ,  $\boldsymbol{\epsilon} = (\epsilon_{l+1}, \dots, \epsilon_{l+N})^T$ ,  $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_l)^T$  and

$$\mathbf{Y}_{(N \times l)} = \begin{bmatrix} y_l & y_{l-1} & \dots & y_1 \\ y_{l+1} & y_l & \dots & y_2 \\ \vdots & \vdots & \ddots & \vdots \\ y_{l+N-1} & y_{l+N-2} & \dots & y_N \end{bmatrix} . \tag{3}$$

The parameters  $\boldsymbol{\alpha}$  are estimated by minimizing the mean square error (MSE), i.e.,  $(1/N)\|\mathbf{y} - \hat{\mathbf{y}}\|^2 = (1/N)\|\mathbf{y} - \mathbf{Y}\hat{\boldsymbol{\alpha}}\|^2$  where  $\|\cdot\|$  is the  $L_2$ -norm. The least squares estimates of parameters are  $\hat{\boldsymbol{\alpha}} = (\mathbf{Y}^T\mathbf{Y})^{-1}\mathbf{Y}^T\mathbf{y}$ . Here we assume that the data are normalized to zero mean and unit variance and, thus there is no need for the constant term in Equations (1) and (2). Due to the normalization the upper limit for the MSE is 1, which is obtained by the mean predictor.

Bootstrapping the regression model can be done in two different ways and both could also be applied in this case. The methods are bootstrapping residuals and bootstrapping pairs [5]. The regressors i.e. the columns of the design matrix  $\mathbf{Y}$  are treated as fixed quantities in the bootstrapping residuals approach. That assumption is strong and it can fail even if Equation (2) for the  $AR(l)$  process is correct. In the bootstrapping pairs approach weaker assumptions about validity of Equation (2) are made.

In the bootstrapping pairs,  $\hat{P}$  is assumed to be an empirical distribution of the observed data pairs  $(y_{l+t}; y_{l+t-1}, y_{l+t-2}, \dots, y_t) = (y_{l+t}; \mathbf{Y}_{l+t})$ , where  $t = 1, \dots, N$  and  $y_{l+t}$  and  $\mathbf{Y}_{l+t} = [y_{l+t-1}, y_{l+t-2}, \dots, y_t]$  are values of the target and the regressors, respectively. In other words,  $y_{l+t}$  is one element of the vector  $\mathbf{y}$  and  $\mathbf{Y}_{l+t}$  is the corresponding row from the matrix  $\mathbf{Y}$ .  $\hat{P}$  puts probability mass of  $1/N$  on each pair  $(y_{l+t}; \mathbf{Y}_{l+t})$ . A bootstrap sample is a random sample of size  $N$  drawn with replacement from the population of  $N$  pairs  $(y_{l+t}; \mathbf{Y}_{l+t})$ .  $B$  independent bootstrap samples  $(\mathbf{y}^{*j}, \mathbf{Y}^{*j}), j = 1, \dots, B$ , of size  $N$  are drawn with replacement from the distribution  $\hat{P}$ . Therefore, some data pairs from the original data  $(\mathbf{y}, \mathbf{Y})$  can appear zero times or once or twice etc. in the bootstrap sample  $(\mathbf{y}^{*j}, \mathbf{Y}^{*j})$ .

The bootstrap replications  $\hat{\alpha}^{*j}$  of the estimates of parameters  $\hat{\alpha}$  are  $\hat{\alpha}^{*j} = (\mathbf{Y}^{*jT} \mathbf{Y}^{*j})^{-1} \mathbf{Y}^{*jT} \mathbf{y}^{*j}$ .  $B$  bootstrap replications  $\hat{\alpha}_i^{*j}$  form a distribution of the estimates of parameters  $\hat{\alpha}_i$ . This distribution of the bootstrap replications can be used in an evaluation of the significance of parameters. In this paper, the evaluation is based on medians and standard deviations of the distributions.

### 3 Input Selection

The usual task is to define the order  $l$  in Equation (1) and use all the inputs  $y_{t-i}, i = 1, \dots, l$  in the prediction of  $y_t$ . This kind of solution is referred to ordinary least squares (OLS) solution later on. The OLS solution is not always satisfactory. The OLS estimates have low bias but a large variance, which may diminish the prediction accuracy. The prediction accuracy can sometimes be improved by shrinking some estimates of parameters toward zero or setting them exactly to zero [7].

In this paper, we present a methodology to select a subset of inputs that have the strongest explanatory power in the AR process. This also makes the interpretation of the dependencies in time series easier.

First, the maximum number of inputs have to be set, i.e., to define the value of  $l$ . It can be selected to be a relatively large to ensure that the final order of the estimated AR process can be large enough. The procedure continues by estimating AR process using all the inputs  $y_{t-i}, i = 1, \dots, l$  that is calculating the OLS estimates of parameters  $\hat{\alpha}_i$ . The distributions of parameters  $\hat{\alpha}_i$  and the standard deviation  $s$  of the training MSE are estimated using the bootstrapping pairs method. In addition, the validation error is calculated using the parameters  $\hat{\alpha}_i$ . The validation error is the MSE for the validation set, i.e., for data that are not used in the training phase.

The next step is to delete the least significant regressor. The median value  $m_i$  and the standard deviation  $\sigma_i$  of parameter  $\hat{\alpha}_i$  are calculated from the bootstrap replications  $\hat{\alpha}_i^{*j}$ . The ratio  $|m_i|/\sigma_i$  is used as a measure of significance of the corresponding parameter. The parameter or regressor with the smallest ratio is pruned from the set of possible inputs. After that, the bootstrapping using the remaining inputs and pruning is repeated as long as there are variables left in the set of inputs.

The quantity  $d_i = (\sum_{j=1}^B (m_i - \hat{\alpha}_i^{*j})^2)^{\frac{1}{2}} / (B - 1)$  could be used instead of the standard deviation as a measure of the width of the distribution of the estimates of parameters. Another alternative is to estimate the width of the distribution using the difference  $\Delta_i = \hat{\alpha}_{i,high}^{*j} - \hat{\alpha}_{i,low}^{*j}$ , where  $\hat{\alpha}_{i,high}^{*j}$  is  $B \cdot (1 - q)$ th and  $\hat{\alpha}_{i,low}^{*j}$  is  $B \cdot q$ th value in the ordered list of the  $B$  replications of  $\hat{\alpha}_i^{*j}$  and  $q$  could be for instance 0.165.

The Algorithm 1 produces sequentially  $l$  different models. The computational complexity of the proposed algorithm is  $\mathcal{O}(l)$  whereas a backward selection type of algorithm would be of complexity  $\mathcal{O}(l^2)$  and the full, exhaustive (brute-force) search of all possible variable configurations would take  $\mathcal{O}(2^l)$ . The purpose is to select a model which is as sparse as possible in the sense that it includes only a

**Algorithm 1** Algorithm for input selection

- 
- 1: Let  $\mathcal{L}$  be the set of indices of the inputs. In the beginning,  $\mathcal{L}$  includes all the indices  $i, y_{t-i}, i = 1, \dots, l$ .
  - 2: if  $i \in \mathcal{L}$  estimate  $B$  bootstrap replications  $\hat{\alpha}_i^{j*}, j = 1, \dots, B$  of the parameters  $\hat{\alpha}_i$  by minimizing MSE using the training data, otherwise set  $\hat{\alpha}_i = 0, i \notin \mathcal{L}$
  - 3: Compute the mean and the standard deviation  $s_{\mathcal{L}}$  of the bootstrap replications of the training MSEs.
  - 4: Estimate parameters  $\hat{\alpha}_i$  using the training data and compute the validation error  $E_v$  using the estimated model.
  - 5: Compute the median  $m_i$  and the standard deviation  $\sigma_i$  for all the estimates  $\hat{\alpha}_i, i \in \mathcal{L}$  from the bootstrap replications  $\hat{\alpha}_i^{j*}$ .
  - 6: Compute the ratio  $|m_i|/\sigma_i, \forall i \in \mathcal{L}$  and delete the index with the smallest ratio from the set  $\mathcal{L}$ .
  - 7: if  $\mathcal{L} \neq \emptyset$  go to step 2, otherwise go to step 8
  - 8: Find the minimum validation error  $E_v^{min}$ . Sum  $E_v^{min}$  and corresponding  $s_{\mathcal{L}}^{min}$ . Find the smallest set of indices  $\mathcal{L}_f$  i.e. the least complex model whose validation error is under the previous sum. The inputs corresponding to the indices in the set  $\mathcal{L}_f$  are included to the final model.
- 

few input variables. Goals of this approach are to avoid the curse of dimensionality, over-parameterization, and overfitting for the non-linear modeling phase. On the other hand, the prediction accuracy should not decrease significantly. The initial model is selected based on the minimum validation error  $E_v^{min}$ . The corresponding standard deviation  $s^{min}$  of training error is also used as the estimate of standard deviation of  $E_v^{min}$ . The final model is the least complex model whose validation error is under the sum  $E_v^{min} + s^{min}$ . The procedure is listed step by step in detail in Algorithm 1.

The ratio  $|m_i|/\sigma_i$  can be seen as a  $Z$ -score test statistic to test the hypothesis [8] that a parameter  $\hat{\alpha}_i = 0$ . The statistics are estimated using bootstrap, so there are not made any assumptions about the probability distributions of parameters. This applies especially when the median and the difference  $\Delta_i$  would be used. We use the median instead of the mean as the location parameter of the distribution, since it is a reasonable estimate for skewed distributions, and distributions with outliers. Furthermore, median is the same as the mean, if the distribution is symmetric. The standard deviation and deviation based on the median are reasonable measures for the symmetric distributions, whereas the difference  $\Delta_i$  describes better the width of an asymmetric distribution. In the case of a symmetric distribution, for instance a Gaussian distribution,  $\sigma_i, d_i,$  and  $\Delta_i$  produce equal estimates of the width of the distributions.

In addition, the ratio can be considered as a signal-to-noise ratio. If the ratio is small then the bootstrap replications of parameters vary strongly around the median and the importance of the corresponding input in the prediction is insignificant. The idea of proposed algorithm is similar to the traditional backward stepwise selection, which sequentially deletes inputs based on the  $F$ -statistics [8].

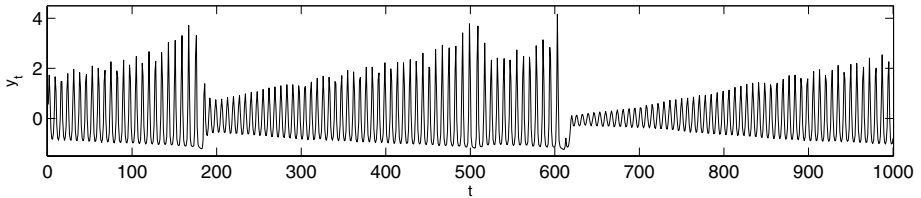
Algorithm 1 can also be applied to long-term prediction. In that case the target is  $y_{t+k}, k > 0$  in the Equation (1).

## 4 Long-Term Prediction

In the literature, two techniques are used for long-term prediction [6]. In recurrent prediction — the first of the two approaches — only one model is built, that is,  $\hat{y}_t = f(y_{t-1}, \dots, y_{t-l})$ . The same model is used repeatedly to obtain the long-term prediction. For instance, for the two-step-prediction the model  $\hat{y}_{t+1} = f(\hat{y}_t, y_{t-1}, \dots, y_{t-l+1})$  is used [9]. The main advantage of this method is that only one model has to be built, the disadvantage being that the quality of predictions is rapidly decreasing with respect to the increasing horizon of prediction. Therefore, in the context of long-term prediction, we focus on the models of the form  $\hat{y}_{t+k} = f_k(y_{t-1}, \dots, y_{t-l})$ , which is known as the direct predictor. Because the computational load of this method, i.e., building different models up to the maximum prediction horizon  $K$ , we need a fast method for input selection. Moreover, we want to select inputs in a non-contiguous manner, for instance, allowing for the inputs such as  $\hat{y}_{t+k} = f_k(y_{t-1}, y_{t-3}, y_{t-8}, y_{t-12})$ .

## 5 Experiments on the Santa Fe Laser Data Set

The data set used in the experiments is the Santa Fe laser data set described in [6]. The training set size  $N_{tr} = 1000$  is selected to be the same as in the Santa Fe time-series competition [6] (see Figure 1). The validation set used for the selection of the model-orders is chosen to be large ( $N_{val} = 4546$ ) in order to perform well on structure selection, and a large testing set ( $N_{test} = 4547$ ) should ensure realistic statistical estimate of the performances of the results.

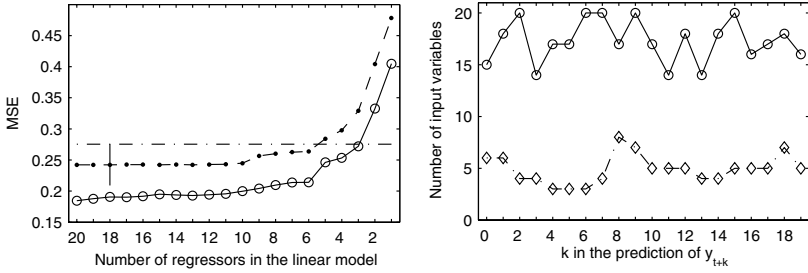


**Fig. 1.** The first thousand samples  $N_{tr}$  from Santa Fe laser time series

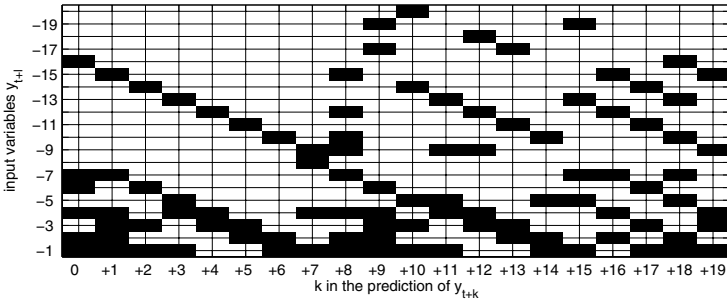
### 5.1 Linear Models and Input Selection

Input selection Algorithm 1 has been used to yield input variables with a maximum number of inputs  $l = 20$  and the maximum prediction horizon being  $K = 19$  corresponding to 20 models. The number of bootstrap resamples has been chosen to be  $B = 1000$ , which is considered to be large enough for reliably estimating the distribution of the parameters in the linear model.

Figure 2 (a) presents an example of input selection in the case of two-step-ahead prediction ( $k = +1$ ). In this case, Algorithm 1 selected the model with 6 inputs. Compared to the model with 18 inputs with the minimum validation



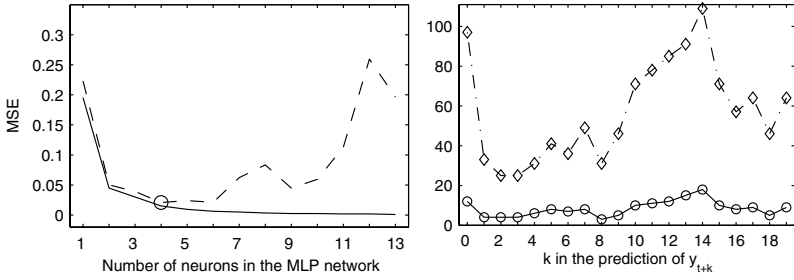
**Fig. 2.** (left) An example of input selection, training error (solid line), validation error (dashed line), vertical line marks the minimum validation error (18 inputs) and a standard deviation of training error, the final model is chosen to be the least complex model (6 inputs) within the standard error (dash-dotted line). (right) number of selected inputs for each 20 models according to the minimum validation error (solid line) and the standard error (dash-dotted line)



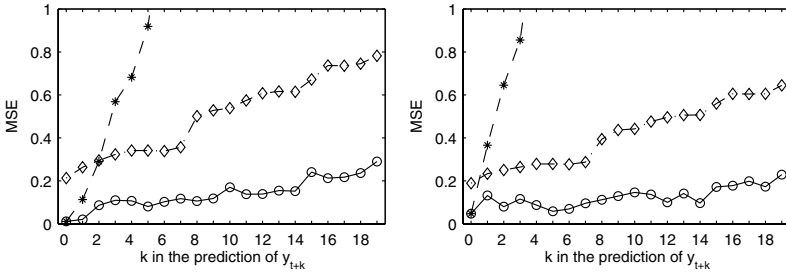
**Fig. 3.** The final models. The targets  $y_{t+k}, k = 0, +1, \dots, +19$  are in the horizontal axis and the possible regressors  $y_{t+l}, l = -1, -2, \dots, -20$  are in the vertical axis. The selected inputs are denoted by black rectangles in each column

error the number of parameters has decreased considerably (by 12). At the same time the validation error has only increased less than one standard deviation of the minimum validation error. In Figure 2 (b), the number of inputs for each 20 models according to the minimum validation error and the final number of selected inputs are shown. The reduction in the number of parameters is on average 12. In Figure 3, the selected inputs for all long-term models are visualized. Parsimonious models are selected based on the proposed algorithm, on average 5 inputs are selected. For instance, the 12-step-ahead model has 5 inputs, marked by 5 black rectangles on the vertical axis at column  $k = +11$  ( $\hat{y}_{t+11} = f(y_{t-1}, y_{t-4}, y_{t-5}, y_{t-9}, y_{t-13})$ ).

The final models were obtained using the median  $m_i$  and the standard deviation  $\sigma_i$  in the input selection. The final models were exactly same as in Figure 3, when the input selection was repeated using  $d_i$  and  $\Delta_i$  instead of  $\sigma_i$ . This might indicate that the distributions are approximately normal distributions.



**Fig. 4.** (left) An example of the selection of number of neurons  $p$  (the target is  $y_{t+1}$ ), training error (solid line), validation error (dashed line), circle represents minimum validation error. (right) the number of neurons  $p$  (solid line) and the number of parameters (dash-dotted line) in each 20 MLP network



**Fig. 5.** MSE for MLP networks (solid line), linear models (dash-dotted line) and recurrent prediction (dashed line) for both validation (left panel) and test sets (right panel), respectively

### 5.2 Non-linear, Non-contiguous Regressor Model

Based on the results from the previous section, we train a non-linear model. Here, we have used a multi-layer perceptron (MLP) network [10]. The output of the network is  $\hat{y}_t = \mu + \sum_{j=1}^p \beta_j \tanh \left( \sum_{i=1}^l w_{ji} y_{t-i} + b_j \right)$ , where  $p$  is the number of neurons in the hidden layer. The network is trained using the Levenberg-Marquardt optimization method by back-propagating the error gradients [10]. Hundred different initializations are used in the training of the models in order to avoid local minima of the training error, and the best model is chosen (for each model-order). The minimum training error (among different initializations) is a monotonically decreasing function with respect to  $p$ . Validation error is evaluated for the best non-linear model to choose the final number of neurons  $p$ . The maximum number of tested neurons was 20. An example is shown in Figure 4 (a). The selected number of neurons  $p$ , which is less than 20 in each network, and the corresponding number of parameters in each 20 MLP networks are shown in Figure 4 (b). The maximum number of parameters in the non-linear models is 109. Because of the parsimony of the selected input set, it is possible to train the model from a small training sample such as ours ( $N_{tr} = 1000$ ).

Figure 5 illustrates the differences between the recurrent and direct prediction for the validation (left panel) and the test (right panel) set up to the maximum prediction horizon  $K$ . The direct non-linear predictions (solid line) clearly outperform the recurrent approach (dashed line). The direct predictions calculated using the linear models (dash-dotted line) are also better than the recurrent non-linear predictions in the long-term.

## 6 Conclusions

The proposed algorithm selected parsimonious sets of inputs for all long-term prediction models. The non-linear models built using the corresponding inputs led to good prediction performance. Direct long-term prediction is superior to recurrent prediction for the whole prediction horizon. The main advantage of the proposed approach is that it combines fast input selection with accurate but computationally demanding non-linear prediction. Linear complexity of the input selection makes this approach viable for input selection in large-scale problems. Further research include experiments with different data sets and comparisons to different input selection methodologies, e.g. the use of mutual information, to achieve accurate comprehension of the performance of the proposed algorithm.

## References

1. Dal Cin, C., Moens, L., Dierickx, P., Bastin, G., Zech, Y.: An Integrated Approach for Real-time Flood-map Forecasting on the Belgian Meuse River. (Natural Hazards), In press.
2. Sulkava, M., Tikka, J., Hollmén, J.: Sparse Regression for Analyzing the Development of Foliar Nutrient Concentrations in Coniferous Trees. In: Proceedings of the Fourth International Workshop on Environmental Applications of Machine Learning (EAML 2004). (2004) 57-58
3. Lendasse, A., Lee, J., Wertz, V., Verleysen, M.: Forecasting Electricity Consumption Using Nonlinear Projection and Self-organizing Maps. *Neurocomputing* **48** (2002) 299-311
4. Hamilton, J.D.: Analysis of Time Series Subject to Changes in Regime. *Journal of Econometrics* **45** (1990) 39-70
5. Efron, B., Tibshirani, R.J.: An Introduction to the Bootstrap. Chapman & Hall/CRC (1993)
6. Weigend, A.S., Gershenfeld, N.A. (eds.): Time Series Prediction: Forecasting the Future and Understanding the Past. Addison-Wesley (1994)
7. Copas, J.: Regression, Prediction and Shrinkage. *Journal of the Royal Statistical Society (Series B)* **45** (1983) 311-354
8. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning – Data Mining, Inference and Prediction. Springer Series in Statistics. Springer (2001)
9. Tresp, V., Hofmann, R.: Nonlinear Time-series Prediction with Missing and Noisy Data. *Neural Computation* **10** (1998) 731-747
10. Bishop, C.: Neural Networks in Pattern Recognition. Oxford Press (1996)



# Direct and Recursive Prediction of Time Series Using Mutual Information Selection

Yongnan Ji, Jin Hao, Nima Reyhani, and Amaury Lendasse

Neural Network Research Centre,  
Helsinki University of Technology, P.O. Box 5400,  
02150 Espoo, Finland  
{yji, jhao, nreyhani, lendasse}@cis.hut.fi

**Abstract.** This paper presents a comparison between direct and recursive prediction strategies. In order to perform the input selection, an approach based on mutual information is used. The mutual information is computed between all the possible input sets and the outputs. Least Squares Support Vector Machines are used as non-linear models to avoid local minima problems. Results are illustrated on the Poland electricity load benchmark and they show the superiority of the direct prediction strategy.

**Keywords:** Time Series Prediction, Mutual Information, Direct Prediction, Recursive Prediction, Least Squares Support Vector Machines and Prediction Strategy.

## 1 Introduction

Prediction is an important part of decision making and planning process in engineering, business, medicine and many other application domains. Long-term prediction is typically faced with growing uncertainties arising from various sources, for instance, accumulation of errors and lack of information [1]. In long-term prediction, when predicting multiple steps ahead, we have several choices. In this work, two variants of prediction approaches, namely, direct and recursive prediction, using Least Squares Support Vector Machines (LS-SVM) [17], are studied and compared. Meanwhile, to improve the efficiency of prediction, mutual information (MI) is used to select the inputs [12]. Based on the experimental results, a combination of input selection and forecast strategy which can give comparatively accurate long-term time series prediction will be presented.

The paper is organized as follows: in section 2, mutual information is introduced. Time series prediction is explained in section 3. In section 4, LS-SVM is defined. In section 5 we present the experimental results and in section 6 conclusions and further works are presented.

## 2 Mutual Information for Input Selection

### 2.1 Input Selection

Input selection is one of the most important issues in machines learning especially when the number of observations is relatively small compared to the number of inputs. In practice, there is no dataset with infinite number of data points and furthermore, the necessary size of the dataset increases dramatically with the number of observations (curse of dimensionality). To circumvent this, one should first select the best inputs or regressors in the sense that they contain the necessary information. Then it would be possible to capture and reconstruct the underlying relationship between input-output data pairs. Within this respect, some model dependent approaches have been proposed [2-6].

Some of them deal with the problem of feature selection as a generalization error estimation problem. In this methodology, the set of inputs that minimize the generalization error is selected using Leave-one-out, Bootstrap or other resampling techniques. These approaches are very time consuming and may take several weeks. However, there are model independent approaches [7-11] which select a priori inputs based only on the dataset, as presented in this paper. So the computational load would be less than in model dependent cases. Model independent approaches select a set of inputs by optimizing a criterion over different combinations of inputs. The criterion computes the dependencies between each combination of inputs and the corresponding output using predictability, correlation, mutual information or other statistics.

In this paper, the mutual information is used as a criterion to select the best input variables (from a set of possible variables) for long-term prediction purpose.

### 2.2 Mutual Information

The mutual information (MI) between two variables, let say  $X$  and  $Y$ , is the amount of information obtained from  $X$  in presence of  $Y$ , and vice versa. MI can be used for evaluating the dependencies between random variables, and has been applied for Feature Selection and Blind Source Separation [12].

Let us consider two random variables: the MI between them would be

$$I(X, Y) = H(X) + H(Y) - H(X, Y) , \quad (1)$$

where  $H(\cdot)$  computes the Shannon's entropy. In the continuous entropy case, equation (1) leads to complicated integrations, so some approaches have been proposed to evaluate them numerically. In this paper, a recent estimator based on  $k$ -Nearest Neighbors statistics is used [13]. The novelty of this approach consists in its ability to estimate the MI between two variables of any dimensional spaces. The basic idea is to estimate  $H(\cdot)$  from the average distance to the  $k$ -Nearest Neighbors (over all  $x_i$ ). MI is derived from equation (1) and is estimated as

$$I(X, Y) = \psi(k) - 1/k - \langle \psi(n_x) + \psi(n_y) \rangle + \psi(N) \quad (2)$$

with  $N$  the size of dataset and  $\psi(x)$  the digamma function,

$$\psi(x) = \Gamma(x) - 1 - d\Gamma(x)/dx \quad (3)$$

$\psi(1) \approx -0.5772156$  and

$$\langle \dots \rangle = N^{-1} \sum_{i=1}^N E[\dots(i)] \quad (4)$$

$n_x(i)$ ,  $n_y(i)$  are the numbers of points in the region  $\|x_i - x_j\| \leq \epsilon_x(i)/2$  and  $\|y_i - y_j\| \leq \epsilon_y(i)/2$ .  $\epsilon(i)/2$  is the distance from  $z_i$  to its  $k$ -Nearest Neighbors.  $\epsilon_x(i)/2$  and  $\epsilon_y(i)/2$  are the projections of  $\epsilon(i)/2$  [14].  $k$  is set to be 6, as suggested in [14]. Software for calculating the MI based on this method can be downloaded from [15].

### 3 Time Series Prediction

Basically, time series prediction can be considered as a modeling problem [16]: a model is built between the input and the output. Then, it is used to predict the future values based on the previous values. In this paper we use two different strategies to perform the long-term prediction, which are direct and recursive forecasts.

#### 3.1 Direct Forecast

In order to predict the values of a time series,  $M + 1$  different models are built,

$$\hat{y}(t + m) = f_m(y(t - 1), y(t - 2), \dots, y(t - n)) \quad (5)$$

with  $m = 0, 1, \dots, M$ ,  $M$  is the maximum horizon of prediction. The input variables on the right-hand part of (5) form the regressor, where  $n$  is the regressor size.

#### 3.2 Recursive Forecast

Alternatively, model can be constructed by first making one step ahead prediction,

$$\hat{y}(t) = f(y(t - 1), y(t - 2), \dots, y(t - n)) \quad (6)$$

and then predict the next value using the same model,

$$\hat{y}(t + 1) = f(\hat{y}(t), y(t - 1), y(t - 2), \dots, y(t - n + 1)) \quad (7)$$

In equation (7), the predicted value of  $y(t)$  is used instead of the value itself, which is unknown. Then,  $\hat{y}(t + 1)$  to  $\hat{y}(t + M)$  are predicted recursively.

### 4 Least Squares Support Vector Machines

LS-SVM are regularized supervised approximators. Comparing with simple SVM, Only linear equation is needed to solve the results, which avoids the local minima in SVM. A short summary of the LS-SVM is given here; more details are given in [17].

The LS-SVM model [18-20] is defined in its primal weight space by,

$$\hat{y} = \omega^T \phi(\mathbf{x}) + b \quad (8)$$

where  $\phi(x)$  is a function which maps the input space into a higher dimensional feature space,  $\mathbf{x}$  is the  $N$ -dimensional vector of inputs  $x_i$ , and  $\omega$  and  $b$  the parameters of the model. In Least Squares Support Vector Machines for function estimation, the following optimization problem is formulated,

$$\min_{\omega, b, e} J(\omega, e) = \frac{1}{2} \omega^T \omega + \gamma \frac{1}{2} \sum_{i=1}^N e_i^2 \quad (9)$$

subject to the equality constraints,

$$y^i = \omega^T \phi(\mathbf{x}^i) + b + e^i, i = 1, \dots, N \quad (10)$$

In equation (10), the superscript  $i$  refers to the number of a sample. Solving this optimization problem in dual space leads to finding the  $\alpha_i$  and  $b$  coefficients in the following solution,

$$h(\mathbf{x}) = \sum_{i=1}^N \alpha_i K(\mathbf{x}, \mathbf{x}_i) + b \quad (11)$$

Function  $K(\mathbf{x}, \mathbf{x}_i)$  is the kernel defined as the dot product between the  $\phi(\mathbf{x})^T$  and  $\phi(\mathbf{x})$  mappings. The meta-parameters of the LS-SVM model are  $\sigma$ , width of the Gaussian kernels (taken to be identical for all kernels), and  $\gamma$ , regularization factor. LS-SVM can be viewed as a form of parametric ridge regression in the primal space. Training methods for the estimation of the  $\omega$  and  $b$  parameters can be found in [17].

### 5 Experimental Results

The dataset used in this experiment is a benchmark in the field of time series prediction: the Poland Electricity Dataset [21]. It represents the daily electricity load of Poland during 2500 days in the 90s.

The first two thirds of the whole dataset is used for training, and the remaining data for testing. To apply the prediction model in equation (5), we set the maximum time horizon  $M = 6$  and the regressor size  $n = 8$ .

First, MI presented in section 2.2 is used to select the best input variables. All the  $2^n - 1$  combinations of inputs are tested. Then, the one that gives the maximum MI is selected. The selection results for direct forecast are:

**Table 1.** Input selection results of MI

|          | $y(t)$ | $y(t+1)$ | $y(t+2)$ | $y(t+3)$ | $y(t+4)$ | $y(t+5)$ | $y(t+6)$ |
|----------|--------|----------|----------|----------|----------|----------|----------|
| $y(t-1)$ | X      | X        | X        | X        | X        | X        | X        |
| $y(t-2)$ | X      | X        | X        | X        | X        | X        | X        |
| $y(t-3)$ |        |          | X        |          | X        | X        |          |
| $y(t-4)$ |        | X        | X        | X        |          |          |          |
| $y(t-5)$ |        | X        | X        |          |          |          |          |
| $y(t-6)$ | X      |          |          | X        |          |          |          |
| $y(t-7)$ |        |          |          | X        |          |          |          |
| $y(t-8)$ |        |          |          |          |          |          | X        |

For example, the 4<sup>th</sup> column means that,

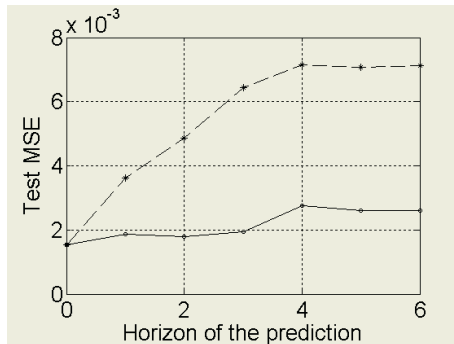
$$\hat{y}(t+3) = f_3(y(t-1), y(t-2), y(t-4), y(t-6), y(t-7)) . \tag{12}$$

Then the LS-SVM is used to make the prediction. To select the optimal parameters model selection method should be used here, in the experiment, leave-one-out is uses. The errors for the leave-one-out procedure of every pairs of  $\gamma$  and  $\sigma$  are listed. Then the area around the minima is zoomed and searched until the hyper parameters are found. For recursive prediction, only one function is used, so one pair of  $\gamma$  and  $\sigma$  is needed, which is (33, 0.1). For direct prediction, seven pairs of parameters are required. They are (33, 0.1), (40, 0.1), (27, 0.1), (27, 0.1), (27, 0.1), (22, 0.1) and (27, 0.1). The mean square error values of the results are listed in the table below:

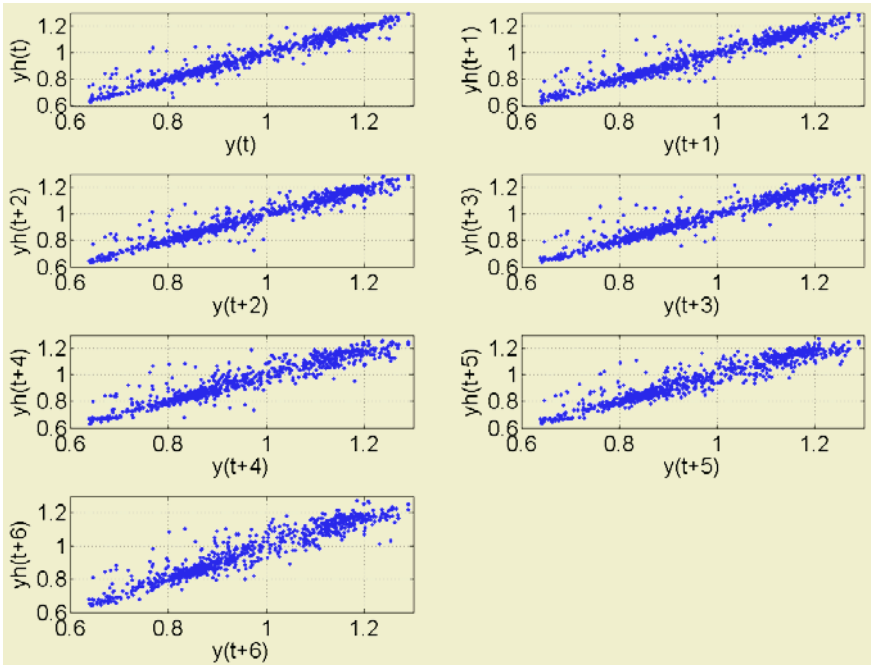
**Table 2.** MSE values of direct and recursive prediction

|           | $y(t)$  | $y(t+1)$ | $y(t+2)$ | $y(t+3)$ | $y(t+4)$ | $y(t+5)$ | $y(t+6)$ |
|-----------|---------|----------|----------|----------|----------|----------|----------|
| direct    | 0,00154 | 0,00186  | 0,00178  | 0,00195  | 0,00276  | 0,00260  | 0,00260  |
| recursive | 0,00154 | 0,00362  | 0,00486  | 0,00644  | 0,00715  | 0,00708  | 0,00713  |

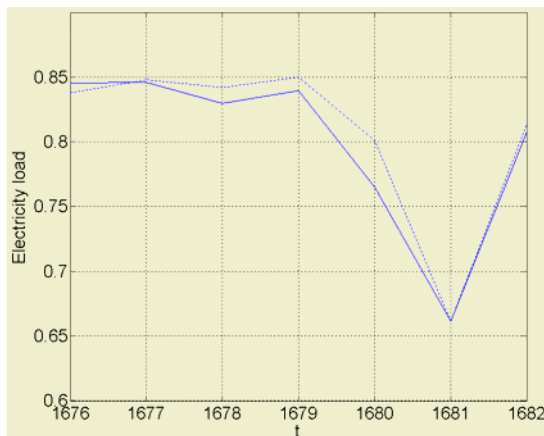
As illustration, the MSE values are presented also in Fig. 1:



**Fig. 1.** Prediction results comparison: dashed line corresponds to recursive prediction and solid line corresponds to direct prediction



**Fig. 2.**  $\hat{y}(t)$  (represented as  $y_h$ ) and  $y(t)$  for each horizon of prediction



**Fig. 3.** An example of prediction:  $\hat{y}(t)$  is represented in dotted line and  $y(t)$  is represented in solid line

In Fig. 1, the horizontal axis represents  $i$  in  $y(t+i)$ , which varies from 0 to 6. The vertical axis represents the corresponding MSE values. The dashed line shows MSE values for recursive prediction and the solid line shows MSE values for direct predic-

tion. From this figure, it can be seen that as  $i$  increases, the performances of the direct predictions are better than that of the recursive ones.

To illustrate the prediction results, the predicted values by direct prediction are plotted against the real data in Fig. 2. The more the points are concentrated around a line, the better the predictions are. It can be seen that when  $i$  is large, the distribution of the points diverts from a line, because the prediction becomes more difficult.

In Fig. 3, one example of the prediction results is given. The dashed line represents seven real values from the Poland dataset. The solid line is the estimation using direct prediction. The figure shows that the predicted values and the real values are very close.

The same methodology has been applied to other benchmark and similar results have been obtained.

## 6 Conclusion

In this paper, we compared two long-term prediction strategies: direct forecast and recursive forecast. MI is used to perform the input selection for both strategies: MI works as a criterion to estimate the dependencies between each combination of inputs and the corresponding output. Though  $2^n - 1$  combinations must be calculated, it is fast compared to other input selection methods. The results show that this MI based method can provide a good input selection.

Comparing both long-term prediction strategies, direct prediction gives better performances than recursive prediction. The former strategy requires multiple models. Nevertheless, due to the simplicity of the MI input selection method, direct prediction strategy can be used in practice. Thus, the combination of direct prediction and MI input selection can be considered as an efficient approach for a long-term time series prediction.

## Acknowledgements

Part of work of Y. Ji, J. Hao, N. Reyhani and A. Lendasse is supported by the project of New Information Processing Principles, 44886, of the Academy of Finland.

## References

1. Weigend A.S., Gershenfeld N.A.: Times Series Prediction: Forecasting the future and Understanding the Past. Addison-Wesley, Reading MA (1994).
2. Kwak, N., Chong-Ho, Ch.: Input feature selection for classification problems. Neural Networks, IEEE Transactions, Vol. 13, Issue 1 (2002) 143–159.
3. Zongker, D., Jain, A.: Algorithms for feature selection: An evaluation Pattern Recognition. Proceedings of the 13th International Conference, Vol. 2, 25-29 (1996) 18-22.
4. Ng., A Y.: On feature selection: learning with exponentially many irrelevant features as training examples. In Proc. 15th Intl. Conf. on Machines Learning (1998) 404-412.

5. Xing, E.P., Jordan, M.I., Karp, R.M.: Feature Selection for High-Dimensional Genomic Microarray Data. Proc. of the Eighteenth International Conference in Machine Learning, ICML2001 (2001).
6. Law, M., Figueiredo, M., Jain, A.: Feature Saliency in Unsupervised Learning. Tech. Rep., Computer Science and Eng., Michigan State Univ (2002).
7. Efron, B., Tibshirani, R., J.: Improvements on cross-validation: The .632+ bootstrap method. Amer., Statist. Assoc. 92 (1997) 548–560.
8. Stone, M., J.: An asymptotic equivalence of choice of model by cross-validation and Akaike's criterion. Royal, Statist. Soc. B39 (1977) 44–7.
9. Kohavi, R.: A study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. Proc. of the 14th Int. Joint Conf. on A.I., Vol. 2, Canada (1995).
10. Efron, B.: Estimating the error rate of a prediction rule: improvements on cross-validation. Journal of American Statistical Association, Vol. 78 Issue. 382 (1993) 316–331.
11. Jones, A., J.: New Tools in Non-linear Modeling and Prediction. Computational Management Science, Vol. 1, Issue 2 (2004) 109-149.
12. Yang, H., H., Amari, S.: Adaptive online learning algorithms for blind separation: Maximum entropy and minimum mutual information, Neural Comput., vol. 9 (1997) 1457-1482.
13. A. Kraskov, H. Stögbauer, and P. Grassberger, Phys. Rev. E, in press <http://arxiv.org/abs/cond-mat/0305641>
14. Harald, S., Alexander, K., Sergey, A., A., Peter, G.: Least Dependent Component Analysis Based on Mutual Information. Phys. Rev. E 70, 066123 September 28 (2004).
15. URL: [http://www.fz-juelich.de/nic/Forschungsgruppen/Komplexe\\_Systeme/software/milca-home.html](http://www.fz-juelich.de/nic/Forschungsgruppen/Komplexe_Systeme/software/milca-home.html).
16. Xiaoyu, L., Bing, W., K., Simon, Y., F.: Time Series Prediction Based on Fuzzy Principles. Department of Electrical & Computer Engineering, FAMU-FSU College of Engineering, Florida State University, Tallahassee, FL 32310.
17. Suykens, J., A., Van Gestel, K., T., De Brabanter, J., De Moor, B., Vandewalle, J.: Least Squares Support Vector Machines. World Scientific, Singapore, ISBN 981-238-151-1 (2002).
18. Suykens, J., A., De brabanter, K., J., Lukas, L., Vandewalle, J.: Weighted least squares support vector machines: robustness and sparse approximation. Neurocomputing, Volume 48, Issues 1-4, October 2002, Pages 85-105.
19. Suykens, J., A., K., Lukas, L., Vandewalle, J.: Sparse Least Squares Support Vector Machine Classifiers, in: Proc. of the European Symposium on Artificial Neural Networks ESANN'2000 Bruges (2000) 37-42.
20. Suykens, J., A., K., Vandewalle, J.: Training multilayer perceptron classifiers based on modified support vector method. IEEE Transactions on Neural Networks, vol. 10, no. 4 Jul. (1999) 907-911.
21. Cottrell, M., Girard, B., Rousset, P.: Forecasting of curves using a Kohonen classification. Journal of Forecasting, 17: (5-6) (SEP-NOV) (1998) 429-439.



# Load Forecasting Using Fixed-Size Least Squares Support Vector Machines

Marcelo Espinoza, Johan A.K. Suykens, and Bart De Moor

K.U. Leuven, ESAT-SCD-SISTA,  
Kasteelpark Arenberg 10, B-3001 Leuven (Heverlee), Belgium  
{marcelo.espinoza, johan.suykens}@esat.kuleuven.ac.be

**Abstract.** Based on the Nyström approximation and the primal-dual formulation of Least Squares Support Vector Machines (LS-SVM), it becomes possible to apply a nonlinear model to a large scale regression problem. This is done by using a sparse approximation of the nonlinear mapping induced by the kernel matrix, with an active selection of support vectors based on quadratic Renyi entropy criteria. The methodology is applied to the case of load forecasting as an example of a real-life large scale problem in industry, for the case of 24-hours ahead predictions. The results are reported for different number of initial support vectors, which cover between 1% and 4% of the entire sample, with satisfactory results.

## 1 Introduction

Computing forecasts of a time series requires the estimation of a good initial model. The more information is included in the model, the better the performance of the forecasts, leading to good performance beyond the first step out of sample. Kernel based estimation techniques, such as Support Vector Machines (SVMs) and Least Squares Support Vector Machines (LS-SVMs) have shown to be powerful nonlinear regression methods [16, 1, 17]. Both techniques build a linear model in the so-called feature space where the inputs have been transformed by means of a (possibly infinite dimensional) nonlinear mapping  $\varphi$ . This is converted to the dual space by means of the Mercer's theorem and the use of a positive definite kernel, without computing explicitly the mapping  $\varphi$ . The SVM model solves a quadratic programming problem in dual space, obtaining a sparse solution. The LS-SVM formulation, on the other hand, solves a linear system in dual space under a least-squares cost function [15], where the sparseness property can be obtained by sequentially pruning the support value spectrum.

Although the LS-SVM system is solved on its dual form, the problem can be formulated directly in primal space by means of an explicit approximation for the nonlinear mapping  $\varphi$ . Furthermore, it is possible to compute a sparse approximation by using only a subsample of selected Support Vectors from the dataset in order to estimate a large-scale nonlinear regression problem in primal space, making use of all possible information from the available data sample. Working in primal space gives enough flexibility to apply different techniques from statistics or the traditional system identification framework [10].

In this paper we apply LS-SVM in primal space to the problem of short-term load forecasting, which is an important area of quantitative research [13, 4], in order to build a model that can be used for 24-hours ahead prediction. Usually a load series shows important seasonal patterns (yearly, weekly, intra-daily patterns) that need to be taken into account in the modelling strategy [9]. In our case, the data series comes from a local low voltage substation in Belgium and it contains 28,000 hourly values. This paper is structured as follows. The description of the LS-SVM is presented in Section 2. In Section 3, the methodology for working in primal space is described, with the particular application to a large scale problem. Section 4 presents the problem and describes the setting for the estimation, and the results are reported in Section 5.

## 2 Function Estimation Using LS-SVM

The standard framework for LS-SVM estimation is based on a primal-dual formulation. Given the dataset  $\{\mathbf{x}_i, y_i\}_{i=1}^N$  the goal is to estimate a model of the form

$$y_i = \mathbf{w}^T \boldsymbol{\varphi}(\mathbf{x}_i) + b + e_i \quad (1)$$

where  $\mathbf{x} \in \mathbb{R}^n$ ,  $y \in \mathbb{R}$  and  $\boldsymbol{\varphi}(\cdot) : \mathbb{R}^n \rightarrow \mathbb{R}^{n_h}$  is a mapping to a high dimensional (and possibly infinite dimensional) feature space. The following optimization problem is formulated:

$$\min_{\mathbf{w}, b, e} \quad \frac{1}{2} \mathbf{w}^T \mathbf{w} + \gamma \frac{1}{2} \sum_{i=1}^N e_i^2 \quad (2)$$

$$\text{s.t. } y_i = \mathbf{w}^T \boldsymbol{\varphi}(\mathbf{x}_i) + b + e_i, \quad i = 1, \dots, N.$$

With the application of the Mercer's theorem for the kernel matrix  $\boldsymbol{\Omega}$  as  $\boldsymbol{\Omega}_{ij} = K(\mathbf{x}_i, \mathbf{x}_j) = \boldsymbol{\varphi}(\mathbf{x}_i)^T \boldsymbol{\varphi}(\mathbf{x}_j)$ ,  $i, j = 1, \dots, N$  it is not required to compute explicitly the nonlinear mapping  $\boldsymbol{\varphi}(\cdot)$  as this is done implicitly through the use of positive definite kernel functions  $K$ . For  $K(\mathbf{x}_i, \mathbf{x}_j)$  there are usually the following choices:  $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$  (linear kernel);  $K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T \mathbf{x}_j + c)^d$  (polynomial of degree  $d$ , with  $c$  a tuning parameter);  $K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|_2^2 / \sigma^2)$  (radial basis function, RBF), where  $\sigma$  is a tuning parameter. From the Lagrangian  $\mathcal{L}(\mathbf{w}, b, e; \alpha) = \frac{1}{2} \mathbf{w}^T \mathbf{w} + \gamma \frac{1}{2} \sum_{i=1}^N e_i^2 - \sum_{i=1}^N \alpha_i (\mathbf{w}^T \boldsymbol{\varphi}(\mathbf{x}_i) + b + e_i - y_i)$ , where  $\alpha_i \in \mathbb{R}$  are the Lagrange multipliers, the conditions for optimality are given by  $\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = 0 \rightarrow \mathbf{w} = \sum_{i=1}^N \alpha_i \boldsymbol{\varphi}(\mathbf{x}_i)$ ,  $\frac{\partial \mathcal{L}}{\partial b} = 0 \rightarrow \sum_{i=1}^N \alpha_i = 0$ ,  $\frac{\partial \mathcal{L}}{\partial e_i} = 0 \rightarrow \alpha_i = \gamma_i e_i$ ,  $i = 1, \dots, N$ , and  $\frac{\partial \mathcal{L}}{\partial \alpha_i} = 0 \rightarrow y_i = \mathbf{w}^T \boldsymbol{\varphi}(\mathbf{x}_i) + b + e_i$ . By elimination of  $\mathbf{w}$  and  $e_i$ , the following linear system is obtained:

$$\begin{bmatrix} 0 & \mathbf{1}^T \\ \mathbf{1} & \boldsymbol{\Omega} + \gamma^{-1} \mathbf{I} \end{bmatrix} \begin{bmatrix} b \\ \boldsymbol{\alpha} \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{y} \end{bmatrix}, \quad (3)$$

with  $\mathbf{y} = [y_1, \dots, y_N]^T$ ,  $\boldsymbol{\alpha} = [\alpha_1, \dots, \alpha_N]^T$ . The resulting LS-SVM model in dual space becomes

$$\mathbf{y}(\mathbf{x}) = \sum_{i=1}^N \alpha_i K(\mathbf{x}, \mathbf{x}_i) + b. \quad (4)$$

Usually the training of the LS-SVM model involves an optimal selection of the kernel parameters and the regularization parameter, which can be done using e.g. cross-validation techniques [8] or Bayesian inference [11].

### 3 Estimation in Primal Space

In order to work in the primal space, it is required to compute an explicit approximation of the nonlinear mapping  $\varphi$ . Then, the final estimation of the model can be done using different techniques.

#### 3.1 Nonlinear Approximation in Primal Space

Explicit expressions for  $\varphi$  can be obtained by means of an eigenvalue decomposition of the kernel matrix  $\boldsymbol{\Omega}$  with entries  $K(\mathbf{x}_i, \mathbf{x}_j)$ . Given the integral equation  $\int K(\mathbf{x}, \mathbf{x}_j) \phi_i(\mathbf{x}) p(\mathbf{x}) d\mathbf{x} = \lambda_i \phi_i(\mathbf{x})$ , with solutions  $\lambda_i$  and  $\phi_i$  for a variable  $\mathbf{x}$  with probability density  $p(\mathbf{x})$ , we can write

$$\varphi = [\sqrt{\lambda_1} \phi_1, \sqrt{\lambda_2} \phi_2, \dots, \sqrt{\lambda_{n_h}} \phi_{n_h}]. \quad (5)$$

Given the dataset  $\{\mathbf{x}_i, y_i\}_{i=1}^N$ , it is possible to approximate the integral by a sample average. This will lead to the eigenvalue problem (Nyström approximation [18])

$$\frac{1}{N} \sum_{k=1}^N K(\mathbf{x}_k, \mathbf{x}_j) u_i(\mathbf{x}_k) = \lambda_i^{(s)} u_i(\mathbf{x}_j), \quad (6)$$

where the eigenvalues  $\lambda_i$  and eigenfunctions  $\phi_i$  from the continuous problem can be approximated by the sample eigenvalues  $\lambda_i^{(s)}$  and eigenvectors  $u_i$  as

$$\hat{\lambda}_i = \frac{1}{N} \lambda_i^{(s)}, \hat{\phi}_i = \sqrt{N} u_i. \quad (7)$$

Based on this approximation, it is possible to compute the eigendecomposition of the kernel matrix  $\boldsymbol{\Omega}$  and use its eigenvalues and eigenvectors to compute the  $i^{\text{th}}$  required component of  $\hat{\varphi}(\mathbf{x})$  simply by applying (5) if  $\mathbf{x} \in \{\mathbf{x}_i\}_{i=1}^N$  (is a training point), or for any point  $\mathbf{x}^{(v)}$  by means of

$$\hat{\varphi}_i(\mathbf{x}^{(v)}) \propto \frac{1}{\sqrt{\lambda_i^{(s)}}} \sum_{k=1}^N u_{ki} K(\mathbf{x}_k, \mathbf{x}^{(v)}). \quad (8)$$

This finite dimensional approximation  $\hat{\varphi}(\mathbf{x})$  can be used in the primal problem (2) to estimate  $\mathbf{w}$  and  $b$ .

### 3.2 Sparseness and Large Scale Problems

It is important to emphasize that the use of the entire training sample of size  $N$  to compute the approximation of  $\varphi$  will yield at most  $N$  components, each one of which can be computed by (7) for all  $\mathbf{x} \in \{\mathbf{x}_i\}_{i=1}^N$ . However, if we have a large scale problem, it has been motivated [16] to use of a subsample of  $M \ll N$  datapoints to compute the  $\hat{\varphi}$ . In this case, up to  $M$  components will be computed. External criteria such as entropy maximization can be applied for an optimal selection of the subsample: given a fixed-size  $M$ , the aim is to select the support vectors that maximize the quadratic Renyi entropy [7]

$$H_R = -\log \int p(\mathbf{x})^2 d\mathbf{x}, \quad (9)$$

which can be approximated by using  $\int \hat{p}(\mathbf{x})^2 d\mathbf{x} = \frac{1}{M^2} \mathbf{1}^T \boldsymbol{\Omega} \mathbf{1}$ . The use of this active selection procedure can be important for large scale problems, as it is related to the underlying density distribution of the sample. In this sense, the optimality of this selection is related to the final accuracy that can be obtained in the modeling exercise.

### 3.3 Estimation Technique

Once the nonlinear mapping has been computed using a sparse approximation based on a subsample, the model has to be estimated in primal space. Let us denote by  $\mathbf{z}_k = \hat{\varphi}(\mathbf{x}_k)$  for  $k = 1, \dots, N$  and consider the new  $\mathbf{z}_k \in \mathbb{R}^m$  as the observations for the linear regression

$$\mathbf{y} = \mathbf{Z}\boldsymbol{\beta} + b\mathbf{1} + \mathbf{e} \quad (10)$$

with  $\mathbf{e} = [e_1, e_2, \dots, e_N]^T \in \mathbb{R}^{N \times 1}$ ,  $\mathbf{y} = [y_1, y_2, \dots, y_N]^T \in \mathbb{R}^{N \times 1}$  and  $\mathbf{Z} = [\mathbf{z}_1^T; \mathbf{z}_2^T; \mathbf{z}_3^T; \dots; \mathbf{z}_N^T] \in \mathbb{R}^{N \times m}$ . The quantity  $m$  is the number of components of  $\hat{\varphi}$  that are going to be considered in the estimation [5], with  $m \leq M$ . For ease of notation, consider the matrix of full regressors  $\mathbf{Z}_F = [\mathbf{1}\mathbf{Z}]$ , and the vector of full coefficients  $\boldsymbol{\beta}_F = [b, \boldsymbol{\beta}]^T$ . The regression (10) can be written as:

$$\mathbf{y} = \mathbf{Z}_F \boldsymbol{\beta}_F + \mathbf{e}. \quad (11)$$

In this paper, we estimate the above equation by Ordinary Least Squares (OLS), where only  $m < M$  components of  $\hat{\varphi}$  are used, and they are selected by looking at the eigenspectrum of the  $M \times M$  kernel matrix  $\boldsymbol{\Omega}$  [3].

## 4 Practical Example: Short-Term Load Forecasting

In this section the practical application is described, in term of the problem context, methodological issues and results.

## 4.1 Description and Objective

The main goal in load forecasting is to estimate a model that can capture all the dynamics between possible explanatory variables for the load, in order to produce forecasts from one hour ahead, up to one day ahead [12]. In the literature there is a broad consensus about important explanatory variables: past values of the load, weather information, calendar information, etc. For further details, the interested reader is referred to [4, 5]. Our objective is to show the application of the large-scale methodology described in the previous section, which is a practical approach to exploit the information contained in a large load time series.

## 4.2 Data and Methodology

The data consists of a 3.5-years load series from a particular substation in Belgium, thus containing more than 28,000 datapoints. We define our training set to consist of 26,000 datapoints. The model formulation to be tested is a nonlinear ARX specification, with the following structure: an autoregressive part of 164 lagged load values, covering a complete week; temperature-related variables measuring the effect of temperature on cooling and heating requirements [2]; and calendar information in the form of dummy variables for month of the year, day of the week and hour of the day [4]. This leads to a set of 211 explanatory variables. The procedure for modelling with the large-scale fixed-size LS-SVM methodology can be described in the following steps:

1. Normalize inputs and outputs to have zero mean and unit variance
2. Select an initial subsample of size  $M$
3. Build the  $M$ -size kernel matrix and compute its eigendecomposition
4. Build the nonlinear mapping approximation for the rest of the data
5. Estimate a linear regression in primal space
6. Estimate the nonlinear mapping approximation for a test data
7. Use the regression estimates with the test data nonlinear mapping to produce the out of sample forecast

In this example, tuning of the hyperparameter  $\sigma$  is performed by 10 fold cross-validation (CV) in the training sample. We keep the value of  $\sigma$  that minimizes the out-of-sample mean squared error (MSE). To illustrate the effect of increasing sizes of  $M$ , the above methodology is tested for sizes of  $M = 200, 400, 600, 800$  and 1000 support vectors, selected with the quadratic entropy criterion. It is important to emphasize that we are using between 1% and 4% of the available dataset to build the nonlinear mapping for the entire sample. Values of  $M$  larger than 1000 are possible, as the only constraint in this approach is the available computational resources at hand.

To compare the results of the nonlinear models, we estimate a linear regression with the same set of initial inputs. After estimation, the inputs that are not statistically significant are removed on a stepwise selection process based on the  $t$ -statistics, yielding a final set of 153 regressors.

## 5 Results

In this section the results of the fixed-size LS-SVM methodology applied to the load modelling problem are reported, for the training procedure, the selection of support vectors and the out of sample performance.

### 5.1 Training Performance

The results for the computed MSE in a crossvalidation basis, and the equivalent result for the linear model, are shown in Table 1 using the selected  $\sigma$ . It is also reported the number of support vectors  $m$  that are used for building the nonlinear mapping on each case.

### 5.2 Support Vector Selection

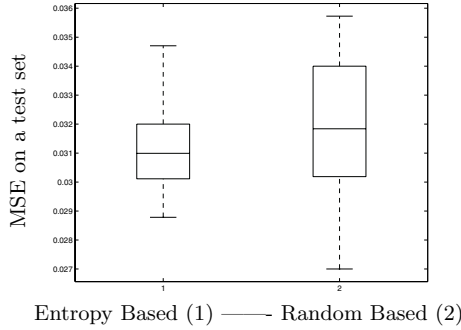
It is possible to compare the performances for a model estimated with a random selection of support vectors with respect to the same model estimated with a quadratic Renyi entropy selection starting from the same random selection. Both models can be compared in terms of performance on the same test set. Figure 1 show the comparison for the results after 20 random initial selections, in which the model is either estimated after quadratic entropy selection taking the random selection as the initial starting point (Case I), or it is estimated directly (Case II). In all tests it has been used  $M = 200$ . Clearly starting from different random selections, the entropy-based selection yields lower dispersion in the errors.

**Table 1.** CV performance of fixed-size LS-SVM with  $M$  support vectors

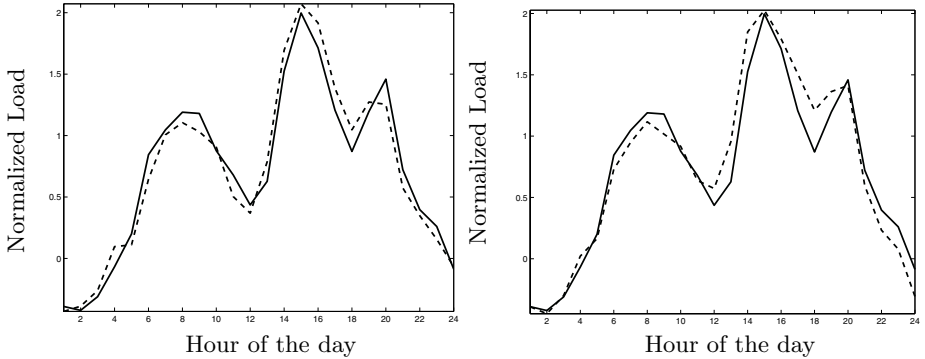
| Estimation | MSE (CV) | Selected components $m$ |
|------------|----------|-------------------------|
| Linear     | 0.043    | 153                     |
| M=200      | 0.032    | 166                     |
| M=400      | 0.022    | 327                     |
| M=600      | 0.017    | 416                     |
| M=800      | 0.016    | 462                     |
| M=1000     | 0.015    | 496                     |

**Table 2.** Performance on a test set for different forecasting horizons

| Model  | MSE test (one-hour-ahead) | MSE test (24-hours-ahead) |
|--------|---------------------------|---------------------------|
| Linear | 0.038                     | 0.042                     |
| M=200  | 0.027                     | 0.031                     |
| M=400  | 0.019                     | 0.025                     |
| M=600  | 0.016                     | 0.023                     |
| M=800  | 0.014                     | 0.021                     |
| M=1000 | 0.014                     | 0.021                     |



**Fig. 1.** Box-plot of the MSE in a test set for models estimated with entropy-based (1) and random (2) selection of support vectors. Results for 20 repetitions



**Fig. 2.** Forecasts (dashed lines) and actual values (full line) for a selected 24-hours period, for the case of one-hour-ahead with  $M = 400$  (Left); and 24-hours-ahead with  $M = 800$  (Right)

### 5.3 Out of Sample Performance

The estimated models are tested in a one-hour-ahead basis and a 24-hours-ahead prediction with iterative simulations running for the next 24 hours. Averages of the MSE for each case are reported on Table 3. An example of one-hour-ahead prediction for the case  $M = 400$  and 24-hours-ahead prediction for  $M = 800$  are shown in Figure 2.

## 6 Conclusion

The methodology of fixed-size LS-SVM has been applied to short-term load forecasting. This paper has shown that it is possible to build a large scale nonlinear regression model from a dataset with 26,000 samples using different subsamples as support vectors, with satisfactory results. The results show that the nonlinear regressions in primal space improve their accuracy with larger values of  $M$ .

The maximum value of  $M$  to be used depends on the computational resources at hand, and it also depends on the underlying distributional properties of the dataset. In this context, it was shown that quadratic entropy active selection of support vectors leads to performances which are less disperse as those obtained by random selection of support vectors. The forecasting performance is very good, with MSE's around 2% for both forecasting modes (one-hour-ahead and 24-hours-ahead) when the nonlinear mapping is computed using 800 selected support vectors, which represent 3% of the sample available. Further research on a more dedicated definition of the initial input variables should lead to further improvements.

**Acknowledgments.** This work is supported by grants and projects for the Research Council K.U.Leuven (GOA- Mefisto 666, GOA- Ambiorics, several PhD/ Postdocs & fellow grants), the Flemish Government (FWO: PhD/ Postdocs grants, projects G.0211.05, G.0240.99, G.0407.02, G.0197.02, G.0141.03, G.0491.03, G.0120.03, G.0452.04, G.0499.04, IC-CoS, ANMMM; AWI; IWT: PhD grants, GBOU (McKnow) Soft4s), the Belgian Federal Government (Belgian Federal Science Policy Office: IUAP V-22; PODO-II (CP/ 01/40), the EU (FP5- Quprodix; ERNSI, Eureka 2063- Impact; Eureka 2419- FLiTE) and Contracts Research / Agreements (ISMC /IPCOS, Data4s, TML, Elia, LMS, IPCOS, Mastercard). J. Suykens and B. De Moor are an associated professor and a full professor at the K.U.Leuven, Belgium, respectively. The scientific responsibility is assumed by its authors.

## References

1. Cristianini, N., and Shawe-Taylor, J., *An introduction to Support Vector Machines*. Cambridge University Press, 2000.
2. Engle, R., Granger, C.J., Rice, J., and Weiss, A. "Semiparametric Estimates of the Relation Between Weather and Electricity Sales," *Journal of the American Statistical Association*, Vol.81, No.394, 310-320, 1986.
3. Espinoza, M., Suykens, J.A.K. and De Moor B. "Least Squares Support Vector Machines and Primal Space Estimation," in Proc. of the *IEEE 42nd Conference on Decision and Control*, Maui, USA, 2003, pp. 3451-3456.
4. Espinoza, M., Joye, C., Belmans, R., and De Moor, B. "Short Term Load Forecasting, Profile Identification and Customer Segmentation: A Methodology based on periodic time series," Internal Report 04-141, ESAT-SISTA, K.U.Leuven, 2004.
5. Espinoza, M., Suykens, J.A.K. and De Moor, B. "Fixed Size Least Squares Support Vector Machines: A Large Scale Application in Electrical Load Forecasting," Internal Report 03-92, ESAT-SISTA, K.U.Leuven, 2003.
6. Fay, D., Ringwood, J., Condon, M. and Kelly, M. "24-h Electrical Load Data-A Sequential or Partitioned Time Series?," *Neurocomputing* 55, 469-498, 2003.
7. Girolami, M. "Orthogonal Series Density Estimation and the Kernel Eigenvalue Problem," *Neural Computation* 14(3), 669-688, 2003.
8. Golub, G., Heath, M. and Wahba, G. "Generalized cross-validation: A method for choosing a good ridge regression parameter," *Technometrics* 21, 215-223, 1979.
9. Hylleberg, S. *Modelling Seasonality*. Oxford University Press, 1992.



10. Ljung, L. *Systems Identification: Theory for the User*. Prentice Hall, 1999.
11. MacKay, D.J.C. "The evidence framework applied to classification networks," *Neural Computation*, 4, 698-714, 1992.
12. Mariani, E. and Murthy, S.S. *Advanced Load Dispatch for Power Systems*. Advances in Industrial Control, Springer-Verlag, 1997.
13. Ramanathan, R., Engle, R., Granger, C.W.J., Vahid-Aragui, F., Brace, C. "Short-run Forecasts of Electricity Load and Peaks," *Int. J. of Forecasting* 13, 1997.
14. Steinerz, H., Pedreira, C. and Castro, R. "Neural Networks for Short-Term Load Forecasting: A Review and Evaluation," *IEEE Trans. Power Syst.*, Vol.16. No.1, 2001.
15. Suykens, J.A.K., Vandewalle, J. "Least Squares Support Vector Machines Classifiers," *Neural Processing Letters* 9, 293-300, (1999).
16. Suykens J.A.K., Van Gestel T., De Brabanter J., De Moor B., Vandewalle J., *Least Squares Support Vector Machines*. World Scientific, 2002, Singapore.
17. Vapnik, V. *Statistical Learning Theory*. John Wiley & Sons, New York, 1998.
18. Williams, C.K.I and Seeger, M. "Using the Nyström Method to Speed Up Kernel Machines," in T.Leen, T.Dietterich, V.Tresp (Eds.), *Proc. NIPS'2000*, vol 13., MIT press.

# TaSe Model for Long Term Time Series Forecasting

Luis Javier Herrera, Héctor Pomares, Ignacio Rojas, Alberto Guillén,  
Olga Valenzuela, and Alberto Prieto

University of Granada, Department of Computer Architecture and Technology,  
E.T.S. Computer Engineering, 18071 Granada, Spain  
<http://atc.ugr.es>

**Abstract.** There exists a wide range of paradigms and a high number of different methodologies applied to the problem of Time Series Prediction. Most of them are presented as a modified function approximation problem using I/O data, in which the input data is expanded using outputs at previous steps. Thus the model obtained normally predicts the value of the series at a time  $(t + h)$  using previous time steps  $(t - \tau_1), (t - \tau_2), \dots, (t - \tau_n)$ . Nevertheless, learning a model for long term time series prediction might be seen as a completely different task, since it will generally use its own outputs as inputs for further training, as in recurrent networks. In this paper we present the utility of the TaSe model using the well-known Mackey Glass time series and an approach that upgrades the performance of the TaSe one-step-ahead prediction model for long term prediction.

## 1 Introduction

There exist several methodologies and approaches to deal with time series prediction problems. The TaSe model, that was first presented in [3] and whose full learning methodology was fully developed in [4], is a modified TSK model [1] that uses a Grid Partitioning (thus a Grid-Based Fuzzy System GBFS [5] type) of the input space that partially overcomes the two main drawbacks of this type of models: the curse of dimensionality problem and the lack of interpretability.

The curse of dimensionality problem has to do with the exponential growth in the number of rules when the number of input dimensions and the number of membership functions per input variable grow. The lack of interpretability has to do with different aspects that fuzzy models should include but that most of the times are lost in the design and in the learning process: the transparency of the model (transparent partition of the input space), the need of maintaining a relatively low number of rules [7] (very related thus with the curse of dimensionality problem) and, more specifically for the TSK models, the interaction of the global and local models [8] (that should allow us to interpret the rules comprising the TSK system as linearizations of the nonlinear subjacent system).

The TaSe model considers these intrinsic aspects of the TSK fuzzy systems, providing high accuracy for function approximation and time series prediction

problems [4] and providing full interpretability to the polynomial consequents using a fundamental key in the field of Function Approximation Theory and Mathematical Analysis, as is the Taylor Theorem. The TaSe model allows us to interpret its rules consequents as the Taylor Series Expansion of the model output around the rule centres, thus giving an explicit interpretation of the model.

The rest of the paper stays as follows: In section 2, we will introduce the main concepts related to the TaSe methodology and its application to the well known Mackey-Glass time series benchmark [2]. Section 3 briefly presents a modified point of view of the time series forecasting problem for long term prediction problems. In section 4 we present a modified training of the TaSe model to improve the prediction of long term time series and section 5 concludes the paper.

## 2 The TaSe Model Learning Methodology

In general, the structure of a multiple-input single-output (MISO) Takagi-Sugeno-Kang (TSK) [1] fuzzy system and its associated fuzzy inference method comprises a set of  $K$  IF-THEN rules in the form

$$Rule^k : \text{IF } x_1 \text{ is } \mu_1^k \text{ AND } \dots \text{ AND } x_n \text{ is } \mu_n^k \text{ THEN } y = R^k \quad (1)$$

where the  $\mu_i^k$  are fuzzy sets characterized by membership functions (MF)  $\mu_i^k(x_i)$  in universes of discourse  $U_i$  (in which variables  $x_i$  take their values), and  $R_k$  are the consequents of the rules.

The output of a fuzzy system with rules in the form shown in (1) can be expressed (using weighted average aggregation) as

$$F(\mathbf{x}) = \frac{\sum_{k=1}^K \mu^k(\mathbf{x}) R^k}{\sum_{k=1}^K \mu^k(\mathbf{x})} \quad (2)$$

provided that  $\mu^k(\mathbf{x})$  is the activation value for the antecedent of the rule  $k$ , which can be expressed as

$$\mu^k(\mathbf{x}) = \mu_1^k(x_1) \mu_2^k(x_2) \dots \mu_n^k(x_n). \quad (3)$$

Now we briefly review the peculiarities and specifications of the TaSe model as well as its learning algorithm. For a complete presentation please refer to [4].

### 2.1 TaSe Specifications

The TaSe model uses rule consequents ( $R^k$  in (1)) in the form

$$f(\mathbf{x}) = f(\mathbf{a}) + (\mathbf{x} - \mathbf{a})^T \left[ \frac{\partial f}{\partial \mathbf{x}_i}(\mathbf{a}) \right]_{i=1 \dots n} + \frac{1}{2} (\mathbf{x} - \mathbf{a})^T W (\mathbf{x} - \mathbf{a}) + \dots + \quad (4)$$

(where  $a$  is the rule centre specified in the respective rule antecedent), i.e. in the form specified by the Taylor Theorem to approximate any function  $f(\mathbf{x})$  as a sequence of polynomials.

But in order to be able to interpret the rule consequents as the Taylor Series Expansion of the model output with respect to the rule centres, two properties are required: first the model output must be continuous and differentiable at least as many times as the order of the polynomial form used in the consequents, and second, the activation value of every MF must be zero at each rule centre, except for the MFs of the respective rule, so that the rule consequent is the only function specified for that precise point (the rule centre). The OLMF basis, first presented in [6], provides such two properties to the fuzzy model and thus will allow us to obtain an interpretable model for function approximation from a set of I/O data [4].

## 2.2 TaSe Learning Methodology

TSK fuzzy learning implies two parallel processes: structure identification (that deals with finding the optimal input space subdivision/number-of-rules according to the specified training I/O data) and parameter adjustment, both for rule antecedents and for rule consequents.

With respect to the parameter adjustment, the rule consequents coefficients can be linearly obtained using Least Squares to solve the error function

$$J = \sum_{m \in D} (f(\mathbf{x}^m) - z^m)^2. \quad (5)$$

The rule antecedents parameters (specifically only the centres positions have to be optimized) will be optimized using the local-search Levenberg-Marquardt algorithm.

The structure identification approach used for the TaSe model uses an incremental iterative method, that will add one MF per iteration in the variable selected to be the optimal one to receive such MF. The structure identification algorithm stops whenever the validation error stops decreasing [4].

## 2.3 Application of the TaSe Model to the Mackey-Glass Time Series Prediction Problem

The Mackey-Glass time series [2] is a very well known time series benchmark, that has been widely used in the literature. This time series is described by the following delay differential equation

$$\frac{dx(t)}{dt} = \frac{ax(t-\tau)}{1+x^{10}(t-\tau)} - bx(t) \quad (6)$$

A sequence of 1200 data points were generated with an initial condition  $x(0) = 1.2$  and  $\tau = 17$  using the 4th order Runge-Kutta method. To make the

**Table 1.** Trace of the structure identification algorithm for the Mackey Glass time series prediction problem

| MFs       |           |          |        | N Rules | RMSE (for predicting $x(t+6)$ ) |            |        |
|-----------|-----------|----------|--------|---------|---------------------------------|------------|--------|
| $x(t-18)$ | $x(t-12)$ | $x(t-6)$ | $x(t)$ | –       | Training                        | Validation | Test   |
| 1         | 1         | 1        | 1      | 1       | 0.032                           | 0.029      | 0.034  |
| 1         | 2         | 1        | 1      | 2       | 0.0084                          | 0.008      | 0.0098 |
| 2         | 2         | 1        | 1      | 4       | 0.0032                          | 0.0037     | 0.0035 |
| 3         | 2         | 1        | 1      | 6       | 0.0024                          | 0.0027     | 0.0028 |
| 3         | 2         | 1        | 2      | 12      | 0.0020                          | 0.0024     | 0.0024 |
| 3         | 3         | 1        | 2      | 18      | 0.0014                          | 0.0020     | 0.0017 |
| 3         | 3         | 1        | 3      | 27      | 0.0011                          | 0.0018     | 0.0013 |

comparisons with earlier works fair, we chose the parameters so that the training vectors for the model have the following format

$$[x(t-18), x(t-12), x(t-6), x(t); x(t+6)] \tag{7}$$

In *Table 1* we show the evolution of the structure identification learning algorithm (that iterates increasing the system complexity until the validation error stops decreasing), taking the last 1000 data of the 1200, using 500 data for training and validation and the final 500 for testing. The algorithm starts with an initial MF configuration of 1 MF per variable and iterates adding 1 MF to the selected variable.

*Table 2* compares the prediction error for different computational paradigms presented in the bibliography for this benchmark problem (taken from [9]). In order to perform an equal comparison (also in complexity), we show our test error for three MF configurations:  $3 \times 2 \times 1 \times 1$  with 91 ( $156 + 1$  mobile centre) parameters and  $3 \times 2 \times 1 \times 2$  with 181 ( $15 \times 12 + 1$  mobile centre) parameters and  $3 \times 3 \times 1 \times 3$  with 408 ( $15 \times 27 + 3$  mobile centres). The results drawn from this example show the convenience of the TaSe model due to its high degree of accuracy for function approximation and time series prediction problems; with only 6 rules, the TaSe model is able to identify the non-linear behaviour of the Mackey-Glass time series better than many other time series prediction methods proposed in the literature (with a similar model complexity), being the consequents of these rules the Taylor Series Expansion of the output of the model centred in their respective rule centres.

### 3 Long Term Predictions

So far we have presented an approach for modelling a set of I/O in which the objective is to predict the output  $\hat{x}(t+6)$  assuming we know the inputs specified in (7). But for long term prediction problems, the objective could be expressed as: “having a short sequence of data, try to predict the rest of the data sequence

**Table 2.** Comparison results of the prediction error of different methods for prediction step equals to 6 (500 training data) See [9]

| Method                                           |       | Prediction Error (RMSE) |
|--------------------------------------------------|-------|-------------------------|
| Auto Regressive Model                            |       | 0.19                    |
| Cascade Correlation NN                           |       | 0.06                    |
| Back-Prop. NN                                    |       | 0.02                    |
| 6th-order Polynomial                             |       | 0.04                    |
| Linear Predictive Method                         |       | 0.55                    |
| Kim and Kim (Genetic Algorithm and Fuzzy System) | 5 MFs | 0.049206                |
|                                                  | 7 MFs | 0.042275                |
|                                                  | 9 MFs | 0.037873                |
| ANFIS and Fuzzy System (16 rules)                |       | 0.007                   |
| Classical RBF (with 23 neurons)                  |       | 0.0114                  |
| TaSe Fuzzy System with 6 rules                   |       | 0.0028                  |
| TaSe Fuzzy System with 12 rules                  |       | 0.0024                  |
| Optimal TaSe Fuzzy System with 27 rules          |       | 0.0013                  |

until we reach a certain time-step". Thus the training data for this model would be that of (7), but in fact, the application of the model to predict the complete data sequence should use iteratively the training data

$$\hat{x}(t+6) = f(\hat{x}(t-18), \hat{x}(t-12), \hat{x}(t-6), \hat{x}(t)) \quad (8)$$

and this is not what our initial model has been trained for.

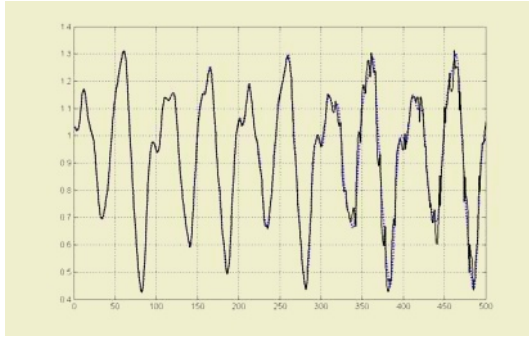
Suppose a toy example of a series  $x = \{1, 2.5, 3.5, 4\}$ . A model that tries to predict  $\hat{x}(t+1) = f(x(t))$  would use as training set the pairs  $[1; 2.5]$ ,  $[2.5; 3.5]$ ,  $[3.5; 4]$ . Thus using least squares, a simple linear model that uses a single constant parameter  $\hat{x}(t+1) = a * x(t)$  would solve the equation

$$[1, 2.5, 3.5]' * a = [2.5, 3.5, 4]' . \quad (9)$$

The solution is  $a = 1.29$  and the MSE = 0.6. But in fact, the long term prediction training error, starting from the initial value  $x(t_0) = 1$  and predicting the following values using the model output is MSE = 2.72 (the "long term" prediction would be  $[1.29, 1.29^2, 1.29^3]$ ). The right way thus to train a long term prediction model would be

$$[1, 1 * a, 1 * a^2]' * a = [2.5, 3.5, 4]' \quad (10)$$

whose solution is  $a = 1.67$  and the "long term" prediction MSE is 0.54. Nevertheless even for this toy example, this is a nonlinear optimization problem with very high complexity, and thus the traditional function approximation modelling techniques don't work here.



**Fig. 1.** Original (dotted line) and approximated (solid line) long term prediction for the test data set

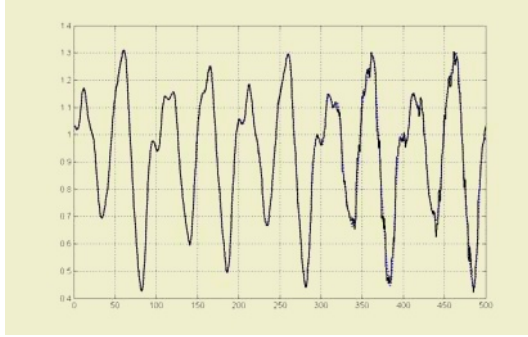
### 4 Long Term Prediction Using the TaSe One-Step Ahead Prediction Model

We have barely seen the overwhelming recursive task that would be to model directly a long term prediction problem from an I/O dataset. Thus, we could try to use directly the one-step ahead prediction model to perform that task. Using the optimal model obtained as shown in *Table 1*, and using the obtained outputs as new inputs (thus using in total 23 (18+5) known outputs to predict 500 unknown values), *Fig. 1* shows the long term prediction for the test data set. The RMSE obtained for the training data set equals 0.030 and for the test data set is 0.031.

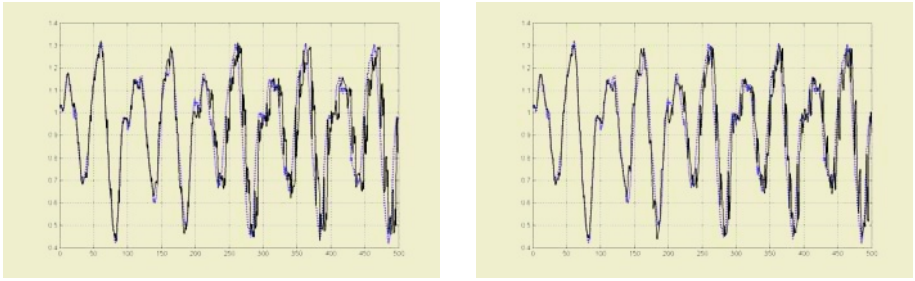
From these results we see that we have obtained a pretty good solution for this long term prediction problem. Thanks to the highly accurate TaSe optimal model, we have been able to obtain a sequence of outputs very similar to the original one.

Nevertheless we present here a modification in the training stage that will allow us to obtain better results. In [10], Schoelkopf et al. expanded the training data by adding small modifications in each training item for the specific classification problem. Similarly we could expand our training data for the series so that we consider small modifications in time in this case. But this is an interpolation problem and the task in some cases is at least as difficult as the prediction problem. Nevertheless in this case, in the training problem we have obtained a very good approximation of the time series (training RMSE = 0.0011). Thus, using the same optimal model, we have expanded the training data to consider a new section of the time series that is a small modification of the original time series, which, in fact presents the same behaviour. *Fig. 2* shows the long term prediction for the test data set using the re-trained optimal TaSe model. The RMSE obtained for the training data set is 0.015 and for the test data set equals 0.019.

Further extensions of the training data didn't bring further significant results in this case. Now consider the original Mackey-Glass time series and an added



**Fig. 2.** Original (dotted line) and approximated (solid line) long term prediction for the test data set using the expanded training data set



**Fig. 3.** a) Original (dotted line) and approximated (solid line) long term prediction for the test data set using the original training data set for the noisy problem. b) Original (dotted line) and approximated (solid line) long term prediction for the test data set using the expanded training data set for the noisy problem

noise component with variance  $0.05 * \sigma_x$  where  $\sigma_x$  is the standard deviation of the time series. The optimal TaSe model has in this case only 12 rules, due to the noisy behaviour. In this case, we performed an iterative process as that done for the noiseless series. We used once and again the training output as new training data in the form shown in (7) to obtain the optimal rule consequents. *Fig. 3* shows the long term prediction for the test data set using the simple training data (*Fig. 3a*) with RMSE 0.12 and 0.13 for the training and test datasets, and re-trained optimal TaSe model in the optimal iteration according to the long term prediction of the training data (*Fig. 3b*), for which the RMSE obtained is 0.10 for both datasets.

Thus this process can be seen as a time series de-noising process. If we find a single output model that is accurate enough and that is able to partially get rid of the noise, the same model output can be used as training in order to obtain better results when performing long term prediction using the single output model that predicts a single  $\hat{x}(t+h) = f(x(t-\tau_0), \dots, x(t-\tau_n))$ .



## 5 Conclusions

In this paper we have briefly reviewed the TaSe fuzzy methodology for function approximation and its applicability to time series prediction using the well-known Mackey Glass time series. Since a complete long term prediction learning methodology implies an overwhelming learning task, we have proposed to use the one-step ahead model obtained with the TaSe model to perform a long term prediction of the series. We have also seen that we could expand the training dataset by considering the output of our model as another sequence of the same series, and that this training data set expansion can provide better results.

## Acknowledgements

This work has been partially supported by the Spanish CICYT Project TIN2004-01419.

## References

1. Takagi, T., Sugeno, M.: Fuzzy identification of systems and its applications to modelling and control. *IEEE Trans. Syst. Man and Cyber.* **15** (1985) 116–132
2. Mackey, M.C., Glass, L.: Oscillation and chaos in physiological control systems. *Science.* **197** (1977) 287–289
3. Herrera, L.J., Pomares, H., Rojas, I., González, J., Valenzuela, O.: Function Approximation through Fuzzy Systems Using Taylor Series Expansion-Based Rules: Interpretability and Parameter Tuning. *Lecture Notes in Computer Science.* New York: Springer-Verlag, **2972** (2004) 252–261
4. Herrera, L.J., Pomares, H., Rojas, I., Valenzuela, O., Prieto, A.: TaSe, a Taylor Series Based Fuzzy System Model that Combines Interpretability and Accuracy. *Fuzzy Sets and Systems.* *Accepted*
5. Pomares, H., Rojas, I., Ortega, J., Prieto, A.: A systematic approach to a self-generating fuzzy rule-table for function approximation. *IEEE Trans. Syst., Man, Cybern.* **30** (2000) 431–447
6. Bikdash, M.: A Highly Interpretable Form of Sugeno Inference Systems. *IEEE Trans. Fuz. Syst..* **7(6)** (1999) 686–696
7. Guillaume, S.: Designing fuzzy inference systems from data: an interpretability-oriented review. *IEEE Trans. Fuzzy Systems,* **9** (2001) 426–443.
8. Johansen, T.A., Babuska, R.: Multiobjective Identification of Takagi-Sugeno Fuzzy Models. *IEEE Trans. Fuz. Syst.* **11(6)** (2003) 847–860
9. Rojas, I., Pomares, H., Bernier, J.L., Ortega, J., Pino, B., Pelayo, F.J., Prieto, A.: Time Series analysis using normalized PG-RBF network with regression weights. *NeuroComputing,* **42**, (2002) 267–285
10. Decoste, D., Schoelkopf, B.: Training Invariant Support Vector Machines. *Machine Learning,* **46**, (2002) 161–190

# Connections Between ICA and Sparse Coding Revisited

Susana Hornillo-Mellado<sup>1</sup>, Rubén Martín-Clemente<sup>1</sup>, and Juan M. Górriz-Sáez<sup>2</sup>

<sup>1</sup> Dpto. de Teoría de la Señal y Comunicaciones, Escuela Superior de Ingenieros, Universidad de Sevilla, Avda. de los Descubrimientos s/n, 41092-Sevilla, Spain  
{susanah, ruben}@us.es

<sup>2</sup> Dpto. de Teoría de la Señal, Telemática y Comunicaciones, E.T.S. de Ingeniería Informática, Universidad de Granada, c/ Periodista Daniel Saucedo s/n, 18071-Granada, Spain  
gorriz@ugr.es

**Abstract.** Recently, the application of Independent Component Analysis (ICA) to natural images has raised a great interest. Some outstanding features have been observed, like the sparse distribution of the independent components and the special appearance of the ICA bases (most of them look like edges). This paper provides a new insight on this behaviour, being supported by experimental results. In particular, a mathematical proof of the relation between ICA and sparse coding is given.

## 1 Introduction

Independent Component Analysis (ICA) is an emergent technique for studying empirical data [3][4][9]. It involves a mathematical procedure that transforms a number of zero-mean *observed variables*  $x_1, \dots, x_N$  into a number of “as statistically independent as possible” variables  $s_1, \dots, s_N$  called *independent components*. In linear ICA, this transformation reads:

$$s_i = \sum_{n=1}^N b_{in}x_n, \forall i = 1, \dots, N \quad (1)$$

where  $b_{in}, n = 1, \dots, N$ , are real coefficients. In matrix form,

$$\mathbf{S} = \mathbf{B} \mathbf{X} \iff \mathbf{X} = \mathbf{A} \mathbf{S} \quad (2)$$

where  $\mathbf{B} = (b_{ij})$  and  $\mathbf{A} = \mathbf{B}^{-1} = (a_{ij})$ .

The application of ICA to natural images has raised a great interest (some results can be found in [6][7][9]) because of two outstanding observed features. First, the special appearance of the ICA bases (the columns of matrix  $\mathbf{A}$ ): most of them look like edges [9]. Second, it is obtained that the distribution of the independent components is sparse. Both facts were first observed by Bell and Sejnowski [2], who linked them with some properties of the human visual system

and proposed an Information-Theoretic model to explain the behavior of the neurons of the visual cortex (see also [1][5][11]). In particular, they hypothesized that a certain kind of cortical neurons (specifically, the *simple cells*) perform an independent component analysis of the visual information received by the photoreceptors in the retinas. The objective of this paper is to analyze these experimental results and provide a mathematical basis to explain the connection between ICA and sparse coding.

## 2 Mathematical Analysis

To apply ICA to an image  $\mathbf{I}$ , we divide it into  $N_2$  patches of  $k_1 \times k_2$  pixels each. These patches are stacked into the columns of matrix  $\mathbf{X}$ , obtaining  $\mathbf{X} = [\mathbf{x}_1 | \mathbf{x}_2 | \dots | \mathbf{x}_{N_2}]$  where  $\mathbf{x}_j = [x_{1j}, x_{2j}, \dots, x_{N_1j}]^T$  contains the  $j$ -th patch, with  $j = 1, 2, \dots, N_2$  and  $N_1 = k_1 k_2$ . We consider each row of matrix  $\mathbf{X}$  as a realization of a random process.

There are different ways to carry out the independent component analysis of empirical data. Among them, consider the maximization of high-order cumulants, like skewness and kurtosis, which is a very popular criterion that is inspired in the Central Limit Theorem [4][9][10]. Let us define  $\mathbf{y} = \mathbf{b}^T \mathbf{X} = [y_1, y_2, \dots, y_{N_2}]$ , where  $y_i = \mathbf{b}^T \mathbf{x}_i$ . Our objective is to find the vector  $\mathbf{b}$  that solves the following constrained optimization problem:

$$\max_{\mathbf{b}} J_p(\mathbf{y}) = \frac{1}{N_2} \sum_{j=1}^{N_2} y_j^p \quad \text{subject to} \quad \frac{1}{N_2} \sum_{j=1}^{N_2} y_j^2 = 1 \quad (3)$$

with  $p > 2$ . The restriction is necessary to avoid the solution  $\mathbf{b} \rightarrow \infty$ . When  $p = 3$ ,  $J_p(\mathbf{y})$  is an estimation of the skewness. If  $p = 4$ ,  $J_p(\mathbf{y})$  leads to the kurtosis.

The Lagrangian  $L(\mathbf{y}, \lambda)$  is given by:

$$L(\mathbf{y}, \lambda) = J_p(\mathbf{y}) - \lambda \left( \sum_{j=1}^{N_2} y_j^2 - N_2 \right) \quad (4)$$

where  $\lambda$  is the Lagrange multiplier. It is shown in the Appendix that the stationary points of  $L(\mathbf{y}, \lambda)$  are the solutions to:

$$\begin{aligned} \frac{\partial}{\partial \mathbf{b}} L(\mathbf{y}, \lambda) &= \frac{1}{N_2} (p \mathbf{X} \mathbf{y}^{p-1} - 2\lambda \mathbf{X} \mathbf{y}^1) = \mathbf{0} \\ \frac{\partial}{\partial \lambda} L(\mathbf{y}, \lambda) &= \frac{1}{N_2} \|\mathbf{y}^1\|_2^2 - 1 = 0 \end{aligned} \quad (5)$$

where  $\frac{\partial}{\partial \mathbf{b}} L(\mathbf{y}, \lambda)$  is a  $N_1 \times 1$  vector whose  $k$ -th component is  $\frac{\partial}{\partial b_k} L(\mathbf{y}, \lambda)$ , and

$$\mathbf{y}^k \stackrel{def}{=} \begin{bmatrix} y_1^k \\ y_2^k \\ \dots \\ y_{N_2}^k \end{bmatrix} \quad (6)$$

One class of solutions is obtained when  $p\mathbf{y}^{p-1} = 2\lambda\mathbf{y}^1$ , i.e., either  $y_j = 0$  or:

$$y_j = \left(\frac{2\lambda}{p}\right)^{\frac{1}{p-2}} \quad (7)$$

for each  $j = 1, 2, \dots, N_2$ , with the restriction  $\|\mathbf{y}^1\|_2 = \sqrt{N_2}$ . *Sparse* solutions are possible<sup>1</sup>. For example, consider

$$y_1 = y_2 = \dots = y_M = \left(\frac{2\lambda}{p}\right)^{\frac{1}{p-2}} \quad (8)$$

with  $M \ll N_2$  where

$$\lambda = \frac{p}{2} \left(\frac{N_2}{M}\right)^{\frac{p-2}{2}} \quad (9)$$

and  $y_n = 0$  for all  $n > M$ . It means that  $\mathbf{b}$  is perpendicular to  $N_2 - M$  patches  $\mathbf{x}_n$ , i.e., to most of the image. *In practice this is what happens when we work with natural images.* The fact that the sources have an sparse distribution was first observed by Bell *et al* [2], who explained this result after identifying  $\mathbf{b}$  with response patterns of simple neurons of the visual cortex (they hypothesized that these neurons perform an independent component analysis of the visual information received by the photoreceptors in the retinas<sup>2</sup>). *Our preceding mathematics suggest now a complementary explanation:* experiments show that, in natural images, the patches  $\mathbf{x}_j$  are almost collinear to each other unless they correspond to edges. Thus, a vector  $\mathbf{b}$  that is orthogonal to most of the image (excepting to the edges) is theoretically possible, and therefore an sparse solution is obtained. The key is that, as shown above, ICA algorithms *look for such a vector*  $\mathbf{b}$  since it produces sparse solutions and maximize the higher-order cumulants of the data.

### 3 Example

Let us take the natural grey-scale image showed in Fig. 1. We divide it into  $8 \times 8$  patches to obtain the matrix of observations and apply ICA.

As we shown in the previous section the distribution of matrix  $\mathbf{S}$  should be sparse. In Fig. 2 we show some of the rows of matrix the  $\mathbf{S}$  corresponding to the ‘‘Lena’’ image.

Interestingly, the  $(k_1 k_2 \times k_1 k_2)$  matrix  $\mathbf{A}$  has a very particular structure. If we arrange its columns into  $(k_1 \times k_2)$  blocks and represent them as images, we observe that most of them look like *edges* with different orientations, as shown in Fig. 3. These images are usually called *ICA bases*, and their special appearance was first observed also by Bell *et al* [2] (see also [1][5]).

<sup>1</sup> Specifically, it can be easily shown that (3) attains a global maximum when one and only one of the  $y_j$  is different from zero.

<sup>2</sup> Other works have dealt with this subject (see, for example [8]), but no mathematical analysis has been given so far.



Fig. 1. The “Lena” image (256 × 256)

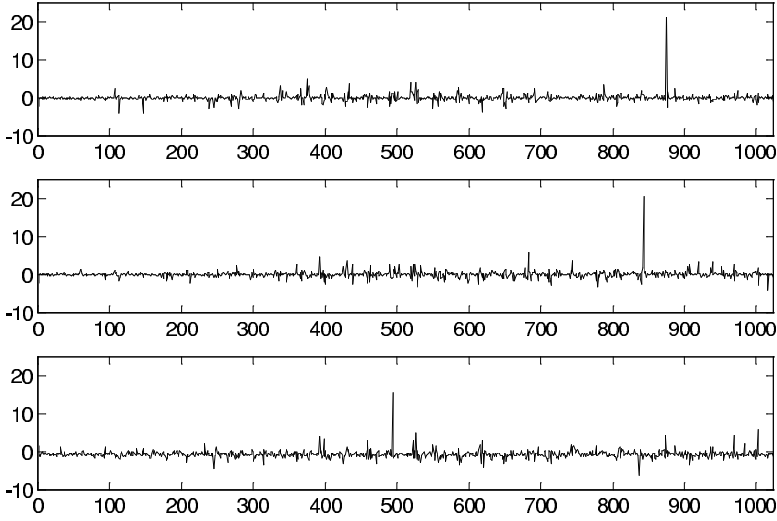
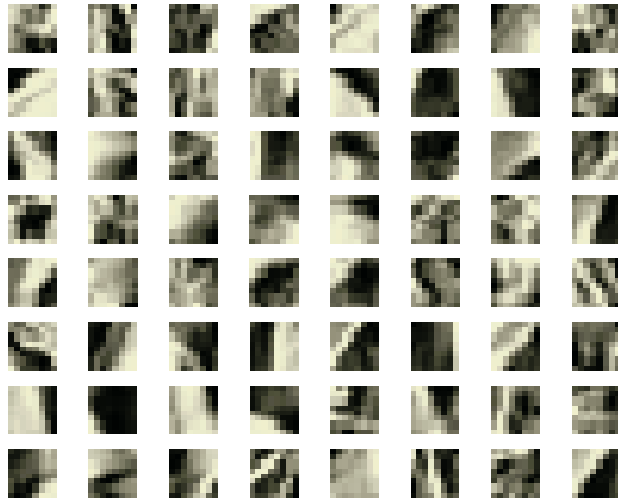


Fig. 2. Some rows of matrix  $\mathbf{S}$

Let us return to the ICA model (2). Since  $\mathbf{X} = \mathbf{A}\mathbf{S}$ , the  $i$ -th column of  $\mathbf{X}$ ,  $\mathbf{x}_i = [x_{1i}, x_{2i}, \dots, x_{N_1i}]^T$ ,  $i = 1, 2, \dots, N_2$ , can be written as:

$$\mathbf{x}_i = \mathbf{A}\mathbf{s}_i = s_{1i}\mathbf{a}_1 + s_{2i}\mathbf{a}_2 + \dots + s_{N_1i}\mathbf{a}_{N_1} \tag{10}$$

where  $\mathbf{s}_i = [s_{1i}, s_{2i}, \dots, s_{N_1i}]^T$ ,  $i = 1, 2, \dots, N_2$ , is the  $i$ -th column of  $\mathbf{S}$  and  $\mathbf{a}_k = [a_{1k}, a_{2k}, \dots, a_{N_1k}]^T$ ,  $k = 1, 2, \dots, N_1$ , is the  $k$ -th column of  $\mathbf{A}$ . As shown in the previous Section,  $\mathbf{s}_i$  has a sparse distribution, so that most of its elements are negligible (e.g., see Fig. 2). It follows that there even exist indexes  $j$  for which



**Fig. 3.** ICA bases obtained for the “Lena” image, that has been divided into  $8 \times 8$  patches



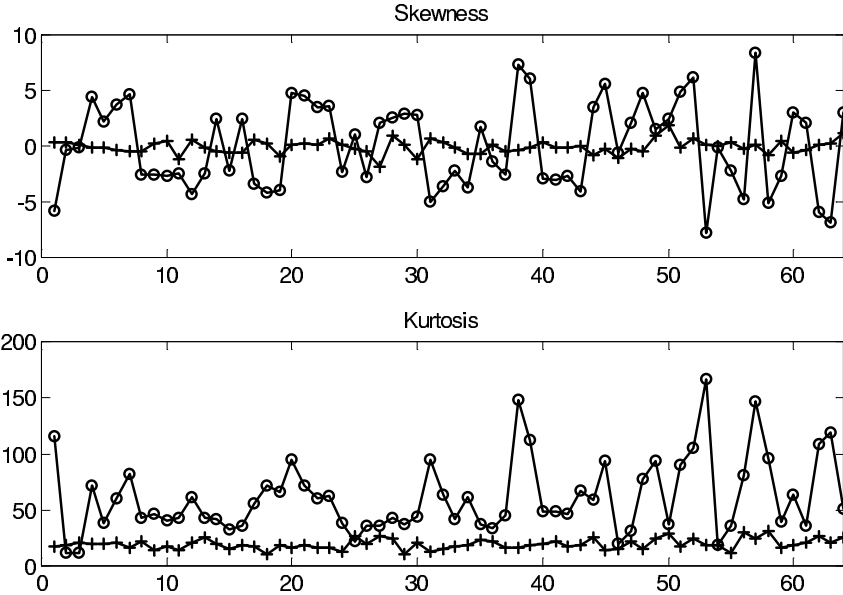
**Fig. 4.** Patches of the image that are similar to the ICA bases

$$\mathbf{x}_i = \mathbf{A} \mathbf{s}_i \cong s_{ji} \mathbf{a}_j \quad (11)$$

It means that most of the ICA bases are, except for a scale factor, like patches of the original image. In practice, as illustrated in Fig. 3, the ICA bases are *edges* that already exist in the image (e.g., the locations of the edges shown in Fig. 3 are highlighted in Fig. 4). This is as expected, since, as discussed in the previous Section, the non-negligible elements  $s_{ji}$  of matrix  $\mathbf{S}$  correspond to edges.

To illustrate that when the ICA bases are edges (3) is maximum, let us do the following experiment. We build two matrices of bases,  $\mathbf{A1}$ , obtained from

edges of the “Lena” image, and **A2**, obtained from patches taken randomly from the same image. We calculate the matrices **S1** and **S2** using **A1** and **A2**, respectively, and obtain both the skewness and the kurtosis of these two groups



**Fig. 5.** Skewness and kurtosis of the independent components  $\mathbf{y}$  when bases are like edges ( $\circ$ ) and when bases are like any other pattern ( $+$ )

of independent components. Skewness is a particular case of  $J_p(\mathbf{y})$  when  $p = 3$  and kurtosis is very similar to  $J_4(\mathbf{y})$  (see (3)). In Fig. 5 are shown the results obtained. Both the skewness and the kurtosis are greater when the ICA bases are edges than when they are any other kind of pattern.

## 4 Conclusions

In digital image processing, the connection between ICA and sparse coding has been raised a great interest. Many works have dealt with this matter, but no mathematical study has been done so far. In this paper we have analyzed this connection and provided a mathematical proof that is supported by experimental results.

The connection between ICA and sparse coding can be applied in many fields of digital image processing, like texture segmentation, edge detection [6], digital watermarking [7] or elimination of noise [9]. These results can be also useful in modelling of biomedical systems, like the human visual system [2].

## References

1. Barlow, H.: Sensory Communication. Possible Principles Underlying the Transformation of Sensory Messages. MIT Press (1961) 217–234
2. Bell, A., Sejnowski, T.: The “Independent Components” of Natural Scenes are Edge Filters. *Vision Research*, vol. 37 (23) (1997) 3327–3338
3. Cao, X., Liu, R.: General Approach to Blind Source Separation. *IEEE Transactions on Signal Processing*, vol. 44 (3) (1996) 562–571
4. Cichocki, A., Amari, S. I.: Adaptive Blind Signal and Image Processing. John Wiley & Sons (2002)
5. Field, F.: What is the Goal of Sensory Coding? *Neural Computation*, vol. 6 (1994) 559–601
6. Hornillo-Mellado, S., Martín-Clemente, R., Puntonet, C. G., Acha, J. I.: Application of Independent Component Analysis to Edge Detection. *Proc. World Automation Congress (WAC 2004)*. Sevilla, Spain (2004)
7. Hornillo-Mellado, S., Martín-Clemente, R., Acha, J. I., Puntonet, C. G.: Application of Independent Component Analysis to Edge Detection and Watermarking. *Lecture Notes in Computer Science*, 2687 (2003) 273–280
8. Hyvärinen, A., Hoyer, P. O., Hurri, J.: Extensions of ICA as Models of Natural Images and Visual Processing. *Proc. International Symposium on Independent Component Analysis and Blind Signal Separation (ICA 2003)*. Nara, Japan (2003) 963–974
9. Hyvärinen, A., Karhunen, J., Oja, E.: Independent Component Analysis. Wiley-Interscience, John Wiley & Sons (2001)
10. Hyvärinen, A., Oja, E.: A Fast Fixed-Point Algorithm for Independent Component Analysis. *Neural Computation*, vol.6 (1997) 1484–1492
11. Olshausen, B. A., Field, D. J.: Sparse Coding with an Overcomplete Basis Set: A Strategy Employed by V1? *Vision Research*, vol. 37 (23) (1997) 3311–3325

## Appendix: Proof of Eqn. (5)

The derivative of (4) with respect to  $k$ -th element of  $\mathbf{b}$ ,  $b_k$ , is given by:

$$\frac{\partial}{\partial b_k} L(\mathbf{y}, \lambda) = \frac{\partial}{\partial b_k} J_p(\mathbf{y}) - \lambda \frac{\partial}{\partial b_k} \left( \sum_{j=1}^{N_2} y_j^2 - N_2 \right) \quad (12)$$

The first term is given by:

$$\begin{aligned} \frac{\partial}{\partial b_k} J_p(\mathbf{y}) &= \frac{1}{N_2} \frac{\partial}{\partial b_k} \sum_{j=1}^{N_2} y_j^p = \frac{1}{N_2} \frac{\partial}{\partial b_k} \sum_{j=1}^{N_2} (\mathbf{b}^T \mathbf{x}_j)^p = \frac{p}{N_2} \sum_{j=1}^{N_2} y_j^{p-1} x_{kj} \quad (13) \\ &= \frac{p}{N_2} [x_{k1}, x_{k2}, \dots, x_{kN_1}] \mathbf{y}^{p-1} \end{aligned}$$

since  $y_j = \mathbf{b}^T \mathbf{x}_j = \sum_{j=1}^{N_1} b_k x_{kj}$ , where  $x_{kj}$  is the entry  $(k, j)$  of matrix  $\mathbf{X}$ , and

$$\mathbf{y}^k \stackrel{def}{=} \begin{bmatrix} y_1^k \\ y_2^k \\ \dots \\ y_{N_2}^k \end{bmatrix} \quad (14)$$



Secondly,

$$\lambda \frac{\partial}{\partial b_k} \left( \sum_{j=1}^{N_2} y_j^2 - N \right) = 2\lambda \sum_{j=1}^{N_2} y_j \frac{\partial}{\partial b_k} y_j = 2\lambda \sum_{j=1}^{N_2} y_j x_{kj} \tag{15}$$

In matrix form, we have:

$$\frac{\partial}{\partial \mathbf{b}} L(\mathbf{y}, \lambda) = \frac{1}{N_2} (p\mathbf{X}\mathbf{y}^{p-1} - 2\lambda\mathbf{X}\mathbf{y}^1) \tag{16}$$

where  $\frac{\partial}{\partial \mathbf{b}} L(\mathbf{y}, \lambda)$  is a  $N_1 \times 1$  vector and  $\frac{\partial}{\partial b_k} L(\mathbf{y}, \lambda)$  is its  $k$ -th component.

On the other hand:

$$\frac{\partial}{\partial \lambda} L(\mathbf{y}, \lambda) = \sum_{j=1}^{N_2} y_j^2 - N_2 = \|\mathbf{y}^1\|_2^2 - N_2 \tag{17}$$

# Analysis of Variance of Three Contrast Functions in a Genetic Algorithm for Non-linear Blind Source Separation

F. Rojas<sup>1</sup>, J.M. Górriz<sup>2</sup>, and O. Valenzuela<sup>3</sup>

<sup>1</sup> Dpt. Arquitectura y Tecnología de Computadores, E.T.S.I. Informática,  
University of Granada, 18071, Spain

frojas@atc.ugr.es, <http://atc.ugr.es>

<sup>2</sup> Dpt. Teoría de la Señal, Telemática y Comunicaciones, University of Granada

<sup>3</sup> Dpt. Matemática Aplicada, University of Granada

**Abstract.** The task of recovering a set of unknown sources from another set of mixtures directly observable and little more information about the way they were mixed is called the blind source separation problem. If the assumption in order to obtain the original sources is their statistical independence, then ICA (Independent Component Analysis) may be the technique to recover the signals. In this contribution, we propose and analyze three evaluation functions (contrast functions in Independent Component Analysis terminology) for the use in a genetic algorithm (PNL-GABSS, Post-NonLinear Genetic Algorithm for Blind Source Separation) which solves source separation in nonlinear mixtures, assuming the post-nonlinear mixture model. A thorough analysis of the performance of the chosen contrast functions is made by means of ANOVA (Analysis of Variance), showing the validity of the three approaches.

## 1 Introduction

Blind Source Separation refers to the problem of recovering a set of non-observable source signals from another set of related and observable “mixtures”, without having any *a priori* knowledge neither about the sources, nor the way they were mixed. In many real world situations, one or more desired signals need to be recovered from the mixtures only. When the assumption in order to obtain the original sources is their statistical independence, then ICA (*Independent Component Analysis*) can be applied. ICA is a method for finding underlying factors or components from multidimensional or multivariate statistical data [1].

This technique deals with the problem of transforming a set of observation patterns  $\mathbf{x}$ , whose components are not statistically independent from one another, into a set of patterns  $\mathbf{y} = F(\mathbf{x})$  whose components are statistically independent from each other. In linear ICA, which is the most extensively studied case, the transformation  $F$  is restricted to being linear. Nonlinear ICA allows  $F$  to be nonlinear.

The nonlinear separation of sources has been addressed in [2], [3] and [4]. However, nonlinear ICA, is rather unconstrained, and normally demands additional information to make the estimations coincide with the estimations. Applying the post-nonlinear constraint, other authors (Taleb and Jutten [5], Rojas et al. [6], [7]) proposed several contrast functions which approximate the mutual information of the estimated components.

A fundamental difficulty in nonlinear ICA is that is highly non-unique without some extra constraints. Therefore finding independent components does not lead us necessarily to the original sources [4]. Blind source separation in the nonlinear case is, in general, impossible. Taleb and Jutten [5], added some extra constraints to the nonlinear mixture so that the nonlinearities are independently applied in each channel after a linear mixture. In this way, the indeterminacies are the same as for the basic linear instantaneous mixing model: invertible scalings and permutations.

The mixture model can be described by the following equation:

$$\mathbf{x}(\mathbf{t}) = \mathbf{f}(\mathbf{A} \cdot \mathbf{s}(\mathbf{t})). \quad (1)$$

The unmixing stage, which will be performed by the algorithm here proposed is expressed by equation 2:

$$\mathbf{y}(\mathbf{t}) = \mathbf{W} \cdot \mathbf{g}(\mathbf{x}(\mathbf{t})). \quad (2)$$

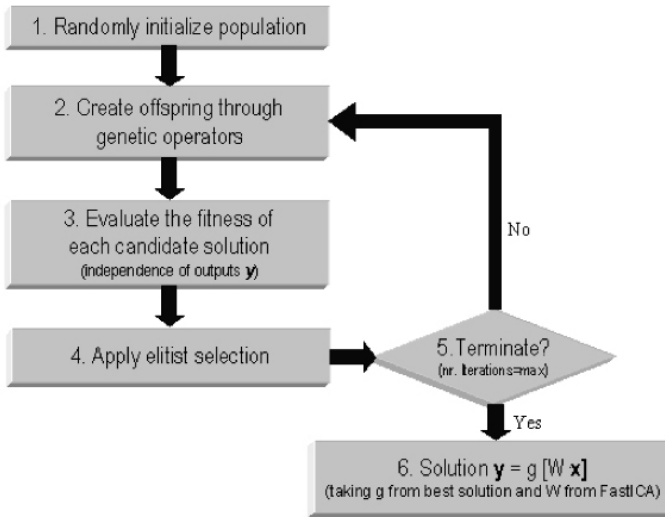
In this contribution, we propose the use of a genetic algorithm to solve post-nonlinear blind source separation. We also present three different evaluation functions that asses each candidate solution. Neural network approaches have the drawback of possibly being trapped into near-optimal solutions in situations where the search space presents many local minima. As an alternative, genetic algorithms deal simultaneously with multiple solutions, not a single solution, and also include random elements, which help to avoid getting trapped into sub-optimal solutions.

## 2 Nonlinear Blind Source Separation Using a Genetic Algorithm

The proposed algorithm will be based on the estimation of mutual information, value which cancels out when the signals involved are independent. Mutual information between the elements of a multidimensional variable  $\mathbf{y}$  is defined as:

$$I(y_1, y_2, \dots, y_n) = \sum_{i=1}^n H(y_i) - H(y_1, y_2, \dots, y_n). \quad (3)$$

In order to exactly compute mutual information, we need also to calculate entropies, which likewise require to know the analytical expression of the probability density function (PDF) which is generally not available in practical applications of speech processing. Thus, we propose several evaluation functions (or



**Fig. 1.** Flow chart of the genetic algorithm for source separation of post-nonlinear mixtures

contrast function in BSS terminology) that approximate mutual information. A *contrast function*,  $\Psi(\cdot)$ , is any non-linear function which is invariant to permutation and scaling matrices, and attains its minimum value in correspondence of the mutual independence among the output components.

Independently of the selected contrast function, the operation of the basic genetic algorithm is invariant, needing the following features to be completely characterized:

1. **Encoding Scheme.** The genes will represent the coefficients of the odd polynomials which approximate the family of nonlinearities  $g$  (see equation 2). The linear matrix will be approximated by a well-known method such as JADE [8].
2. **Initialization Procedure.** Both polynomial and matrix coefficients which form part of the chromosome are randomly initialized.
3. **Fitness Function.** The key point in the performance of a GA is the definition of the fitness function. In this case, the fitness function that we want maximize will be precisely the inverse of the approximation of mutual information given in equation 3:

$$Fitness(y) = \frac{1}{I(y)} = \frac{1}{\sum_{i=1}^p H(y_i) - H(y_1, y_2, \dots, y_p)} \quad (4)$$

4. **Genetic Operators.** Typical crossover and mutation operators will be used for the manipulation of the current population in each iteration of the GA. The crossover operator is “Simple One-point Crossover”. The mutation operator

(“Non-Uniform Mutation” [9]), is more favorable to exploration in the early stages of the algorithm, while exploitation takes more importance when the solution given by the GA is closer to the optimal.

5. Parameter Set. Population size, number of generations, probability of mutation and crossover and other parameters relative to the genetic algorithm operation were chosen depending on the characteristics of the mixing problem.

The flow chart for the genetic algorithm that solves blind source separation in post-nonlinear mixtures is shown in Figure 1.

### 3 Contrast Functions Proposed

In order to calculate the degree of independence of a set of variables precisely it is necessary to have the mathematical expression of their probability density functions. Actually, since that information is not typically available, is usual to utilize approaches of independence measures. In this research work, several functions have been evaluated, most of them based on the calculation of approaches of the mutual information or measures derived from it. One of the great advantages of genetic algorithms is its flexibility in the use of evaluation functions.

#### 3.1 Cross-Cumulants Minimization

It is possible to build a contrast function based on the simultaneous minimization of several cumulants, cross-cumulants 2-2, 3-1 and 1-3 to be precise. We can also add some penalization factors to the contrast functions, promoting solutions whose estimations are closed to zero mean and unit variance ( $C_1$  and  $C_2$ , respectively).

$$\begin{aligned} \Psi_{CrossCumulants}(y_i, y_j) = & cum_{22}(y_i, y_j) + \\ & + cum_{31}(y_i, y_j) + cum_{13}(y_i, y_j) + \alpha C_1 + \beta C_2 \end{aligned} \quad (5)$$

where  $\alpha$  and  $\beta$  are weights (real numbers) for the penalization functions.

Unfortunately, this simple approach is only valid for separating two signals. Although it can be extended to three or more computing the contrast function by pairs, the computational cost would be exponentially increasing with the number of mixtures.

#### 3.2 PDF Direct Approximation Using Histograms

We propose to approximate densities through the discretization of the estimated signals building histograms and then calculate their joint and marginal entropies. In this way, we define a number of bins  $m$  that covers the selected estimation space and then we calculate how many points of the signal fall in each of the bins ( $B_i$   $i = 1, \dots, m$ ). Finally, we easily approximate marginal entropies using the following formula:

$$H(y) = - \sum_{i=1}^n p(y_i) \log_2 p(y_i) \approx - \sum_{j=1}^m \frac{\text{Card}(B_j(y))}{n} \log_2 \frac{\text{Card}(B_j(y))}{n} \quad (6)$$

where  $\text{Card}(B)$  denotes cardinality of set  $B$ ,  $n$  is the number of points of estimation  $y$ , and  $B_j$  is the set of points which fall in the  $j^{\text{th}}$  bin.

The same method can be applied for computing the joint entropies of all the estimated signals:

$$\begin{aligned} H(y_1, \dots, y_p) &= \sum_{i=1}^p H(y_i | y_{i-1}, \dots, y_1) \approx \\ &\approx - \sum_{i_1=1}^m \sum_{i_2=1}^m \dots \sum_{i_n=1}^m \frac{\#B_{i_1 i_2 \dots i_p}(y)}{n} \log_2 \frac{\#B_{i_1 i_2 \dots i_p}(y)}{n}. \end{aligned} \quad (7)$$

where  $p$  is the number of components which need to be approximated.

Therefore, substituting entropies in equation 3 by approximations of equation 6 and 6, we obtain a contrast function which will reach its minimum value when the estimations are independent.

### 3.3 Gram-Charlier Expansion for Mutual Information Approximation

A widely applied method for mutual information approximation is the application of the Gram-Charlier expansion, which only needs some moments of  $y_i$  as suggested by Amari et al.[10] to express each marginal entropy of  $y$  as:

$$H(y_i) \approx \frac{\log(2\pi e)}{2} - \frac{(k_3^i)^2}{2 \cdot 3!} - \frac{(k_4^i)^2}{2 \cdot 4!} + \frac{3}{8} (k_3^i)^2 k_4^i + \frac{1}{16} (k_4^i)^3 \quad (8)$$

where  $k_3^i = m_3^i$ , and  $k_4^i = m_4^i - 3$ .

Substituting equation 8 in the calculation of mutual information (equation 3), we obtain a new contrast function:

$$\begin{aligned} \text{eval}_{IM-GramCharlier}(\mathbf{g})^{-1} &= \Psi_{IM-GramCharlier}(\mathbf{y}) \\ &= \sum_{i=1}^n \left[ \frac{\log(2\pi e)}{2} - \frac{(k_3^i)^2}{2 \cdot 3!} - \frac{(k_4^i)^2}{2 \cdot 4!} + \frac{3}{8} (k_3^i)^2 k_4^i + \frac{1}{16} (k_4^i)^3 \right] - \\ &\quad - \log |\det(\mathbf{W})| - \sum_{i=1}^n E \left[ \log \left| \sum_{k=1}^P (2k-1) p_{ik} x_i^{2k-2} \right| \right] \end{aligned} \quad (9)$$

The approximation of entropy in equation 8 is only valid for uncorrelated random variables, being necessary to preprocess the mixed signals (*prewhitening*) before estimating their mutual information. Whitening or sphering of a mixture of signals consists of filtering the signals so that their covariances are zero (uncorrelatedness), their means are zero, and their variances equal unity.

## 4 ANOVA of Simulations

Analysis of Variance (ANOVA) is a statistical method that yields values that can be tested to determine whether a significant relation exists between variables of interest. This method is widely used in industry to help identify the source of potential problems in the production process, and identify whether variation in measured output values is due to variability between various manufacturing processes, or within them.

In our case, we will use ANOVA in order to analyze the estimations in terms of their similarity to the original sources and testing which variables (factors) affect more to the performance of the proposed algorithm. It will be of special interest the influence of the contrast function chosen, so that we can determine which one works better in practice. Table 1 shows the factors defined for the analysis of variance.

**Table 1.** Chosen factors for the ANOVA

| Factor                | Levels of the chosen factors |                     |                         |         |
|-----------------------|------------------------------|---------------------|-------------------------|---------|
|                       | Level 1                      | Level 2             | Level 3                 | Level 4 |
| Contrast Function     | $\Psi_{IM-GramCharlier}$     | $\Psi_{Histograms}$ | $\Psi_{CrossCumulants}$ |         |
| Nr. Samples           | 2000                         | 4000                | 8000                    |         |
| Population Size       | 10                           | 20                  | 30                      | 40      |
| Generations number    | 10                           | 20                  | 40                      | 80      |
| Crossover probability | 0.05                         | 0.2                 | 0.5                     |         |
| Mutation probability  | 0.01                         | 0.1                 | 0.3                     |         |
| Selection probability | 0.01                         | 0.1                 | 0.2                     |         |

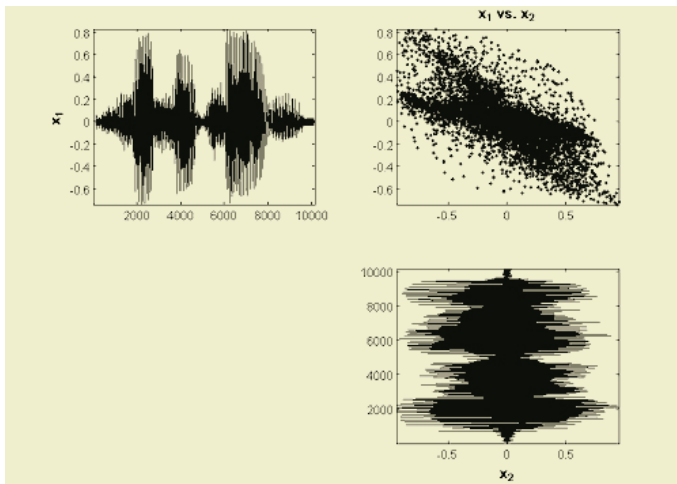
The simulation applies the following linear and non-linear transformation to a set of two voice signals of 10.165 samples each one (see mixtures in figure 2):

$$\mathbf{A} = \begin{bmatrix} 0.4891 & -0.1202 \\ -0.4641 & 0.8668 \end{bmatrix}, \mathbf{F} = \begin{bmatrix} \tanh(x) \\ \tanh(x/2) \end{bmatrix} \quad (10)$$

In this case, we performed an ANOVA with the output variable as the average crosstalk between estimations and the sources and another ANOVA for the computation time. Table 2 briefly shows the different results obtained depending on the contrast function chosen for the genetic algorithm.

From the results, it can be drawn that the contrast function based on the Gram-Charlier expansion for mutual information approximation achieves better results. Also, from the ANOVA is deduced that results also improve if the number of samples increases, as it is normally expected. Other factors, as the crossover or mutation probabilities do not have a strong influence on the crosstalk results.

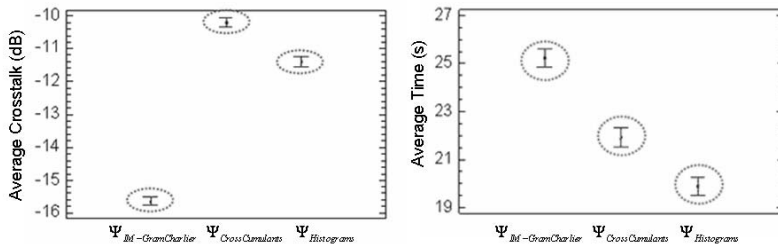
ANOVA was also applied to assess how each of the different levels of the factors in Table 2 affect the computation time. As can be expected, all the



**Fig. 2.** Representation of the post-nonlinear mixtures along time and joint distribution

**Table 2.** Multiple range table for the factor “Contrast Function” in the ANOVA

| Contrast Func.           | Nr. of runs | Avg. Crosstalk | Homogenous Group |
|--------------------------|-------------|----------------|------------------|
| $\Psi_{Histograms}$      | 1296        | -11.399 dB     | B                |
| $\Psi_{CrossCumulants}$  | 1296        | -10.201 dB     | C                |
| $\Psi_{IM-GramCharlier}$ | 1296        | -15.638 dB     | A                |



**Fig. 3.** Homogenous groups for each contrast function for the average crosstalk (left) and average computation time (right)

factors have a strong statistical significance over the algorithm response time (e.g. as population size increases, computation time also increases).

Regarding the effect of the chosen contrast function over the computation time, algorithm using  $\Psi_{IM-GramCharlier}$  takes an average time of 25.228s, while using  $\Psi_{CrossCumulants}$  and  $\Psi_{Histograms}$  takes an average of 21.92s and 19.88s, respectively. Therefore, contrast function  $\Psi_{IM-GramCharlier}$ , although is the one which gives the best results in terms of crosstalk, is also the most complex regarding time of computation (see Figure 3).



## 5 Conclusion

We have presented three contrast functions to evaluate candidate solutions in a proposed genetic algorithm for post-nonlinear blind source separation. A thorough analysis such as ANOVA was applied in order to assess the validity and efficiency of the approaches, showing that the function based on the mutual information approximation by means of the Gram-Charlier expansion achieves the best similarity results between the estimations and the sources.

## References

1. Hyvärinen, A., Karhunen, J., Oja, E.: Independent Component Analysis. John Wiley, New York (2001)
2. Burel, G.: Blind separation of sources: a nonlinear neural algorithm. *Neural Networks* **5** (1992) 937–947
3. Lee, T.W., Koehler, B., Orglmeister, R.: Blind source separation of nonlinear mixing models. In: *Neural Networks for Signal Processing VII*, IEEE Press (1997) 406–415
4. Hyvärinen, A., Pajunen, P.: Nonlinear independent component analysis: Existence and uniqueness results. *Neural Networks* **12** (1999) 429–439
5. Taleb, A., Jutten, C.: Source separation in post-nonlinear mixtures. *IEEE Trans. on Signal Processing* **47** (1999) 2807–2820
6. Rojas, I., Puntonet, C., Cañas, A., Pino, B., Fernández, J., Rojas, F.: Genetic algorithms for the blind separation of sources, *IASTED* (2001)
7. Rojas, F., Puntonet, C., Rodríguez, M., Rojas, I., Clemente, R.: Blind source separation in post-nonlinear mixtures using competitive learning, simulated annealing and a genetic algorithm. *IEEE Transactions on Systems, Man and Cybernetics, Part C* **34** (2004) 407 – 416
8. Cardoso, J.F., Souloumiac, A.: Blind beamforming for non Gaussian signals. *IEE Proceedings-F* **140** (1993) 362–370
9. Michalewicz, Z.: *Genetic Algorithms + Data Structures = Evolution Programs*. Tercera edn. Springer-Verlag, New York (1999)
10. Amari, S.I., Cichocki, A., Yang, H.: A new learning algorithm for blind source separation. In: *Advances in Neural Information Processing Systems 8*. MIT Press (1996) 757–763

# Robust Processing of Microarray Data by Independent Component Analysis

Francisco Díaz<sup>1</sup>, Raul Malutan<sup>1</sup>, Pedro Gómez<sup>1</sup>, Victoria Rodellar<sup>1</sup>,  
and Carlos G. Puntonet<sup>2</sup>

<sup>1</sup> Facultad de Informática, Universidad Politécnica de Madrid, Campus de Montegancedo,  
s/n 28660 Boadilla del Monte, Madrid, Spain  
pedro@pino.datsi.fi.upm.es

<sup>2</sup> Dept. De Arquitectura y Tecnología de Computadores, Universidad de Granada,  
Avenida Daniel Saucedo, E-18071 Granada, Spain  
carlos@atc.ugr.es

**Abstract.** Microarray Data Processing is becoming a field of important activity for Signal Processing and Pattern Recognition areas, as the extraction and mining of meaningful data from large groupings of microarray patterns is of vital importance in Medicine, Genomics, Proteomics, Pharmacology, etc. In this paper emphasis is placed on studying and cataloging the nature of possible sources of corruption of microarray data and in establishing a pre-processing methodology for discriminating sources of corruption from microarray data (denoising). We also discuss ways of precisely reconstructing original contributions (theoretically hybridized data) using ICA methods. Some classical examples are shown, and a discussion follows the presentation of results.

## 1 Introduction

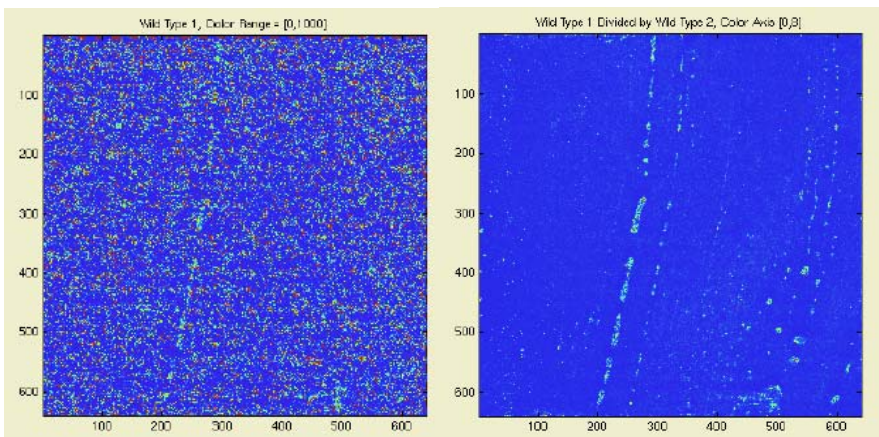
Cells react to stimuli from their environment or resulting from their internal state by the induction or repression of genes via among other factors up- or down-regulating the amounts of corresponding mRNA molecules. Different experimental conditions show different characteristic expression patterns. High-throughput genome-wide measurements of gene transcript levels have become available with the recent development of microarray technology. Intelligent and efficient mathematical and computational analysis tools are needed to read and interpret the information content buried in those large data sets.

During the last years special importance has been placed on the interpretation, classification and recognition of relationships expressed in microarray data to infer the activity of specific genes, using clustering techniques and other statistical analysis tools like principal component analysis (PCA). In most cases it was tacitly assumed that the obtainment of microarray data from genetic samples is a fully reliable process *per se*, not having to take into account the large complexity of the procedures used. Some of them concern the construction of the microarrays, the hybridization process, the scanning and the final numeric processing. In the present work the emphasis is focussed on the specific processing of microarray data to compensate or even cancel undesirable effects resulting from unreliability in the mentioned processes to increase

the robustness of microarray data generation. The problem is stated by defining possible sources of corruption and the desired, undistorted hybridization data in terms of a signal or image decomposition process, in which both the corrupting effects and the desired data combine as in an additive or convolutive model to result in the observed microarray data. Under this hypothesis, the desired microarray data are considered a realization of more or less unpredictable or randomly distributed stochastic process, while the corrupting processes, affecting the microarray, are to be seen as processes having some kind of spatial pattern far from randomness, due to the intrinsic nature of the corruption causes. Therefore some pre-processing methods, using 2D Fourier Transforms, could be used to obtain a first estimation of the set of observations causing corruption and of the theoretical microarray data from hybridization. Further on from this starting point, the problem may be configured as a typical combination of independent sources, amenable of being solved using powerful Independent Component Analysis, as FastICA, NMLE, JADE, and others [11]. In this paper emphasis will be placed in studying and cataloging the nature of the possible sources of corruption and in establishing a pre-processing methodology for discriminating sources of corruption from microarray data (de-noising). Possible routes to precisely reconstruct original contributions using ICA methods will be discussed also. Some examples will be shown, and a discussion will follow the presentation of preliminary results.

## 2 Problem Statements: Robust Microarray Pre-processing

The robustness of the estimations derived from microarray processing is an important issue of consideration. Due to distribution patterns of the sample tests on the microarrays, the molecular interference at a microscopic level, the irregularities in illumination due to field distributions and to optical interference, the cross-talk phenomena in the hybridization process etc. different experimental conditions inevitably will produce different characteristic expression patterns. This is for example seen in **Fig. 1**,



**Fig. 1.** a) Original microarray data corrupted by defective microarray preparation (taken from [8]). b) Corruption patterns after separation using multiplicative decomposition

where microarray results (a) present defects (b) which become evident after contrasting against a second experiment. These defects seem to be due to contamination during the microarray production process. A thorough discussion of possible sources of perturbations which can affect the final reliability of the data extraction process may be found in [10][2].

The results from microarrays, designed as multi-probe test of chips produced by different commercial microarray providers, can be treated as multivariate 2-D images on which Independent Component Analysis can be used instead of other less resolving data fusion algorithms. The approach proposed in the present work is based on the review of the basic decomposition model for microarray data, considering that, as illustrated in Fig. 2, several independent sources are contributing to produce the final observation.

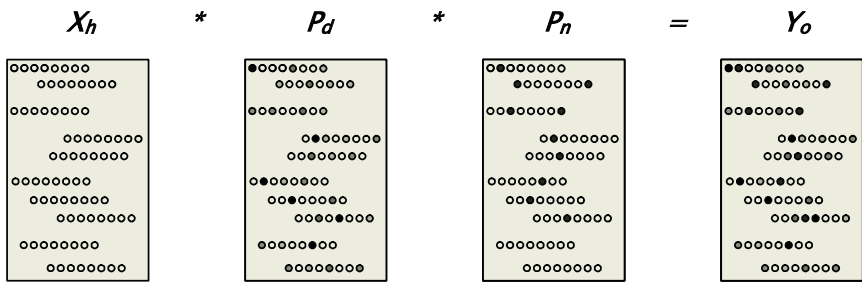


Fig. 2. Decomposition model for microarray data

The mathematical model for source decomposition would then be of a convolving or multiplicative nature. To reduce it to an additive linear model, as proposed in [5], the logarithms of microarray data should be used instead. Resulting from scanning procedures, images form matrices of positive semi-definite variables, thereby filling any zeroes with a non-zero residues:

$$Y_o^l = w_d P_d^l + w_h X_h^l + w_n P_n^l; \tag{1}$$

$$Y_o^l = \log(Y_o); \quad P_d^l = \log(P_d); \quad X_h^l = \log(X_h); \quad P_n^l = \log(P_n)$$

where  $Y_o$  represents the matrix of observations (microarray data),  $P_d$  the matrix of probe arrangements (distribution pattern),  $X_h$  represents the hybridization matrix, whose extraction would be our ultimate objective, and  $P_n$  designates the matrix of corrupting processes (noise matrix). According to the linear superposition model, the coefficients  $w_d$ ,  $w_h$ , and  $w_n$  would be gain factors or weights accounting for the contribution of the factors of decomposition to the observation process. The decomposition model would then be referred to as a multiple source independent component analysis.

Such a hypothesis may be established assuming that the contributions of perturbation noise due to array manufacture and hybridization show low wave number

spectral contents corresponding to structures of large sizes or wave lengths. Keeping this hypothesis in mind, the perturbations, produced by probe arrangement, crosstalk, reactive distributions etc., which affect sizable parts of the array, will be reduced to low wave number contents of the spatial spectrum of the image. This way the problem would then be split in two:

1. First of all a set of observations should be produced establishing some kind of discrimination on the information contents of the resulting data based on the spatial distribution of the corrupting processes.
2. The second part of the problem would be reduced to a classical ICA formulation: given two sets of observations obtained from the microarray data using wave number information by low pass-band filtering, find the independent sources in the microarray data, corresponding to low and high wave number contents.

The decomposition process would be carried out by means of a 2D Fourier Transform as follows:

$$\Psi_o(j, k) = \sum_j \sum_k Y_o^l(m, n) e^{-ikm \frac{2\pi}{N}} e^{-ijn \frac{2\pi}{N}} \tag{2}$$

$m, n$  designate the spatial indexes of the array,  $j, k$  being the wave number indices,  $N \times N$  the size of the observations matrix  $Y_o$  and  $i$  the square root of  $-1$ . A two-dimensional high-pass filter will be applied to this resulting matrix  $\Psi_o$  to produce the resulting observation matrix related with high wave number contents, therefore associated with small wavelength perturbations

$$\Psi_u = H_u \Psi_o \tag{3}$$

Finally the matrix  $\Psi_u$  with its high wave number contents will be transformed back to the spatial domain to produce a microarray data matrix devoid of long wave length spatial correlations

$$Y_u^l(m, n) = \frac{1}{N^2} \sum_j \sum_k \Psi_u(j, k) e^{ikm \frac{2\pi}{N}} e^{ijn \frac{2\pi}{N}} \tag{4}$$

Matrices  $Y_o$  and  $Y_u$  will constitute the sets of observations to start a process of independent component analysis with the final objective of obtaining a more robust estimation of the hybridization pattern  $P_h$ , as discussed in the following section.

### 3 Independent Component Analysis for Microarray Analysis

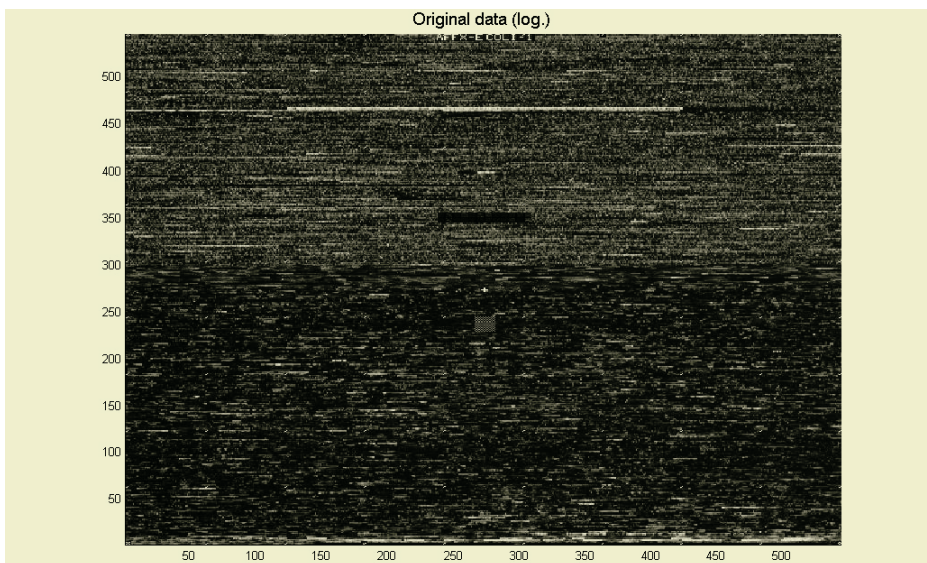
ICA, a modern statistical method, allows us to better understand data in complex and noisy environments. One exploits the fact that ICA can separate the patterns in which

we are interested from independent other effects like random sample variations or biological patterns unrelated to the subject of investigation. The technique has the potential of significantly increase the quality of the resulting data, and improve the biological validity of subsequent analysis [11].

With microarray data, ICA extracts *expression modes*  $X_h$ , the independent components (ICs), each of which represents a linear influence of a hidden cellular variable. Each retrieved IC is considered a putative biological process, which can be characterized by the functional annotations of genes that are predominant within the component. Each component thus defines functional corresponding groups of induced and repressed genes. Samples and genes can be visualized by projecting them to particular expression modes  $X_h$  or to their influences  $W$ , respectively. A projection to expression modes helps to highlight particular biological functions, to reduce noise, and to compress the data in a biologically sensible way. A recently proposed Ensemble Learning implementation of ICA [12], which identifies latent variables underlying the data, seems particularly well suited to deal with noisy microarray data sets. In particular, this technique allows one to estimate the inherent measurement noise and other factors corrupting the recorded data. In general, there is yet little published data on the use of ICA for analysis of microarray data, however.

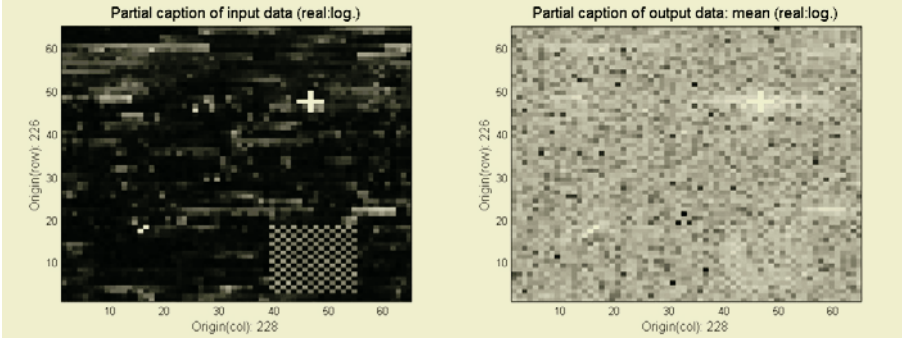
#### 4 A Case Study: Data from E. Coli

With the purpose of checking the technique proposed in the paper a sample microarray shown in Fig. 3 from the analysis of E. Coli was used [9].



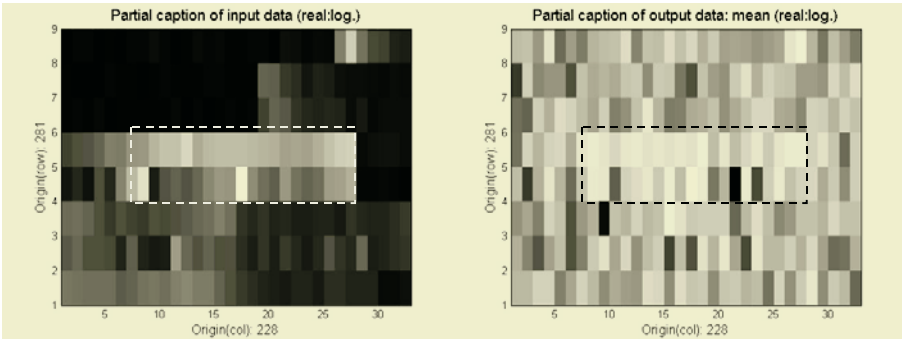
**Fig. 3.** Microarray data from genetic expression of E. Coli (taken from )

On this figure a smaller frame near its middle part (as highlighted) has been chosen to better illustrate the pre-processing steps proposed. In Fig. 4.a a close up of the selection is shown where several regular specific patters useful in the study are seen.



**Fig. 4.** a) Close up view of the E. Coli microarray section used for the study. b) Idem from the output section after preprocessing

In the upper left (files 59-60, columns 8-28) the specific profile of a probe set may be seen. Probe sets are special arrangements of probe pairs testing a specific single gene. Each probe is a short sequence of nucleotides (usually 25) with perfect match to a specific segment of the gene. Paired with the 20 perfect match probes (PM) another set of 20 probes have been included which match with the same gene segments but for a single nucleotide. This second set is called the mismatch set (MM). Both sequences show some activity regarding the specific gene tested, although it is expected that the PM set will show higher hybridization levels than the MM set. Differential estimation algorithms as MAS, MBEI or RMA [3][6][4] will evaluate the so called *expression signal* for the probe set. Another feature in the array is the cross placed at the exact



**Fig. 5.** Comparative analysis of the 20 pair probe set showing the improvements in the expression level with respect to background. a) Original features. b) Results after enhancement

centre of the microarray with maximum response. A third feature is the checkerboard pattern in the lower right (files 3-18, columns 39-54). The pre-processing technique explained in section 2 has been applied to the entire microarray, and the corresponding frame in the results have been plotted in Fig. 4.b. What can be appreciated is that specific patterns with small extensions in the spatial domain (high wave numbers) are left almost invariant (see the cross mark), while patterns showing large extension are removed (see the checkerboard pattern). The 20 pair probe has been apparently removed, but a closer inspection like in Fig. 5 reveals that this is only a subjective impression, as the pattern is still there, the relative levels of the pairs having been enhanced.

On the other hand it may be shown that their differential profiles have been improved. In a simple comparative analysis it may be seen that the average PM-MM estimate of the expression signal is neatly improved by a factor of two after pre-processing. Partial correlation shows also an improvement, although not so relevant.

**Table 1.** Expression signal estimates

| Estimation method   | Input data | Preproc. data |
|---------------------|------------|---------------|
| Average PM-MM       | 0.1092     | 0.2105        |
| Partial Correlation | 0.6304     | 0.7773        |

In any case the issue of signal estimation remains an open question far from being well settled. The *average difference algorithm* used by microarray processing software is not universally accepted, other researchers preferring *perfect matches only* adjusting the background [6][3], or using a *model-based approach* [4] as quoted in [1].

## 5 Conclusions

Through the present paper the possibility of obtaining indirect estimates of the spatial corruption patterns in microarrays using filtering in the wave number domain as a pre-processing for a subsequent Independent Component Analysis has been discussed. Results show that the working hypothesis assuming that spatial perturbations are mostly confined to the space of low wave numbers seems to be confirmed. Independent Component Analysis might prove effective in detecting corruption patterns and helping in improving expression signal estimation. This could open new possibilities for the definition of more accurate and robust methods for expression signal enhancement, which could help in the detection and classification of genetic information in cases where microarray data are corrupted, scarce or present low resolution.

## Acknowledgments

This research is being carried out under grants TIC2002-02273 and TIC2003-08756 from the *Programa Nacional de las Tecnologías de la Información y las Comunicaciones (Spain)*.



## References

- [1] Amaratunga, D. and Cabrera, J., *Exploration and Análisis of DNA Microarray and Protein Array Data*, Ed. Wiley Interscience, Hoboken, N.J., 2004, pp. 54-56.
- [2] D.H. Nguyen, A.B. Arpat, N. Wang, R.J. Carroll: DNA Microarray Experiments: Biological and Technological Aspects, *Biometrics* 58 (2002), pp. 701-717.
- [3] Irizarry, R. A., Hobbs, B., Collin, F., Beazer-Barclay, Y. D., Antonellis, K. J., Scherf, U. and Speed, T. P., "Exploration, normalization and summaries of high density oligonucleotide array probe level data", *Biostatistics*. Vol. 4, Number 2 (2003), pp. 249-264.
- [4] Li, C. and Wong, W. H., "Model-based analysis of oligonucleotide arrays: expression index computation and outlier detection", *Proc. Nat. Acad. Sci.*, Vol. 98 (2002), pp. 31-36.
- [5] M. K. Kerr: Linear Models for Microarray Data Analysis: Hidden Similarities and Differences, *Journal of Computational Biology*, Vol. 10, No. 6 (2003), pp. 891-901.
- [6] Naef, F., Lim, D. A., Patil, N., and Magnasco, M., "From features to expression: High density oligonucleotide arrays revisited". *Proc. DIMACS Workshop on Analysis of Gene Expression Data*, (2001).
- [7] P. P. Vaidyanathan: Genomics and Proteomics: A Signal Processor's Tour, *IEEE Circuits and Systems Magazine*, 4<sup>th</sup> quarter (2004), pp. 6-29.
- [8] Reference to be completed
- [9] Reference to be completed
- [10] Schena, M. and Davis, R. W. (1999). "Genes, genomes and chips", in *DNA Microarrays* (ed. M. Schena), pp. 1-16, Oxford University Press, Oxford, UK.
- [11] Su-In Lee and S. Batzoglou: Application of independent component analysis to microarrays. *Genome Biology*, Vol. 4 (2003), R76.1-R76-21.
- [12] Miskin, PhD thesis, University of Cambridge, 2000

# Multichannel Blind Signal Separation in Semiconductor-Based GAS Sensor Arrays

Guillermo Bedoya, Sergi Bermejo, and Joan Cabestany

Technical University of Catalonia, Department of Electronics Engineering,  
Campus Nord, Building C4, J. Girona 1-3, 08034 Barcelona, Spain

**Abstract.** Traditional approaches to gas sensing are usually related with gas identification and classification, i.e., recognition of aromas. In this work we propose an innovative approach to determine the concentration of the single species in a gas mixture by using nonlinear source separation techniques. Additionally, responses of tin oxide sensor arrays were analyzed using nonlinear regression techniques to determine the concentrations of ammonia and acetone in gas mixtures. The use of the source separation approach allows the compensation of some of the most important sensor disadvantages: the parameter spreading and time drift.

## 1 Introduction

Metal-oxide semiconductors have been used as gas sensors for many years. The benefits of this type of sensors include the low cost and good sensitivity against combustible (oxidizable) gases. The principle is based on the change of the conductivity. As a common material tin oxide ( $S_nO_2$ ) is used. Its electrical conductivity is caused by point defects, whereby oxygen defects work as donors. Reducing gases decrease the amount of adsorbed  $O_2$  and thereby lead to an increase in the conductivity. This conductivity effect can also result from the reaction of a reducing gas with the oxygen of the semiconductor whereby defects are formed in the material, which diffuse into the volume at higher temperatures. In order to avoid adsorption of unwanted gases (including humidity) and to accelerate the desorption of reaction products, the sensors are operated at temperatures between  $200C$  and  $500C$ . Conductivity sensors respond on different oxidizing gases like  $CO$ ,  $H_2$ ,  $CH_4$ ,  $H_2S$ ,  $NO_x$ , ..., thus they have no selectivity for a single gas.

### 1.1 Problem Statement

Traditional gas sensing systems are intended to detect and distinguish the presence of gases in the ambient atmosphere, i.e., they are designed to differentiate between the different odors in a specific environment. However, there are many applications in which is desirable to be able to predict the concentrations of single gases in a gas mixture using the same simple and cheap nonselective sensors mentioned before. Unfortunately, semiconductor gas sensors respond to the presence of reducing or oxidizing gases by nonlinear conductance changes, and, due

to the involved chemistry, they do not react to just one gas component, but with different sensitivities to different gas components. In this work, we deal with the improvement of the sensors selectivity by means of advanced signal processing techniques (nonlinear extensions of blind source separation -*BSS*- algorithms), as well as the compensation of the parameter spreading phenomena considering the absence of a parametric model for the gas sensors response. Here we propose the use of blind source separation techniques in the nonlinear context to separate single species from binary gas mixtures, by using the sensors cross sensitivities and the statistical information derived from the parameter spreading phenomena.

The rest of this paper is organized as follows: In section *II* we present a general overview on our proposed approaches to solve the single species separation from binary mixtures of gases. In section *III* a general approach to invert the sensors nonlinearity by using kernels is presented. Section *IV* shows a general overview on the extracted information from the real data base provided by the people of the sensors area at the Universitat Rovira i Virgili, as well as some experimental results. Finally, section *VI* presents some conclusions and remarks.

## 2 Proposed Approaches

Due to the complexity of the problem, there exist several approaches to solve the problem formulated in section *I*. The application of BSS techniques to separate/extract signals from mixtures of them has been widely used and studied since Jutten and Herault [1] proposed the first algorithm at the end of eighties. However, for several real applications the well known linear model fails and nonlinear extensions become necessary. In order to determine the input analyte concentrations we can use the traditional methods, e.g., non linear regression or ANN to invert the sensor's transfer function and the posterior reconstruction of gas concentrations, or we can use hybrid based methods like the proposed here.

### 2.1 Nonlinear Regression and MLP Solution

In a traditional approach, each of the  $N$  partially selective sensors, could be modelled by fitting a polynomial curve to each sensor response. From previous experimental studies [2], it has been demonstrated that the general response of a sensor  $R$  to a gas mixture was found to be:

$$\frac{R}{R_0}^{-1/\beta} = \left( \frac{1 + \sum_j K_j [G_{1j}]^{n_{1j}} [G_{2j}]^{n_{2j}}}{[O_2]} \right) \quad (1)$$

Where  $R_0$  is the response before the addition of gas,  $G_{ij}$  denotes the  $i$ th gas concentration and  $n_{ij}$  is a numerical coefficient, and  $j$  depends on the number gases. As general remarks it is necessary to appoint that the effects of one gas can be enhanced or suppressed by the presence of another gas. So, we have to find the polynomial coefficients of the experimental curve, which can be determined using a best fit solution.

Using nonlinear regression techniques or a multilayer perceptron (*MLP*), we fit the sensor curves and we found the terms in (1), the input analyte concentrations can then be determined using an iterative technique to find a solution to the simultaneous nonlinear equations that result from the curve fitting process. A simple method that sequentially checks every possible input analyte combination in the component space for a closeness of fit to the actual sensor outputs was used in this work. This basic method was implemented in order to compare results with the nonlinear BSS approach.

## 2.2 Nonlinear Extensions to BSS Algorithms

The problem of source separation has been extended to particular nonlinear models, e.g., the so-called post-nonlinear (PNL) mixtures [3] and algorithms for BSS in PNL have been proposed. In our case for nonlinear mixtures, the observations are  $e = f(As)$ , where  $f(\cdot)$  is an unknown nonlinear invertible mapping, and  $A$  is a regular unknown mixing matrix. Here, the sources  $s$  can be recovered by estimating non-linear mappings  $g_i(\cdot)$ , for compensating the  $f_i(\cdot)$ , and the demixing matrix  $B$  for compensating the mixing matrix  $A$ , so that  $y = Wg(e)$  has independent components. One can show [4] this occurs if and only if  $g_i \circ f_i$  is linear and  $WA = DP$ , provided that  $A$  has at least two non-zero entries per row or per column. Consequently, output independence leads to source separation, with the same indeterminacies as in linear memory less mixtures. A fundamental characteristic of the nonlinear BSS problem is that in the general case, solutions always exist, and they are highly non-unique. One reason for this is that if  $x$  and  $y$  are two independent random variables, any of their functions  $f(x)$  and  $g(y)$  are also independent. An even more serious problem is that in the nonlinear case,  $x$  and  $y$  can be mixed and still be statistically independent.

There have been proposed several methods to invert the nonlinear stage and then apply a BSS algorithm, some of them are oriented to solve the Post Non-linear model (where the nonlinearities are invertible and component-wise), while other methods are designed for general nonlinear mixtures. Between the PNL methods we can cite:

- Minimization of mutual information: Quadratic dependence measure for Nonlinear Blind Sources Separation, Minimization-Projection algorithms, separability of bounded sources in PNL mixtures, separation of Markovian sources in NL mixtures, adaptive splines, radial basis functions, see e.g., [5], [6], [8].
- Linearization methods: ACE and Gaussianization; where it is proposed the alternating conditional expectation (*ACE*) method of non-parametric statistics for approximate inversion of the post-nonlinear functions, see [7].

On the another hand, separation methods for general nonlinear mixtures include: Variational Bayesian methods, Kernel-based methods, see [8] and [9] and combinations of kernel *PCA* and Nonlinear Factor Analysis.

### 3 A Kernel-Based Approach

As a first approach, we propose the extension of mutual information minimization *MMI*-based algorithms to the nonlinear case. Initially, the main idea is to preprocess the input data using a nonlinear mapping, projecting data into a feature space and solve the *MMI* problem. However, direct mapping lead us to high dimensional feature spaces, so, the problem can be tractable only for binary mixtures with a few of samples. Then, we can solve this problem by using kernel functions to avoid the feature space representation or computation [9].

#### 3.1 The Linear BSS Algorithm and the Nonlinear Extension

Suppose there are  $K$  signals present at an array of  $M$  sensors,  $K < M$ . The sensor outputs are appropriately pre-processed and sampled at arbitrary time instants  $n$ , where  $n = 1, \dots, N$ . Based on the simplifying assumptions above, the array output vector  $x(n)$ , may be modelled as a function of the inputs  $x(n) = f(s(n))$ , where:

$$x(n) = f(\mathbf{A}s(n)), \quad (2)$$

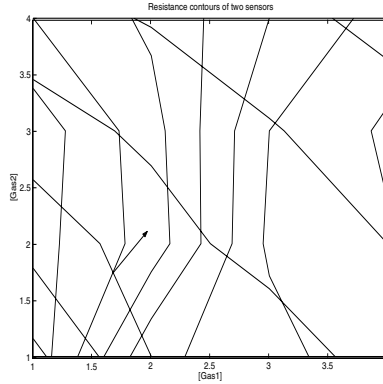
Where  $\mathbf{A}$  represents the mixing matrix (representing for example the sensors' sensitivity matrix and parameters) and  $s(n)$  are the signals present at the  $M$  sensors array. A matrix of the  $k$  sensors' sensitivity and other parameters against different measurands can be derived, then the founded matrix can be used as a linear approximation of the sensor array  $x(n) = S^T s(n)$ .

As pointed before, by using kernels, if  $f$  is an invertible (or approximately invertible) non linear function, then, it can be mapped into a feature space where the separation can be achieved, i.e., by using linear algorithms in feature space, the nonlinearity in input space can be solved.

## 4 Experimental Setup

### 4.1 Real Data from Semiconductor-Based Gas Sensor

The gas mixtures subject to inspection by the proposed scheme should satisfy the two criteria set by the BSS-model. First, the single species should be as *independent* as possible before sampling by the sensor array. Second, a mixture of two single species should follow a *linear* relationship. The last condition is not met if the mixing of gases involves a chemical reaction, thus generating new molecular structures in the gas. Figure 1 shows the orthogonality of two sensors responses by contour plots. If the sensors gradients are parallel everywhere in the space of gas concentrations, then the sensors are completely redundant: Measurements on the sensors yield identical information. However, orthogonal sensors with linearly independent gradients do provide independent information.



**Fig. 1.** Contours of constant resistance for two sensors. Given measured resistances of the sensors, the intersection of the two corresponding contours indicates the composition of the mixture

## 4.2 Single Gas Species and Binary Mixtures

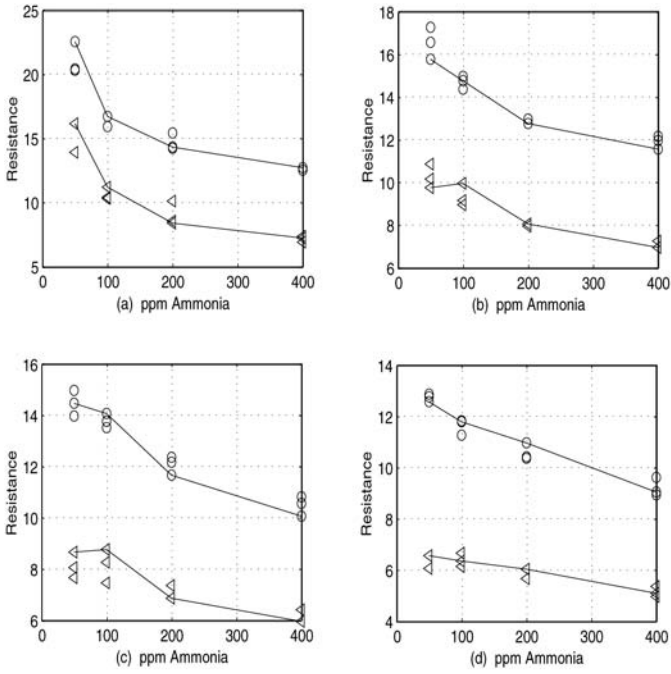
We have used an array of 15 commercial tin-oxide sensors (*Figaro Engineering*) of  $n$  different types: family *TGS800* provided by the department of electronic engineering, Universitat Rovira i Virgili (Tarragona, Spain). The output of this array was recorded in presence of two gases: Ammonia for concentrations 0, 50, 100, 200 and 400 ppm, and Acetone for concentrations 0, 50, 100, 200 and 400 ppm, see figures 2 and 3.

The experimental data set consist of 202 different files with  $17 \times 6248$  samples, each file was obtained by varying the gas concentrations. However, the data set has not a temporal structure. In order to generate a large data set that does have some temporal structure, we consider two different techniques: first we generate an unique file containing the response of each sensor in the array to different gas concentrations, emulating a continuous flow gas monitoring system. Secondly, we trained a *MLP* using the concentrations of gases as input and the array of sensors as output. Then we generate simulated mixtures corresponding to the response of the sensor array in presence of a randomly generated gas source data.

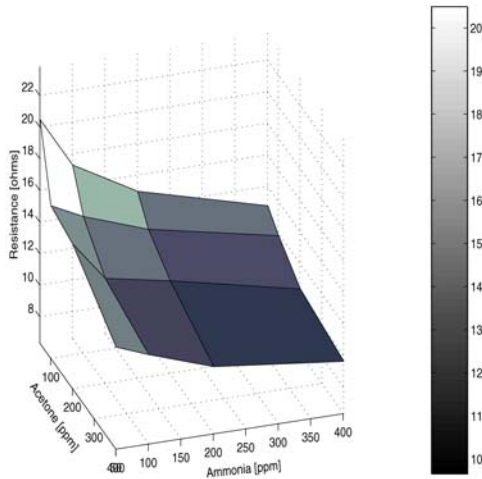
## 4.3 The Sensor Array Architecture

In order to generate the sensor array response, we used the fitted curve of each single sensor and the calculated parameters, but with each sensor in the artificial array having a different sensitivity, i.e., varying the value of the parameter  $kmix$  in (1). In this model, only one property of the sensor is changed. In real world applications, several tin oxide sensors can be doped with different amounts of *Pd* or *Pt* to change selectivity and resistance that can be measured for each sensor.

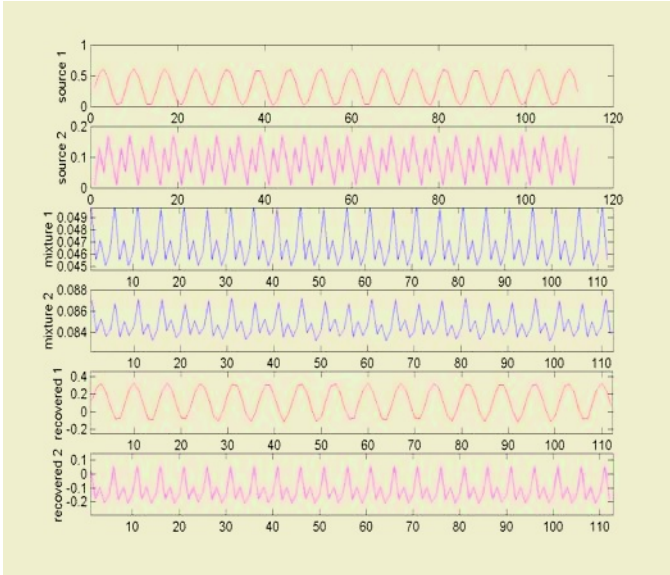
It was generated an artificial response for two gas sensors in the array. Using the nonlinear fitting and the *MLP* model results for each sensor in the array, we choose the sensor 4 response for a binary mixture (ammonia and acetone), with ammonia constant at 50ppm and 400ppm, while acetone varies: 50ppm, 100ppm,



**Fig. 2.** Responses of two sensors (circle and triangle-left) for three experimental repetitions to 50 – 400ppm Acetone at (a) 50ppm Ammonia, (b) 100ppm Ammonia, (c) 200ppm Ammonia, (d) 400ppm Ammonia



**Fig. 3.** Experimental sensor response at different gas concentrations



**Fig. 4.** Original gas source data. The nonlinear mixtures. Recovered data

200ppm and 400ppm. Using the fitted parameters according to equation (1), we generate the artificial static response considering an array of two sensors (in this case corresponding to the founded values for sensor 4). We used the trained *MLP* as well as the fitted model to generate virtual mixtures corresponding to the responses of the sensor array in the presence of the gas source data.

#### 4.4 Separation of Binary Mixtures

Using the kernel trick to preprocess the data using a nonlinear mapping, projecting it to some feature space, leads us to a partial solution. Here we are using a polynomial kernel of degree 2, but, it is supposed that using higher order polynomial kernels it can be possible to recover more than twenty "quasi" sources and then applying the BSS algorithm. We have to appoint that here we are using artificially generated sensor responses, corresponding to data with temporal structure. A new challenge for this work is the use of more realistic data sets.

## 5 Conclusions and Future Work

The goal of the present research was to verify the separation ability of the proposed approach to determine the single species concentrations in a binary mixture of gases. It is interesting to note that the separation is achieved under certain constraints, e.g., the range of species concentration and the operating temperature. Our preliminary study and some empirical results showed that the kernel used to learn the nonlinear function by mapping the data to some higher dimensional feature space can be improved with some additional computational



cost (by using high degree polynomial kernels). In our approach, each single sensor is modelled by a parameter set, then, the response is linearized before applying the linear source separation techniques. After a successful separation, the adaptive non-linear BSS algorithm can continuously learn from the environment and the demixing matrix can be adapted to the new response of the sensor array by calculating the actual sensor parameters, this allows a compensation of the parameter spreading phenomena and the time dependent drift. Experimental results confirm that single gas species identification by a sensor array can be improved using the non-linear BSS model. Additionally, we have been implementing a new approach to determine the concentration of single gas species from binary mixtures by using blind source separation of linear-quadratic mixtures, by rearranging equation (1), using a recurrent structure. Moreover, we applied nonlinear BSS techniques to ion sensitive field effect transistor (ISFET) sensor arrays, in order to quantify the ionic activities in liquid samples.

## References

1. J. Herault and C. Jutten. "Blind separation of sources, Part I: An adaptive algorithm based on neuromimetic structure," *Signal Processing*, vol. 24. (1991) – 1-10.
2. S.D. Moss, C.C. Johnson and J. Janata, "Hydrogen, Calcium and Potassium Ion-Sensitive FET Transducers: A preliminary Report". *IEEE Trans. on Biomedical engineering*, Vol. 25, No. 1. (1978) –49-56.
3. A. Taleb and C. Jutten. "Source Separation in Post-Non linear Mixtures". *IEEE Trans. on Signal Processing*, Vol. 47, No 10. (1999)–2807-2820.
4. P. Comon. "Independent component analysis, a new concept?" *Signal Processing*, vol. 36. (1994) –287-314.
5. M. Solazzi, R. Parisi, and A. Uncini, Blind source separation in nonlinear mixtures by adaptive spline neural networks, in *Proc. Int. Conf. on Independent Component Analysis and Signal Separation (ICA2001)*, San Diego, CA, USA. (2001) – 254259.
6. A. Ziehe, M. Kawanabe, S. Harmeling, and K.-R. Muller, Separation of post-nonlinear mixtures using ACE and temporal decorrelation, in *Proc. Int. Conf. on Independent Component Analysis and Signal Separation (ICA2001)*, San Diego, USA, (2001) –433438.
7. A. Ziehe, M. Kawanabe, S. Harmeling, and K.-R. Muller, Blind separation of post-nonlinear mixtures using gaussianizing transformations and temporal decorrelation, *Nara (Japan)*. (2003)
8. S. Harmeling, A. Ziehe, M. Kawanabe, and K.-R. Muller. "Kernel feature spaces and nonlinear blind source separation". In T.G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems*, volume 14. MIT Press, (2002) – 761-768.
9. Bernhard Scholkopf. "Statistical Learning And Kernel methods". Technical report MSR-TR-2000-23. Microsoft research. Gunne, Germany. (2000).

# Clustering of Signals Using Incomplete Independent Component Analysis

Ingo R. Keck<sup>1</sup>, Elmar W. Lang<sup>1</sup>, Salua Nassabay<sup>2</sup>, and Carlos G. Puntonet<sup>2</sup>

<sup>1</sup> Institute of Biophysics, Neuro- and Bioinformatics Group,  
University of Regensburg, D-93040 Regensburg, Germany  
{ingo.keck, elmar.lang}@biologie.uni-regensburg.de

<sup>2</sup> Departamento de Arquitectura y Tecnología de Computadores,  
Universidad de Granada/ESII, E-1807 Granada, Spain  
{salua, carlos}@atc.ugr.es

**Abstract.** In this paper we propose a new algorithm for the clustering of signals using incomplete independent component analysis (ICA). In the first step we apply the ICA to the dataset without dimension reduction, in the second step we reduce the dimension of the data to find clusters of independent components that are similar in their entries in the mixture matrix found by the ICA. We demonstrate that our algorithm out-performs *k-means* in the case of toy data and works well with a real world fMRI example, thus allowing a closer look the way how different parts of the brain work together.

## 1 Introduction

The problem of finding data samples that belong together in huge data spaces is an old problem in data analysis. For independent component analysis (ICA) the literature so far concentrates on the comparison of the independent components themselves. Two recent published algorithms for these problem are the tree-dependent ICA [1] and the topographic ICA [2], which have also been applied to functional magnetic resonance imaging (fMRI) data [3].

fMRI data is a grateful example to utilize spatial ICA as the functional segregation of the brain [4] closely matches the requirement of statistical spatial independence. A severe problem in the case of fMRI is the mass of data that has to be analyzed to find the interesting components as each fMRI session will typically yield hundreds of different components. However, the parts of the brain that work together during the experiment also will form clusters of activation in the time domain that can be exploited to cluster the components that represent the collaborating parts of the brain.

## 2 Theory

First we will give a short overview on independent component analysis. Then we will describe the idea behind clustering with incomplete ICA and demonstrate our algorithm.

### 2.1 Independent Component Analysis

Let  $s_1(t), \dots, s_m(t)$  be  $m$  independent signals with unit variance for simplicity, represented by a vector  $\mathbf{s}(t) = (s_1(t), \dots, s_m(t))^T$ , where  $T$  denotes the transpose. Let the mixing matrix  $\mathbf{A}$  generate  $n$  linear mixtures  $\mathbf{x}(t) = (x_1(t), \dots, x_n(t))^T$  from these source signals according to:

$$\mathbf{x}(t) = \mathbf{A}\mathbf{s}(t) \tag{1}$$

Note that each column of the mixing matrix  $\mathbf{A}$  represents the contribution of one source to each mixture.

Assume that only the mixtures  $\mathbf{x}(t)$  can be observed. Then the task to recover the original sources  $\mathbf{s}(t)$  along with the mixing matrix  $\mathbf{A}$  is commonly referred to as “independent component analysis”. For the complete case  $n = m$  many algorithms exist to tackle this problem, e.g. Infomax (based on entropy maximization [5]) and FastICA (based on negentropy using fix-point iteration [6]), just to mention some of the most popular ones. The other cases like the more difficult overcomplete ( $n < m$ ) and the more trivial undercomplete ( $n > m$ ) case have also been widely studied in the literature, see e.g. [7, 8].

### 2.2 The Influence of Dimension Reduction

In this paper we concentrate on the incomplete case: What will happen if we try to extract deliberately fewer sources than can be extracted from the mixtures  $\mathbf{x}(t)$ ? This incomplete case differs from the overcomplete case in that we do not want to extract a subset of all independent sources. Rather we try to cluster all sources into fewer components than could be extracted in principle. This is equal to a dimension reduction of the data set. It is therefore obvious that this is not possible without to sacrifice some information content. A common way to keep the loss of information as low as possible is to apply a principal component analysis to the mixtures  $\mathbf{x}(t)$  and to do the dimension reduction based on the eigenvectors  $\mathbf{e}_i$  corresponding to the smallest eigenvalues  $\lambda_i$  of the data covariance matrix  $\mathbf{C}$  [9]. This is also a convenient and often necessary preprocessing step (“whitening”) for many ICA algorithms, as it reduces the degrees of freedom in the space of the solutions by removing all second order correlations of the data and setting the variances to unity:

$$\tilde{\mathbf{x}} = \mathbf{E}\mathbf{\Lambda}^{-\frac{1}{2}}\mathbf{E}^T\mathbf{x}, \tag{2}$$

where  $\mathbf{E}$  is the orthogonal matrix of eigenvectors of the covariance matrix of  $\mathbf{x}$ , with  $\mathbf{C}(\mathbf{x}) = \mathbf{E}((\mathbf{x} - \mathbf{E}(\mathbf{x}))(\mathbf{x} - \mathbf{E}(\mathbf{x}))^T)$ , and  $\mathbf{\Lambda}$  the diagonal matrix of its eigenvalues.

It can easily be shown that this dimension reduction will cluster the independent components  $s_i(t)$  based on their presence in the mixing matrix  $\mathbf{A}$ , as the covariance matrix of  $\mathbf{x}$  depends on  $\mathbf{A}$ : [10]

$$\mathbf{E}(\mathbf{x}\mathbf{x}^T) = \mathbf{E}(\mathbf{A}\mathbf{s}\mathbf{s}^T\mathbf{A}^T) \tag{3}$$

$$= \mathbf{A}\mathbf{E}(\mathbf{s}\mathbf{s}^T)\mathbf{A}^T \tag{4}$$

$$= \mathbf{A}\mathbf{A}^T \tag{5}$$

If two columns in the mixing matrix  $\mathbf{A}$  are almost identical up to a linear factor, i.e. are linearly dependent, this means that the two sources represented by those columns are almost identically represented (up to a linear factor) in the mixtures. A matrix with two linearly dependent columns does not have full rank, hence will have at least one zero eigenvalue due to its restricted dimensionality. This also holds for the transpose  $\mathbf{A}^T$  of the matrix  $\mathbf{A}$  as the transpose has the same dimensionality as the original matrix, as well as for the product of both matrixes  $\mathbf{A}\mathbf{A}^T$ .

Setting this close-to-zero eigenvalue to zero in the course of a dimensionality reduction, will thus combine two almost identical columns of  $\mathbf{A}$  to a single one. This means that components that appear to be similar to each other in most of the mixtures will be clustered together into new components by the dimension reduction with PCA. After the dimension reduction, a standard ICA algorithm can be used to find the independent components in the dimension reduced dataset.

### 2.3 Clustering with Incomplete ICA

The idea behind the clustering with an intentionally incomplete ICA is to compare different ICA runs with a different level of dimension reduction applied beforehand. First, a complete ICA is performed extracting the maximal number of independent components from the data set. In a second run, an ICA is performed on a reduced data set which resulted from a dimension reduction during PCA preprocessing.

The independent components (IC) of the complete ICA without dimension reduction are then compared with the IC of several incomplete ICA runs. Independent components which form part of the components of the incomplete ICA are then grouped into the cluster which is represented by the IC of the incomplete ICA at hand. Hence the ICs of any incomplete ICA form sort of prototype ICs of the clusters formed by ICs from the complete set.

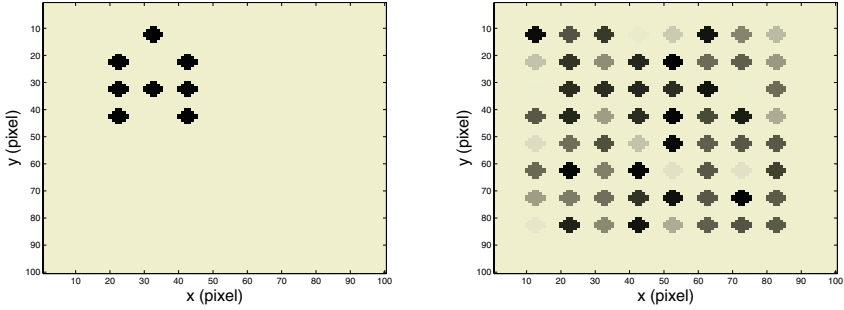
### 2.4 Algorithm

The algorithm for clustering by incomplete ICA can be described as follows:

1. apply a standard ICA to the dataset without dimension reduction: ICA1
2. apply a standard ICA to the dataset with dimension reduction: ICA2
3. find the independent components in ICA1 that are similar to some forms of components in ICA2.

## 3 Examples

First we show for toy data that clustering with incomplete ICA outperforms the standard *k-means* clustering algorithm. Then we apply the algorithm to real world data from a fMRI experiment.



**Fig. 1.** The figure on the left shows the set of circles that represent the letter “A”. The figure on the right shows one mixture of the toy data set. The A-set in the upper left corner and the C-set in the lower right corner are almost undetectable

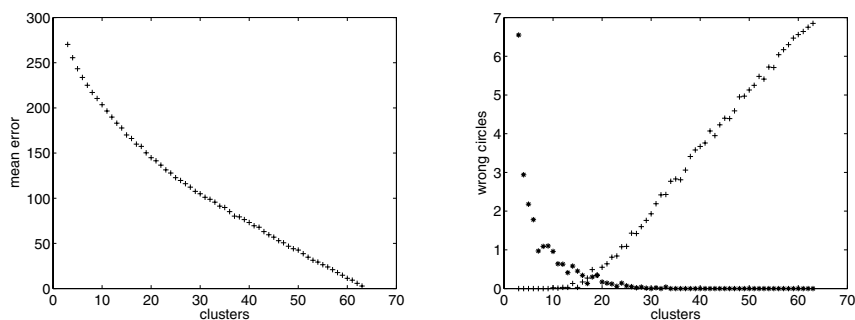
### 3.1 Toy Data

To test the quality of the clustering, we choose to create a toy data set. 64 sources were designed where each of the sources represents a circle in a square lattice of  $100 \times 100$  lattice points. The mixing matrix was initialized randomly and then modified so that two sets of circles – one representing the letter “A” in the upper left corner, the other representing the letter “C” in the lower right corner – appeared together in the mixtures by setting the columns of these sets to the same values with random differences of up to 5%. Figure 1 shows the set of “A” as well as one mixture. The results for the C-class circles were similar to the results of the A-class and are therefore omitted.

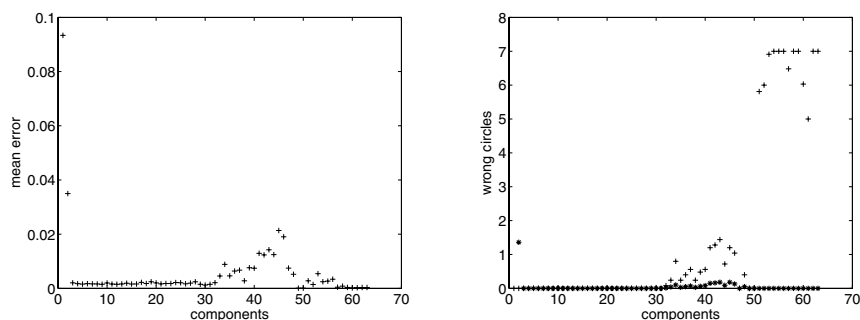
**k-Means Analysis.** A standard *k-means* clustering analysis was performed to cluster the columns of the mixing matrix **A**. Figure 2 shows on the left the mean overall error of 100 *k-means* analysis runs for 3 up to 63 clusters. It can be seen that in this case this statistic gives no hint on the number of clusters in the dataset. On the left side of the figure the mean number of wrong circles is plotted. While the *k-means* analysis obviously clusters all the A-class circles in one cluster up to an overall number of searched-for clusters of 10, it fails to do so with an average error of almost 1 not-A circle. For more than 20 clusters this error disappears, but at the same time A-class circles appear in other clusters.

**Incomplete ICA.** For this analysis the *FastICA* algorithm [6] was used. 1 up to 63 components (data reduction via a preceding PCA) were searched in 100 runs for each analysis. As the ICA first had to de-mix the mixture of circles a basic de-noising of the resulting components was necessary (every pixel with a level of 70% was counted as activated).

On the left part of figure 3 we show the mean error of the decomposition  $\langle e \rangle = \langle 1 - k_{\text{CORR}} \rangle$  against the number of components that were used for the preceding PCA, with the correlation coefficient  $k_{\text{CORR}}$  of the column of the first A-class circle in the mixing matrix **A** and the column of the estimated mixing



**Fig. 2.** The figure on the left shows the overall mean error against the number of clusters that were used for the k-means analysis. In the right figure the number of wrong circles in the A-cluster (stars) and the number of A-circles in the wrong cluster (crosses) are plotted against the number of clusters searched. It can be seen that within this data set the k-means analysis fails to cluster the right circles, independent of the number of clusters that were searched

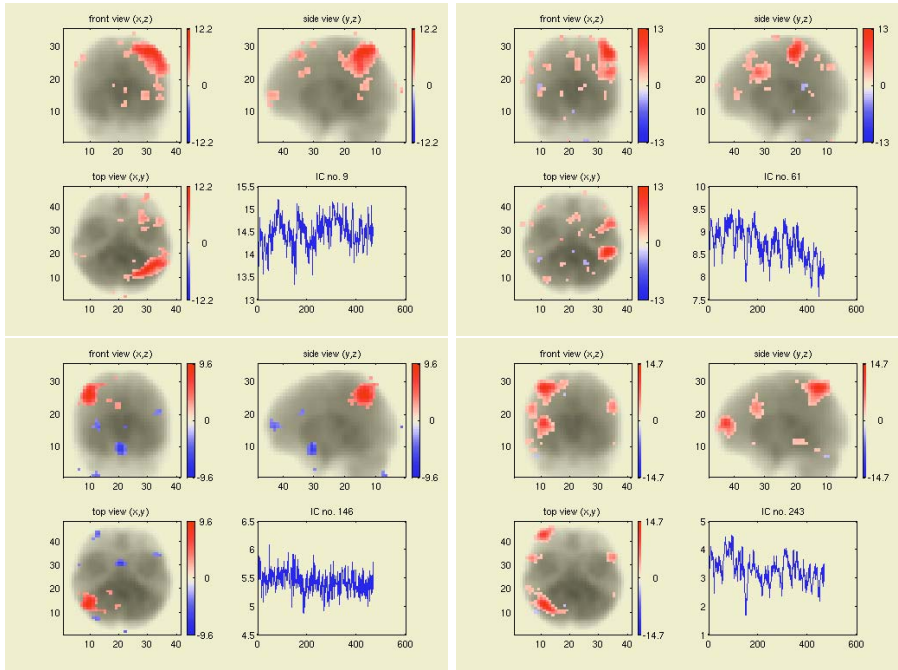


**Fig. 3.** The plot on the left shows the overall mean error against the number of PCA components that were used for the incomplete ICA. In the plot on the right side the number of wrong circles in the A-cluster (stars) and the number of A-circles in the wrong cluster (crosses) are plotted against the number of PCA components. The clustering with the incomplete ICA worked without flaw from 3 up to 31 PCA components

matrix of the A-cluster. The incomplete ICA obviously had no problem in finding the right column of the mixing matrix for the class-A circles. On the right side of figure 3 the plot for the number of falsely classified circles shows that the algorithm worked well for 3 up to 31 used components, thus being remarkably stable. For more than 52 components the ICA separated the single circles of the A- and C-Cluster, as can be expected due to the setting of the data.

### 3.2 fMRI Data

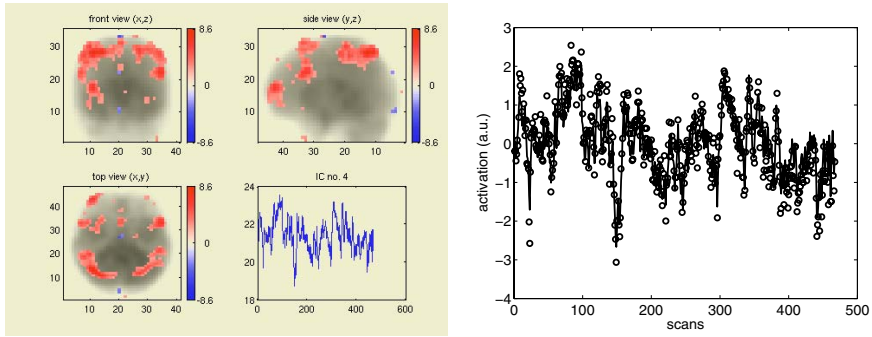
The clustering algorithm was then applied to fMRI data of a modified *Wisconsin Card Sorting Test* of one subject. This data set consisting of 467 scans



**Fig. 4.** The components that form the cluster of Figure 5, found in ICA1. The images appear flipped

was created at the institute for medicine at the Research Center Jülich, Germany, preprocessed to remove motion artefacts, normalized and filtered with a gaussian filter to increase the signal to noise ratio. Spatial ICA was used for the analysis so that the independent components correspond to activation maps and the columns of the mixing matrix correspond to the time courses of this activation. [11].

For the clustering with the incomplete ICA the data was first reduced via PCA to 400 dimensions, so that almost all information was retained. Then the (spatial) ICA1 was calculated using the *extended Infomax* algorithm. After this step the (spatial) ICA2 was calculated, with the dimensionality of the data reduced to 20. The 20 independent components of ICA2 were manually compared to a cluster of activations found in the data set using a general linear model analysis. Figure 5 (left) shows the component that was found to correspond this cluster. Then all independent components of ICA1 were searched automatically for activations in the same places. Figure 4 shows these components together with their time courses. As can be seen in this figure the time courses of the components differ in detail, while their sum closely follows the time course of the cluster component of ICA2 (Figure 5 (right)). These components and their time courses can now be analyzed further within the framework of normal brain research to understand their connectivity and collaboration.



**Fig. 5.** Left side: This cluster of activations was found in ICA2. The position of activations matches the activations found in a general linear model analysis of the data. Right side: comparison of the time course of the cluster from ICA2 (solid line) with the sum of the time courses of its components from ICA1 (circles)

## 4 Conclusion

We were able to show that with the presented algorithm the independent components of a standard ICA can successfully be clustered depending on their appearance in the mixtures. Our algorithm out-performs the *k-means* algorithm with respect to the validity of the result in the toy data example and works well for fMRI data. This can lead to new insights in the way the different parts of the brain work together as our algorithm is a computationally cheap method to cluster the interesting independent components in fMRI data analysis while at the same time minimizing the manual work of the brain researcher.

## Acknowledgment

The authors would like to thank Karsten Specht and Gereon Fink from the Research Center Jülich for the fMRI data set. This work was supported by the German ministry of education and research BMBF (project ModKog) and by the project TEC 2004-0696 (SESIBONN).

## References

1. F.R. Bach, M.I. Jordan, “Beyond independent components: Trees and Clusters”, *Journal of Machine Learning Research* **4**, 1205–1233, 2003.
2. A. Hyvärinen, P. Hoyer, “Topographic independent component analysis”, *Neural Computation* **13**, 1527–1558, 2001.
3. A. Meyer-Bäse, F.J. Theis, O. Lange, C.G. Puntonet, “Tree-Dependent and Topographic Independent Component Analysis for fMRI Analysis”, in *Proc. ICA 2004*, volume 3195 of *Lecture Notes in Computer Science*, 782–789, Granada, Spain, 2004.



4. R. S. J. Frackowiak, K. J. Friston, Ch. D. Frith, R. J. Dolan, J. C. Mazziotta, "Human Brain Function", Academic Press, San Diego, USA, 1997.
5. A.J. Bell, T.J. Sejnowski, "An information-maximisation approach to blind separation and blind deconvolution", *Neural Computation*, **7**(6), 1129–1159 (1995).
6. A. Hyvärinen, "Fast and Robust Fixed-Point Algorithms for Independent Component Analysis", *IEEE Transactions on Neural Networks* **10**(3), 626–634 (1999).
7. S. Amari, "Natural Gradient Learning for Over- and Under-Complete Bases in ICA", *Neural Computation* **11**, 1875–1883 (1999).
8. F.J. Theis, A. Jung, C.G. Puntonet, E.W. Lang, "Linear geometric ICA: Fundamentals and algorithms", *Neural Computation*, **15**, 419–439, 2003
9. A. Hyvärinen, E. Oja, "Independent Component Analysis: Algorithms and Applications", *Neural Networks*, **13**(4-5), 411–430, 2000.
10. A. Belouchrani, K. Abed-Meraim, J.-F. Cardoso, E. Moulines, "A Blind Source Separation Technique Using Second-Order Statistics", *IEEE Transactions on Signal Processing*, **45**(2), 434–444, 1997
11. M.J. McKeown, T.J. Sejnowski, "Analysis of fmri data by blind separation into independent spatial components", *Human Brain Mapping* **6**, 160–188 (1998).

# A Hybridization of Simulated Annealing and Local PCA for Automatic Component Assignment Within ICA

M. Böhm<sup>1</sup>, K. Stadlthanner<sup>1</sup>, E.W. Lang<sup>1</sup>, A.M. Tomé<sup>2</sup>, A.R. Teixeira<sup>2</sup>,  
F.J. Theis<sup>1</sup>, and C.G. Puntonet<sup>3</sup>

<sup>1</sup> Institute of Biophysics, University of Regensburg, D-93040 Regensburg, Germany  
`elmar.lang@biologie.uni-regensburg.de`

<sup>2</sup> DETUA/IEETA, Universidade de Aveiro, P-3800 Aveiro, Portugal  
`ana@ieeta.pt`

<sup>3</sup> Dept. Arquitectura y Tecnología de Computadores, Universidad de Granada,  
E-18071 Granada, Spain  
`carlos@atc.ugr.es`

**Abstract.** Independent component analysis (ICA) as well as blind source separation (BSS) often faces the problem of assigning the independent or uncorrelated components estimated with ICA or BSS techniques to underlying source signals, artifacts or noise contributions. In this work an automatic assignment tool is presented which uses *a priori* knowledge about the form of some of the signals to be extracted. The algorithm is applied to the problem of removing water artifacts from 2D NOESY NMR spectra. The algorithm uses local PCA to approximate the water artifact and defines a suitable cost function which is optimized using simulated annealing. The blind source separation of the water artifact from the remaining protein spectrum is done with the recently developed algorithm dAMUSE.

## 1 Introduction

Blind Source Separation (BSS) methods consider the separation of observed sensor signals into their underlying source signals knowing neither these source signals nor the mixing process. Considering biomedical applications, BSS methods are especially valuable to remove artifacts from the signals recorded. In many biomedical applications quite a number of independent components have to be determined with ICA algorithms and it is not *a priori* clear how many components should be assigned to the signals representing artifacts. This is especially obvious in 2D NOESY NMR proton spectra of proteins, where a prominent water artifact distorts the recorded spectra considerably. Recently artifact removal was considered using BSS techniques based on a generalized eigenvalue decomposition (GEVD) of a matrix pencil [11], [7]. Replacing the GEVD with the algorithm dAMUSE [12], [10], BSS and denoising can be achieved in one stroke. The method is very efficient and fast and outperformed FastICA and SOBI in

all cases studied [9]. But, the estimated components related with the water artifacts had to be assigned by hand. With more than 100 estimated components this turns out to become a rather tedious undertaking prone to be biased by subjective judgements of the assignment criteria.

In this work we propose a local PCA approximation to the free induction decay (FID) related with the water artifact. Hereby we explicitly use knowledge about the form of the water FID and the fact that inevitably it will always dominate the total FID of these systems. We formulate a suitable cost function to be optimized by simulated annealing [3] to determine those underlying uncorrelated components, which are estimated with dAMUSE [12] and are related with the water artifact.

The following section introduces the new algorithm AutoAssign. To illustrate the proposed method, it is applied to experimental 2D NOESY NMR spectra of aqueous solutions of several protein fragments<sup>1</sup>.

## 2 Theory

### 2.1 BSS – Model and Signal Separation with dAMUSE

Given  $N$  complex sensor signals  $x(t_{1,n}, t_{2,l}) \equiv x_n[l]$  sampled at  $L$  discrete time instances, and arranged in a data matrix  $\mathbf{X}_{N \times L}$  with  $N$  rows and  $L$  columns, where the rows of the data matrix correspond to 1D free induction decays (FIDs) of the 2D NOESY experiment taken at  $N$  discrete evolution times  $t_{1,n} \equiv [n]$ ,  $n = 1, \dots, N$ . Note that FIDs represent superpositions of exponentially decaying sinusoids with different frequencies and decay constants, called interferograms. Blind source separation (BSS) then relies on the following linear mixing model  $\mathbf{x}[l] = \mathbf{A}\mathbf{s}[l] + \boldsymbol{\epsilon}[l]$  where  $l = 0, \dots, L - 1$  and  $\mathbf{x}[l] = (x_1[l], \dots, x_N[l])^T$  designates the observed signals (interferograms) sampled at time instance  $[l]$ ,  $\mathbf{s}[l]$  the underlying uncorrelated source signals (single FIDs),  $\mathbf{A}$  the stationary mixing matrix and  $\boldsymbol{\epsilon}[l]$  an additional zero mean white Gaussian noise term which is independent of the source signals.

A generalized eigenvalue decomposition using congruent matrix pencils may be used to separate water artifacts, i.e. those FIDs originating from water proton magnetic moments, from 2D NOESY NMR spectra of proteins [8]. It provides the basis for the algorithm dAMUSE [12] used in this work. It solves the BSS problem relying only on second order GEVD techniques using congruent matrix pencils [13], [11]. The latter are formed with correlation matrices  $(\mathbf{R}_{x_1}, \mathbf{R}_{x_2})$  of zero mean sensor signals  $\mathbf{x}[l]$ , i.e. the observed FIDs. The algorithm dAMUSE extends the GEVD using congruent matrix pencils to signals embedded in a high-dimensional feature space of delayed coordinates to provide a means to perform BSS and denoising simultaneously [10]. The method uses the concept of a trajectory matrix borrowed from singular spectral analysis (SSA)[2]. Consider  $N$  sensor signal components  $x_n[l]$  comprising  $L$  samples, each row of the trajectory

<sup>1</sup> We are grateful to W. Gronwald and H. R. Kalbitzer to provide those spectra.

matrix [5] contains  $M$  delayed versions  $x_n(l + (M - m)K)$ ,  $m = 0, \dots, M - 1$ , where  $K$  denotes the delay in number of sampling intervals between consecutive rows and  $M$  gives the dimension of the embedding space. The total trajectory matrix  $\mathbf{X}^e$  of all  $N$  signals is formed by concatenating the component trajectory matrices  $\mathbf{X}_n^e$  according to:  $\mathbf{X}^e = [\mathbf{X}_1^e, \mathbf{X}_2^e, \dots, \mathbf{X}_N^e]^T$ . After embedding, the instantaneous mixing model can be written as  $\mathbf{X}^e = \mathbf{A}^e \mathbf{S}^e$  where  $\mathbf{S}^e$  represents the source signal trajectory matrix,  $\mathbf{A}^e = \mathbf{A}_n \otimes \mathbf{I}_{M \times M}$  is a block matrix and  $\mathbf{I}_{M \times M}$  denotes the identity matrix. The sensor pencil can be computed with  $\mathbf{R}_{x1} = \langle \mathbf{X}^e (\mathbf{X}^e)^H \rangle$  and  $\mathbf{R}_{x2} = \langle \mathbf{Z} \mathbf{Z}^H \rangle$  using the trajectory matrix  $\mathbf{X}^e$  and a filtered version  $\mathbf{Z} = \mathbf{X}^e \mathbf{C}^H$  with  $\mathbf{C}$  a circular convolution matrix and  $^H$  denoting the Hermitian conjugate [10].

The eigenvalues and eigenvectors of the congruent matrix pencil  $(\mathbf{R}_{x1}, \mathbf{R}_{x2})$  can be obtained via standard eigenvalue decompositions (EVD) applied in two consecutive steps:

- Compute a standard EVD of the symmetric positive definite correlation matrix  $\mathbf{R}_{x1} = \mathbf{V} \mathbf{\Lambda} \mathbf{V}^H$ , i.e, the eigenvectors ( $\mathbf{v}_i$ ) and eigenvalues ( $\lambda_i$ ) and organize the eigenvalues in descending order ( $\lambda_1 > \lambda_2 > \dots > \lambda_q \dots > \lambda_{NM}$ ). For denoising purposes, among others a variance criterion can be established to retain only the largest eigenvalues exceeding a threshold parameter  $\Theta$  [12].
- The transformation matrix can then be computed using the  $q$  largest eigenvalues and respective eigenvectors  $\mathbf{Q} = \mathbf{\Lambda}^{-\frac{1}{2}} \mathbf{V}^H$  where  $\mathbf{Q}$  is an  $q \times NM$  matrix.
- Compute the matrix  $\tilde{\mathbf{R}} = \mathbf{Q} \mathbf{R}_{x2} \mathbf{Q}^H$  and its standard eigenvalue decomposition: the eigenvector matrix  $\mathbf{U}$  and eigenvalue matrix  $\mathbf{D}_x$

The eigenvectors of the pencil  $(\mathbf{R}_{x1}, \mathbf{R}_{x2})$  form the columns of the eigenvector matrix  $\mathbf{E} = \mathbf{Q}^H \mathbf{U} = \mathbf{V} \mathbf{\Lambda}^{-\frac{1}{2}} \mathbf{U}$  which can be used to compute the output signals as described in [12].

## 2.2 The Algorithm AutoAssign

Applying the BSS algorithms above to 2D NOESY NMR spectra to separate the water artifact and other related artifacts from the protein spectra, the most tedious task is to assign the uncorrelated components estimated to the water signal. Because of erratic phase relations, up to 40 estimated components out of 128 or 256 need to be assigned to the water resonance. Hence an automated and objective assignment procedure deemed necessary.

The idea is to embed the signal in a high-dim feature space of delayed coordinates and to apply a cluster analysis to the columns of the corresponding trajectory matrix. Within each cluster a *local* PCA is then performed to obtain a low-dim approximation to the signals using only the most important principal components to approximate the signals. The latter are then fed into a suitable cost function which is optimized with simulated annealing.

Embedding and local PCA: Consider a signal  $\mathbf{x}_n[l] = (x_n[l], x_n[l + 1], \dots, x_n[l + (M - 1)])^T$  embedded in an  $M$ -dim feature space. Divide the space in  $k_c$  subspaces

$\mathcal{N}^{(k)}$  using *k-means* clustering and center the signals in each cluster locally by subtracting the cluster mean  $\bar{\mathbf{x}}_n^{(i)} = (\mathcal{N}^{(i)})^{-1} \sum_{\mathbf{x}_n[l] \in \mathcal{N}^{(i)}} \mathbf{x}_n[l]$ ,  $i = 1, \dots, k_c$ . Next a principal component analysis (PCA) is performed on each cluster separately. Then a local approximation  $\tilde{\mathbf{x}}_n[l] = \sum_{j=1}^{p(i)} \alpha_j[l] \mathbf{w}_j^{(i)} + \bar{\mathbf{x}}_n^{(i)}$  to the time domain signal is computed, using only the eigenvectors  $\mathbf{w}_j$  to the  $p$  largest eigenvalues and  $\alpha_j[l] = \mathbf{x}_n[l] \cdot \mathbf{w}_j^{(i)}$  and  $\mathbf{x}_n[l] \in \mathcal{N}^{(i)}$ . This yields the new trajectory matrix  $\tilde{\mathbf{X}}_n$ , the entries of which represent local PCA approximations of the original signals

$$\tilde{\mathbf{X}}_n = \begin{bmatrix} \tilde{x}_n[M-1] & \tilde{x}_n[M] & \dots & \tilde{x}_n[L-1] \\ \tilde{x}_n[M-2] & \tilde{x}_n[M-1] & \dots & \tilde{x}_n[L-2] \\ \tilde{x}_n[M-3] & \tilde{x}_n[M-2] & \dots & \tilde{x}_n[L-3] \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{x}_n[0] & \tilde{x}_n[1] & \dots & \tilde{x}_n[(L-1)-(M-1)] \end{bmatrix} \quad (1)$$

The final global approximation  $\langle \tilde{\mathbf{x}}_n[l] \rangle_{[l]}$ ,  $l = M-1, \dots, L-1$  is obtained by averaging all entries at the same time instance  $[l]$  which lie along diagonals.

As the water FID provides the dominant contribution to each interferogram observed, the approximation can be simplified further by retaining only the principal component to the largest eigenvalue, i.e.  $\mathbf{x}_{n,1}[l] = \alpha_1[l] \mathbf{w}_1$ . The approximation thus contains the contribution from the water signal almost exclusively.

Simulated annealing: This approximation to the FID related with the water artifact is then used to define a cost function  $\mathcal{E}(\boldsymbol{\beta}) = \sum_{l=0}^{L-1} (x_{n,\boldsymbol{\beta}}[l] - x_{n,1}[l])^2$  to be minimized with simulated annealing [3]. The BSS approximation to the water signal using the uncorrelated components estimated with the dAMUSE algorithm is obtained as  $x_{n,\boldsymbol{\beta}}[l] = \sum_j \beta_j (\mathbf{A})_{nj} s_j[l]$  where a new configuration  $\boldsymbol{\beta}$  is generated by changing any  $\beta_j$  randomly. A configuration is represented by a vector  $\boldsymbol{\beta}$  which contains as many components  $\beta_j$  as there are sources  $\mathbf{s}_j$ . To each source one element of  $\boldsymbol{\beta}$  is assigned which can take on the values  $\beta_j \in \{0, 1\}$  only. The difference in the values of the cost function for the current and the new configuration  $\Delta\mathcal{E} = \mathcal{E}(\boldsymbol{\beta}_{new}) - \mathcal{E}(\boldsymbol{\beta}_{old})$  determines the probability of acceptance of the new configuration in the simulated annealing algorithm according to

$$\frac{P[\boldsymbol{\beta}_{new}]}{P[\boldsymbol{\beta}_{old}]} = \min \left\{ 1, \exp \left( -\frac{\Delta\mathcal{E}}{k_B T} \right) \right\} \quad (2)$$

After convergence, the configuration which best fits to the local PCA approximation of the water signal is obtained. Nullifying these components deliberately, the water-artifact-free protein spectrum  $\tilde{\mathbf{x}}_n$  can be reconstructed using the remaining estimated source signals  $\tilde{\mathbf{s}}_n$  via  $\tilde{\mathbf{x}}_n = \mathbf{A} \tilde{\mathbf{s}}_n$ .

### 3 Results and Discussion

The algorithms discussed above have been applied to several experimental 2D NOESY proton NMR spectra of proteins dissolved in water. A simple pre-

saturation of the water resonance was applied to prevent saturation of the dynamic range of the analog-digital-converter (ADC). Every data set comprises 512 or 1024 FIDs  $S(t_1, t_2) \equiv x_n[l]$  or their corresponding spectra  $\hat{S}(1_1, \omega_2) \equiv \hat{x}_n[l]$ , with  $L = 2048$  samples each, which correspond to  $N = 128$  or  $N = 256$  FIDs evaluated at  $t_1 \equiv [n]$ . With each increment of the evolution period also the phase is incremented, hence only FIDs with equal phase modulations have been considered for analysis. A BSS analysis, using the algorithm dAMUSE [12] in combination with AutoAssign, was applied to all data sets.

### 3.1 Experimental Spectra

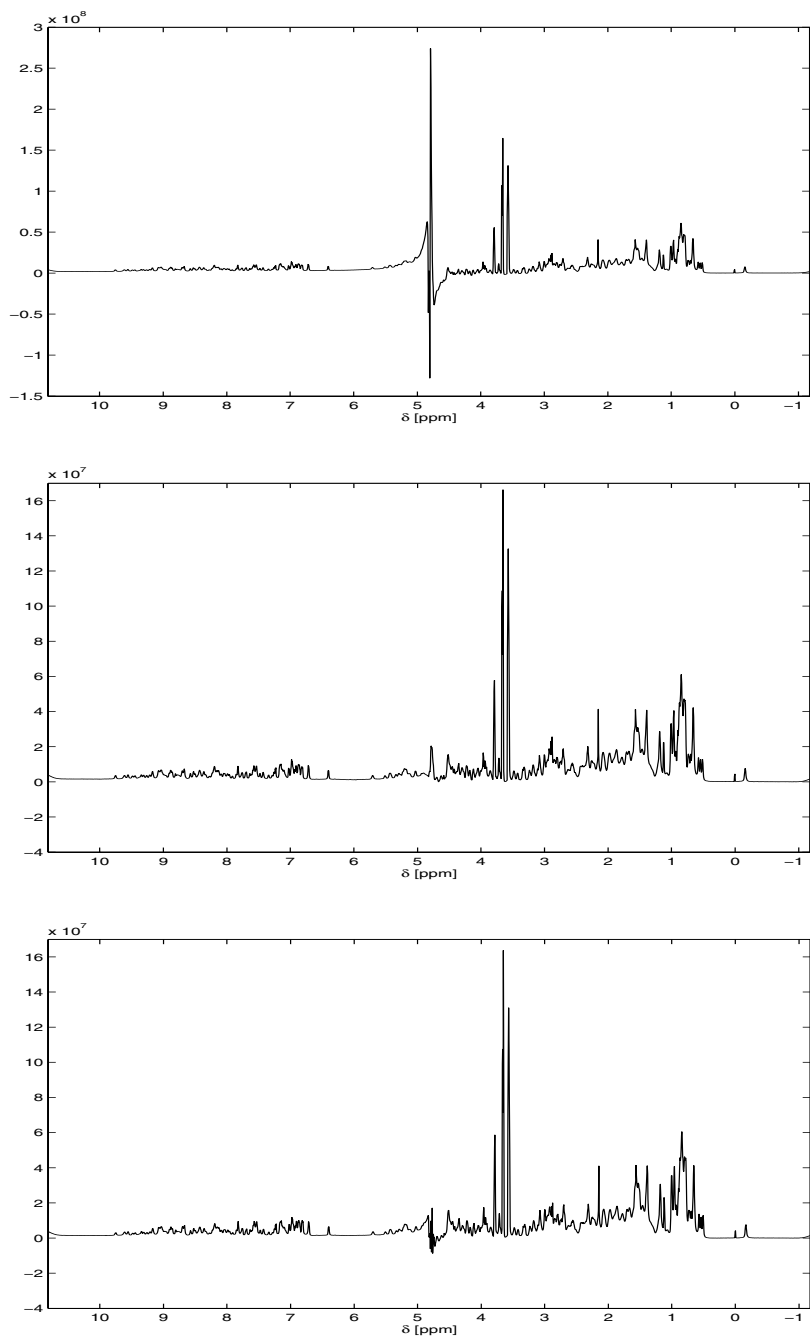
To test the performance of the algorithm AutoAssign in combination with the algorithm dAMUSE we applied it the following four sets of 2D NOESY proton NMR spectra of proteins dissolved in water:

- First we applied the algorithms to 2D NOESY proton NMR spectra of the cold-shock protein TmCSP of the bacterium *Thermotoga maritima*. It represents a small globular protein with 66 amino acids [4].
- The spectra in Fig. 1 refer to the RAS-binding domain (RBD) of the protein RalGDS. The domain forms a 87 amino acids long C-terminus of the latter protein [1].
- The synthetic peptide P11 consists of 24 amino acids only and represents the helix H11 of the human Glutathion reductase [6].
- Finally the 2D NOESY NMR spectra of the *Histidine containing phospho-carrier protein* HPr of the bacterium *Staphylococcus carnosus* with 88 amino acids have been analyzed.

In each case a number  $N_{IC} = N_{PC}$  of uncorrelated components has been estimated using dAMUSE and  $N_w$  components have been automatically assigned to the water artifact using the algorithm AutoAssign.

**Table 1.** Parameter values for the embedding dimension of the feature space of dAMUSE ( $M_{dAMUSE}$ ) and localPCA ( $M_{localPCA}$ ), the number ( $K$ ) of sampling intervals used per delay in the trajectory matrix, the number of clusters ( $k_c$ ) used with local PCA in AutoAssign, the number  $N_{pc}$  of principal components retained after the first step of the GEVD and the half-width ( $\sigma$ ) of the Gaussian filter used in the algorithm dAMUSE

| Parameter         | TmCSP | RalGDS | P11 | HPr  |
|-------------------|-------|--------|-----|------|
| $M_{localPCA}$    | 40    | 50     | 30  | 50   |
| $M_{dAMUSE}$      | 2     | 4      | 3   | 2    |
| $k_c$             | 2     | 2      | 2   | 2    |
| $N_{PC} = N_{IC}$ | 168   | 168    | 148 | 396  |
| $N_w(dAMUSE)$     | 46    | 56     | 46  | 160  |
| $\sigma$          | 0.3   | 0.1    | 0.3 | 0.03 |
| K                 | 1     | 1      | 1   | 1    |



**Fig. 1.** RalGDS: a)- 1D slice of the original 2D NOESY spectrum corresponding to the shortest evolution period  $t_1$ , b)- corresponding protein spectrum obtained by subtracting the local PCA approximation to the FID of the water artifact from the total FID, c)- corresponding protein spectrum reconstructed with the dAMUSE and AutoAssign algorithms

**Table 2.** Signal-to-noise ratios of the reconstructed spectra of TmCSP, RalGDS, P11 and HPr obtained dAMUSE and AutoAssign

|        | SNR [dB] |        |      |      |
|--------|----------|--------|------|------|
|        | TmCSP    | RalGDS | P11  | HPr  |
| dAMUSE | 19.3     | 21.9   | 21.9 | 19.8 |

To estimate a local PCA approximation of the water artifact, data have been projected into a  $M_{localPCA}$ -dimensional feature space and  $k_c$  clusters have been determined in feature space with a *k-means* algorithm. Also a Gaussian filter with width  $\sigma$  centered near the water resonance in the 1D proton NMR spectra has been used. The algorithm AutoAssign automatically identified a number  $N_w$  of uncorrelated components which have to be assigned to the water artifact. The algorithm dAMUSE was used to solve the corresponding BSS problem. Remember that the automatic assignment has been done with the FIDs corresponding to the shortest evolution period only and it is assumed that this assignment also holds for all other evolution periods as well. Note that all parameters have been varied and optimal parameters have been selected according to the best minimum of the cost function of the SA-algorithm. The parameters yielding the best minimum of the cost function with the simulated annealing algorithm are collected in Table 1.

With all experimental spectra the SNR has been determined relative to the *approximated* protein spectra, which were obtained by subtracting a local PCA approximation of the FID of the water artifact from the total FID as explained in the theory section. Hence with all experimental spectra the corresponding *approximated* spectra formed the reference against which the *reconstructed* spectra have been compared. The SNRs of the reconstructed spectra obtained with dAMUSE plus AutoAssign are summarized in Table 2. Fig. 1 shows a 1D slice of the 2D NOESY NMR spectrum together with the *approximated* spectrum. The local PCA yields a very good approximation of the contribution of the water artifact to the total FID. After subtraction and Fourier transformation only a small water peak remains in the resulting protein spectrum. The *reconstructed* spectrum obtained with the algorithms dAMUSE and AutoAssign is also shown in Fig. 1. Excellent results are also obtained with no intensity distortions of the protein peaks of the spectra. Also all baseline distortions are perfectly straightened out and even peaks very close to the water artifact are well recovered. Comparable results are obtained with the other proteins as well but are not shown because of space limitations.

## References

1. M. Geyer, C. Herrmann, S. Wohlgemuth, A. Wittinghofer, and H. R. Kalbitzer. Structure of the ras-binding domain of ralgef and implications for ras binding and signalling. *Nature Structural Biology*, 4:694–699, 1997.



2. M. Ghil, M.R. Allen, M. D. Dettinger, and K. Ide. Advanced spectral methods for climatic time series. *Reviews of Geophysics*, 40(1):3.1–3.41, 2002.
3. S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:pp. 671, 1983.
4. W. Kremer, B. Schuler, S. Harrieder, M. Geyer, W. Gronwald, C. Welker, R. Jänicke, and H. R. Kalbitzer. Solution NMR structure of the coldshock protein from the hyperthermophilic bacterium *thermotoga maritima*. *Eur. J. Biochem.*, 268:2527 – 2539, 2001.
5. V. Moskvina and K. M. Schmidt. Approximate projectors in singular spectrum analysis. *SIAM Journal Mat. Anal. Appl.*, 24(4):932–942, 2003.
6. A. Nordhoff, Ch. Tziatzios, J. A. V. Broek, M. Schott, H.-R. Kalbitzer, K. Becker, D. Schubert, and R. H. Schirme. Denaturation and reactivation of dimeric human glutathione reductase. *Eur. J. Biochem*, pages 273–282, 1997.
7. K. Stadlthanner, F. Theis, E. W. Lang, A. M. Tomé, W. Gronwald, and H. R. Kalbitzer. A matrix pencil approach to the blind source separation of artifacts in 2D NMR spectra. *Neural Information Processing - Letters and Reviews*, 1:103 – 110, 2003.
8. K. Stadlthanner, A. M. Tomé, F. J. Theis, W. Gronwald, H. R. Kalbitzer, and E. W. Lang. Blind source separation of water artifacts in NMR spectra using a matrix pencil. In *Fourth International Symposium On Independent Component Analysis and Blind Source Separation, ICA'2003*, pages 167–172, Nara, Japan, 2003.
9. K. Stadlthanner, A. M. Tomé, F. J. Theis, W. Gronwald, H. R. Kalbitzer, and E. W. Lang. On the use of independent component analysis to remove water artifacts of 2D NMR protein spectra. In *7th Portuguese Conference on Biomedical Engineering, BIOENG'2003*, Lisbon, Portugal, 2003.
10. A. R. Teixeira, A. M. Tomé, E. W. Lang, and K. Stadlthanner. Delayed AMUSE - A Tool for blind source separation and Denoising. In *Independent Component Analysis and Blind Signal Separation, Proc. ICA'2004*, volume LNCS 195, pages 287–294, 2004.
11. Ana Maria Tomé. An iterative eigendecomposition approach to blind source separation. In *3rd Intern. Conf. on Independent Component Analysis and Signal Separation, ICA'2003*, pages 424–428, San Diego, USA, 2001.
12. Ana Maria Tomé, Ana Rita Teixeira, Elmar Wolfgang Lang, Kurt Stadlthanner, and A.P. Rocha. Blind source separation using time-delayed signals. In *International Joint Conference on Neural Networks, IJCNN'2004*, volume CD, Budapest, Hungary, 2004.
13. Lang Tong, Ruey-wen Liu, Victor C. Soon, and Yih-Fang Huang. Indeterminacy and identifiability of blind identification. *IEEE Transactions on Circuits and Systems*, 38(5):499–509, 1991.

# An ICA Approach to Detect Functionally Different Intra-regional Neuronal Signals in MEG Data

Giulia Barbati<sup>1</sup>, Camillo Porcaro<sup>1</sup>, Filippo Zappasodi<sup>1,2</sup>,  
and Franca Tecchio<sup>1,2</sup>

<sup>1</sup> AFaR – Center of Medical Statistics and Information Technology,  
Fatebenefratelli Hospital, Rome, Italy

<sup>2</sup> ISTC – Institute of Science and Technologies of Cognition - CNR, Rome, Italy

**Abstract.** Cerebral processing mainly relies on functional connectivity among involved regions. Neuro-imaging techniques able to assess these links with suitable time resolution are electro- and magneto-encephalography (EEG and MEG), even if it is difficult to localize recorded extra-cranial information, particularly within restricted areas, due to complexity of the ‘inverse problem’. By means of Independent Component Analysis (ICA) a procedure ‘blind’ to position and biophysical properties of the generators, our aim in this work was to identify cerebral functionally different sources in a restricted area. MEG data of 5 subjects were collected performing a relax-movement motor task in 5 different days. ICA reliably extracted neural networks differently modulated during the task in the frequency range of interest. In conclusion, a procedure solely based on statistical properties of the signals, disregarding their spatial positions, was demonstrated able to discriminate functionally different neuronal pools activities in a very restricted cortical area.

## 1 Introduction

Signal timing is essential for brain function. In order to understand the mechanisms sustaining human brain processing, it is crucial to obtain measures with the same time resolution of the cerebral processing itself: this is possible by studying directly the electrical neuronal activity. Neurophysiological techniques (electro- and magneto-encephalography, EEG and MEG) allow this investigation with the characteristic property to detect large neuronal synchronized pools activity. The main goal is to access information about underlying neural sources activity starting from the recorded EEG and MEG signals. Standard approach has been up to now to solve the so called ‘inverse problem’, i.e. to use Maxwell’s equations, which codify the relationship between electrical currents and the generated electrical and magnetic fields, to calculate spatial distribution of the intra-cerebral currents starting from the magnetic and/or electric field detected in an enough-wide extra-cephalic spatial surface. This problem does not admit a unique solution, i.e. many different current configurations induce the same field distribution. For this reason, it is necessary to pose

supplementary information, in order to define the generating currents [1]; the procedure provides position, intensity and direction of the modeled cerebral currents related to neural networks placed in different cerebral areas. Extensive reports have been devoted to show that neuronal networks with specific functional roles are highly interconnected one to each other in a distributed network [2] and cortical networks controlling for integrated tasks could be partially overlapping in space [3]. For this reason, modeling cerebral sources on the base of their positions limits the ability in studying the neural activity. Moreover, cerebral processing mainly relies on functional connectivity among involved regions, with changes in the synchronization of the neuronal firing rhythms [4]; if synchronization within a restricted cortical area is under investigation, differentiating different neural components on the base of their positions severely limits the evaluation.

On the last decade, a class of feature extraction procedures not based on a priori assumptions on source spatial and physical properties, named blind source separation techniques (BSS) [5,6] was developed. This approach discriminates different sources on the base of the statistical properties of their generated signal. A subclass of algorithms is the Independent Component Analysis (ICA). The use of ICA algorithm is based on three assumptions: (1) the linear and instantaneous mixing of sources contribution to the sensors; (2) the existence of sources that are statistically independent from one another; (3) the stationarity of the mixing process. The good applicability of the procedure on recordings of electric and magnetic brain activity mainly stems from point (1), which is true without approximation. The main difference of the ICA procedure with respect to the inverse problem approach is contained in the discrimination criteria based on the probability distribution of the sources generated signal.

ICA algorithms were successfully applied to EEG and MEG recordings mostly in order to identify and remove artifactual from the physiological activities of interest [7]. By using ICA procedure, Makeig and colleagues [8] successfully provided answer to a long debated question, whether responses recorded from outside the scalp result from stimulus-evoked brain events or stimulus-induced changes in ongoing brain dynamics. This answer comes out from the ability of the ICA approach to extract single trial responses instead of identifying them by averaging out several responses; in this way the modulation of the continuous ongoing activity induced by stimulus processing could be assessed in detail.

Aim of the present work was to demonstrate that ICA make us able to identify sources with different functional roles in a very restricted area, without a priori assumption on the model of underlying cerebral sources; to do this, the rolandic neuronal activity in the dominant hemisphere was focused, comparing a very simple hand motor task to rest state. Neuronal involvement during hand movement was assessed via a well established criteria of neuronal areas behavior: the on-going activity reduction during movement with respect to relax [9]. To strengthen consistency of this approach, by evaluating the test-retest reliability of the procedure, 5 subjects were repeated 5 times in different days. Moreover, since it is always a major problem, both in the general theory and in real ICA applications, how many and which among the estimated components (ICs) are to be selected and flagged as really describing different sources of activity, cluster analysis has recently been used to group ICs verifying the same flagging criteria; information theory distances [10,11] or

generalized Euclidean distance [8] have been the more frequent clustering measures used. In the present work, accordingly with our aim, a “functional” measure was introduced in a ICs clustering algorithm: a standardized Euclidean distance between “IC reactivity during movement with respect to relax” (defined in method section).

## 2 Methods

### 2.1 Experimental Setup

Five healthy subjects (34  $\pm$ 6 years; 3 male; right-handed) were enrolled for the study.

Subject was required to lie on a bed, with open eyes in order to reduce the effects of the occipital spontaneous activity in the rolandic region. During a time interval of 4 minutes, she/he was alternating periods of 20 sec in the two following conditions: 1) complete relax [*Relax*] and 2) opening-closing the right hand [*Hand\_movement*] at a frequency of around 2 c/sec. The recording session was repeated for each subject five times in different days (mean intersession period 3 weeks). Brain magnetic fields were recorded by means of a 28-channel system [12] operating in a magnetically shielded room, the active channels being regularly distributed on a spherical surface (13.5 cm of curvature radius) and covering a total area of about 180 cm<sup>2</sup>. Recording apparatus was centred on the C3 position of the 10-20 International EEG system. Ag–AgCl electrodes served for recording electrooculogram (EOG) and electrocardiogram (ECG) in order to control for eye blinking or cardiac interferences. Surface electromyographic (EMG) signal from right flexor digitorum superficialis muscle was acquired to monitor the hand movement and relaxation. MEG and EMG data were sampled at 1000 Hz (pre-sampling analogical filter 0.48-256 Hz), and collected for off-line processing.

### 2.2 Data Analysis

The ICA algorithm ERICA [13] was applied on data recordings of each session for each subject, with PCA pre-whitening and reduction; number of dimensions chosen to preserving at least 94% of total explained variance. The algorithm was applied on full length recordings, containing continuous successive periods of *Relax* and *Hand\_movement* conditions. After ICs estimation, the ‘switching system’ described in Barbati et al. [7]<sup>1</sup> was applied, in order to exclude non-cerebral activities, as for example eyes movements, cardiac artefacts or environmental noise. The output of the switching system resulted in a certain number of non-artefactual components coming from each session of each subject. Separately for each subject, a clustering procedure was adopted to identify ‘functionally distinct’ ICs groups. Input matrix for cluster analysis was constructed as follows: power spectrum density (PSD) difference between *Relax* and *Hand\_movement* conditions for each IC was computed; the two frequency bands of interest: alpha (7.5-12.5 Hz) and beta (13-25 Hz) were selected,

---

<sup>1</sup> In this application, with respect to procedure published in [7], the switching system did not use the ‘global kurtosis’ index and the threshold of PSD correlation coefficient with EOG and ECG was modified from the ‘maximum value’ to ‘greatest than 0.90’.

and the areas of the PSD difference in these two bands were computed (IC reactivity area). The clustering measure between two ICs was the 2-dimensions Standardized Euclidean Distance between “IC reactivity area” in alpha and beta. A hierarchical clustering algorithm based on average linkage was used to obtain a dendrogram for each subject. No a priori threshold was used to define the final number of clusters. To test for repeatability of obtained clusters, i.e. to check whether they contained ICs coming from each session, a ‘repeatability score’ was defined as 1 if a cluster was present in at least all but one sessions, 0 otherwise. To assign a functional role to the identified clusters, a MANOVA analysis with between-subject factor CLUSTER was performed, considering as dependent variables the 2-D “IC reactivity area” in alpha and beta bands. Non repeatable clusters were excluded from the analysis. Correction for multiple comparisons was applied if necessary, i.e. if more than two repeatable clusters were present. On the basis of MANOVA post-hoc comparisons results, clusters were classified by the size of their alpha and beta means reactivities and labelled as “N=not reactive”, “L=low reactive”, “H=high reactive”. We defined a ‘reactivity score’ by assigning a score of 1 for N, a score of 2 for L, and a score of 3 for H. Finally, we computed a ‘total score’ by multiplying the ‘repeatability score’ with the ‘reactivity score’. Source labels were assigned for each subject on the basis of the ‘total score’ of the clusters: *Mu* was the greatest score, *Alpha* the second one, ‘low/not reactive’ the third, ‘noise’ the 0-score.

To describe spatial characteristics of the sources labelled as *Mu* and *Alpha*, the ICs belonging the respective clusters were separately retro-projected, so as to obtain the field distribution along time ( $\mu_{\text{rec}}$  and  $\alpha_{\text{rec}}$ ). A moving equivalent current dipole (ECD) model inside an homogeneous best-fitted sphere was used; the ECD characteristic (spatial coordinates, strength and orientations) were calculated at 1 ms intervals in 8 sec time epochs in the *Relax* and *Hand\_movement* conditions. ECD coordinates were expressed in a right-handed Cartesian coordinate system defined on the basis of three anatomical landmarks<sup>2</sup>. ECD characteristic were also calculated in the same time epochs using the field distribution of original signals separately filtered in beta (**Filt  $\beta$** ) and alpha bands (**Filt  $\alpha$** ), to somehow extract respectively activities related to the *Mu* and *Alpha* sources. To compare source positions related to these conditions to the marker of the hand sensory cortical projection, we recorded in the same sessions the evoked activity following the right median nerve stimulation. Standard parameters for nerve stimulation were used (interstimulus 631 ms, intensity adjusted to provoke painless thumb twitch). About 200 artefact free trials were averaged to obtain the evoked response. Position and orientation of ECD related to M20, the initial response deflection around 20 ms from the stimuli, were calculated.

### 3 Results

Reactivity during hand movement with respect to relax, i.e. the reduction of power during movement all over the frequency range between 2-45 Hz most evident in alpha

<sup>2</sup> x-axis passing through the two preauricular points directed rightward, the positive y-axis passing through the nasion, the positive z-axis consequently.

and beta bands, was clearly visible over the recording channels in all repetitions for each subject; we show one paradigmatic subject in Figure 1a.

### 3.1 IC Extraction

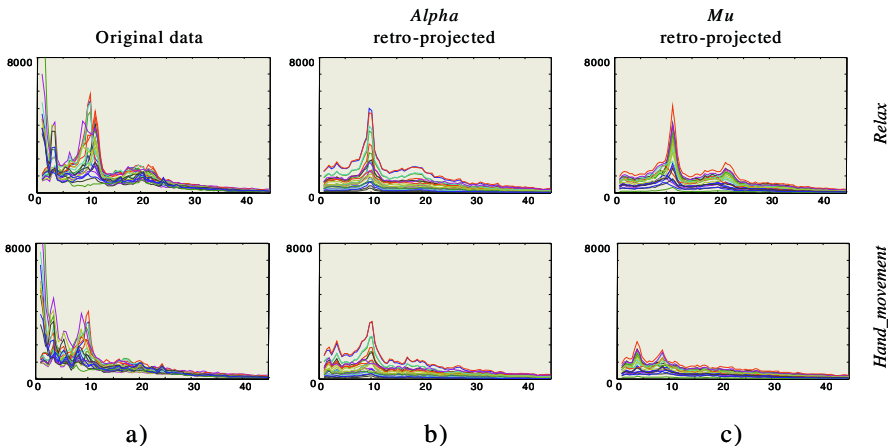
The IC extraction in the repetitions of the 5 subjects produced an average number for each session of  $11 \pm 3$  ICs. The mean number of ICs automatically excluded in each session by the switching system was  $5 \pm 2$ , and the mean number of retained ICs that formed the input cluster matrix was  $7 \pm 2$ . Two subjects (S2, S5) had only four sessions. The total number of ICs given to cluster algorithm for each subject was respectively: 35, 28, 31, 32, 28.

### 3.2 IC Clustering

Although no a priori threshold was used to define the final number of clusters, consistently 2-3 repeatable clusters (at least 4 out of 5 sessions or 3 out of 4) were

**Table 1.** For the first two subjects, characteristics of the obtained clusters are reported in the first 4 columns; steps to assign source labels are reported in the last 4 columns

| Subject code | Cluster identifier | Cluster size | Rep. | Rep_Score | Rec_score | T_Score | Source Label |
|--------------|--------------------|--------------|------|-----------|-----------|---------|--------------|
| S1           | C1                 | 5            | 4/5  | 1         | L         | 2       | Alpha        |
| S1           | C2                 | 14           | 4/5  | 1         | N         | 1       | no_reac      |
| S1           | C3                 | 1            | 1/5  | 0         | -         | 0       | Noise        |
| S1           | C4                 | 15           | 5/5  | 1         | H         | 3       | Mu           |
| S2           | C1                 | 18           | 4/4  | 1         | L         | 2       | Alpha        |
| S2           | C2                 | 1            | 1/4  | 0         | -         | 0       | noise        |
| S2           | C3                 | 9            | 4/4  | 1         | H         | 3       | Mu           |



**Fig. 1.** In a representative subject (S1), the PSD in *Relax* (**top**) and during *Hand\_movement* (**bottom**) are shown for original channel signals (**a**), *Alpha* retro-projected channel signals (**b**) *Mu* retro-projected channel signals (**c**). Note that in the original channel signals a clear reactivity is present in alpha and less in beta bands; in the retro-projected channel signals, activity is instead clearly decomposed in a factor with both frequency ranges low reactive (*Alpha*) and another with both frequency ranges high reactive (*Mu*)

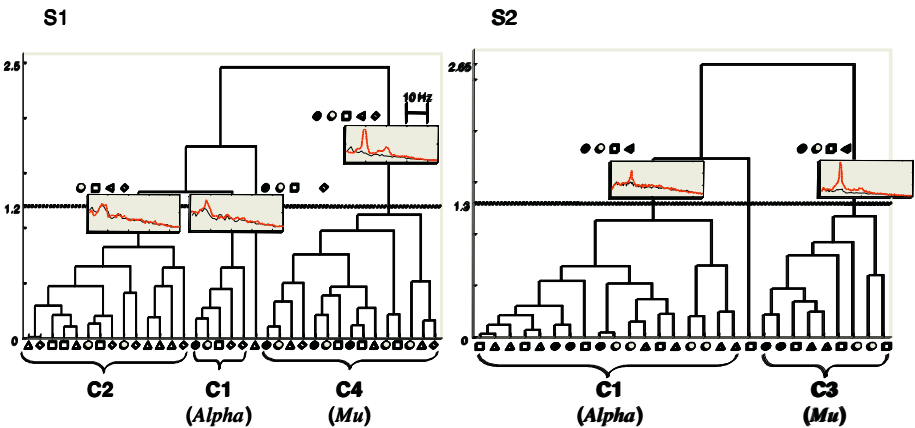
found in each subject, with a cut-off around the 50% of the dendrogram maximum distance (Figure 2). A total number of 3 to 5 clusters resulted with this cutoff. Size and repeatability of each cluster for the first two subjects are reported in Table 1.

### 3.3 Source Labelling

MANOVA indicated in all subjects presence of clusters with significantly different reactivity ( $p < 0.01$  consistently for each subject). Post-hoc comparisons produced cluster reactivity sorting as defined in method section. Accordingly, clusters were scored from the lowest ('reactivity score'=1) to the highest (=3, Table 1). On the base of this 'reactivity score' and the previously defined 'repeatability score' the clusters were flagged. All the ICs belonging the flagged clusters were considered to represent a cerebral activity source (Table 1). *Alpha* and *Mu* sources resulted both present in four out of five subjects.

### 3.4 Labelled Source Localization

No difference was found between ECD position and orientation in *Relax* with respect to *Hand\_movement* conditions, for any of the retro-projected channel signals ( $\mu_{rec}$ ,  $\alpha_{rec}$ ) or filtered original data (**Filt  $\beta$**  and **Filt  $\alpha$** ). Sources obtained from field distribution constructed by retro-projecting all ICs belonging to *Alpha* and *Mu* sources resulted much more concentrated in position and orientation than those from band filtering original data (*Vol* in Table 2).



**Fig. 2.** Dendrograms for the first two subjects. Horizontal dot line indicates cutoff value. Different symbols represent ICs coming from different sessions. Prototypical PSD in relax (fine line) and in contraction (bold line) for each repeatable cluster is depicted. Beside the physiological inter-subject PSD variability, it is clearly evident the presence of high- and low-reactive ICs. In particular, *Mu* source reacted during *Hand\_movement* with respect to *Relax* much stronger in alpha and beta bands with respect to *Alpha* source

**Table 2.** For the first two subjects, median across the 8 s of the ECD positions (x, y, z) and direction unitary vectors (px, py, pz) of signals referring to  $\mu_{rec}$ ,  $\alpha_{rec}$ , **Filt  $\beta$**  and **Filt  $\alpha$** . M20 of the source activated at 20 ms from the median nerve stimulation at wrist is reported. In square brackets 5th and 95th percentiles of the ECD coordinates are indicated. Volume of the cube containing 90% of ECD positions (side=95th-5th coordinate percentiles) is indicated in the four cases (Vol). Position and direction of  $\mu_{rec}$  and M20 are highly consistent (mean distance 19.6 mm), with the  $\mu_{rec}$  ECD anterior to M20 ECD (7 mm, paired-t test p=0.04)

|    |                                 | position   |            |            | Vol<br>(cm <sup>3</sup> ) | direction  |            |            |
|----|---------------------------------|------------|------------|------------|---------------------------|------------|------------|------------|
|    |                                 | x (mm)     | y (mm)     | z (mm)     |                           | px         | py         | pz         |
| S1 | M20                             | -43        | 4          | 90         |                           | .61        | .45        | .65        |
|    | $\mu_{rec}$                     | <b>-40</b> | <b>7</b>   | <b>115</b> | <b>1</b>                  | <b>.29</b> | <b>.86</b> | <b>.43</b> |
|    |                                 | [-41,-34]  | [-2,10]    | [102,116]  |                           | [.19,.52]  | [.65,.90]  | [.39,.55]  |
|    | $\alpha_{rec}$                  | <b>-47</b> | <b>-41</b> | <b>108</b> | <b>1</b>                  | <b>.65</b> | <b>.17</b> | <b>.74</b> |
|    |                                 | [-55,-25]  | [-45,-40]  | [107,113]  |                           | [.46,.65]  | [.11,.66]  | [.59,.75]  |
|    | <b>Filt <math>\beta</math></b>  | <b>-38</b> | <b>7</b>   | <b>89</b>  | <b>285</b>                | <b>.27</b> | <b>.83</b> | <b>.49</b> |
|    |                                 | [-82,-21]  | [-42,23]   | [47,119]   |                           | [.03,.60]  | [.21,.94]  | [.22,.95]  |
|    | <b>Filt <math>\alpha</math></b> | <b>-56</b> | <b>-4</b>  | <b>98</b>  | <b>238</b>                | <b>.46</b> | <b>.75</b> | <b>.53</b> |
|    |                                 | [-97,-28]  | [-48,-12]  | [44,140]   |                           | [.07,1.00] | [.18,.94]  | [.30,.97]  |
| S2 | M20                             | -45        | 0          | 105        |                           | .47        | .83        | .31        |
|    | $\mu_{rec}$                     | <b>-34</b> | <b>4</b>   | <b>99</b>  | <b>1</b>                  | <b>.44</b> | <b>.68</b> | <b>.59</b> |
|    |                                 | [-38,-33]  | [3,10]     | [94,108]   |                           | [.33,.51]  | [.52,.78]  | [.52,69]   |
|    | $\alpha_{rec}$                  | <b>-1</b>  | <b>2</b>   | <b>98</b>  | <b>84</b>                 | <b>.26</b> | <b>.92</b> | <b>.28</b> |
|    |                                 | [-15,11]   | [-30,45]   | [72,115]   |                           | [.04,.91]  | [.11,.99]  | [.04,.59]  |
|    | <b>Filt <math>\beta</math></b>  | <b>-29</b> | <b>13</b>  | <b>103</b> | <b>304</b>                | <b>.38</b> | <b>.72</b> | <b>.55</b> |
|    |                                 | [-62,0]    | [-35,37]   | [58,126]   |                           | [.09,.73]  | [.15,.93]  | [.26,.78]  |
|    | <b>Filt <math>\alpha</math></b> | <b>-36</b> | <b>13</b>  | <b>110</b> | <b>327</b>                | <b>.31</b> | <b>.81</b> | <b>.51</b> |
|    |                                 | [-74,-3]   | [-27,23]   | [44,136]   |                           | [.04,.61]  | [.25,.96]  | [.22,.83]  |

## 4 Discussion

ICA has already been demonstrated able to identifying source time courses along the whole analysis period, allowing both the study of the modulation of continuous ongoing activity induced by the stimulus and extracting single trial responses instead of averaging out several trials, which introduces smoothing and loss of information. Present ICA and clustering procedure provided evidence that it is possible to separate neural sources with superimposed spectral contents and different functional behaviors in a very restricted cortical area. In particular, compatibly with the known strong inter-subject PSD variability, *Mu* source reacted during opening and closing the hand much stronger than the *Alpha* source in alpha and beta bands in all subjects; i.e., it was demonstrated, for the first time by extra-cranial recordings within a restricted cortical area, that on the base solely of the statistical properties of the generated signal it is possible to identify neuronal pools activities with different functional roles. The result has been obtained by “flagging” the extracted ICs by a clustering procedure based on functional source properties, i.e. the reactivity to the motor task, disregarding spatial characteristics. Since cluster analysis is mainly a descriptive statistical tool, further tests were performed on obtained clusters to verify that they were differently reactive at a statistically significant level in alpha and beta bands. It



remains an open question whether this, and others neurophysiological ICA and BSS applications results, are due to intrinsic statistical properties of different functional neuronal pools or to algorithmic limitations in obtaining perfect independence. A surprising side-effect in our application was that these broad-band activities generators displayed a more "dipole-like" magnetic field scalp distribution than the raw data, even though neither the locations of the sensors nor the biophysics of volume propagation figure into the ICA algorithm. This could represent an improvement when the involved source position is under investigation. The confirmed efficiency in automatically extract the non-artefactual ICs already applied by Barbati and colleagues [7], promise to be very useful particularly in whole-head helmet arrays recordings. In conclusion, evidences are provided that, from a physically restricted cortical area, ICA is able to identify functionally different neuronal pools activities, repeatable intra-subject, non-dependent on spatial location; the extracted cerebral sources obtained in this way are promising candidates for the evaluation of intra-regional synchronization during stimulus and task-specific processing.

## References

1. Babiloni F, Carducci F, Cincotti F, Del Gratta C, Pizzella V, Romani GL, Rossini PM, Tecchio F, Babiloni C. Linear inverse source estimate of combined EEG and MEG data related to voluntary movements. *Hum Brain Mapp.* 2001 Dec;14(4):197-209.
2. Poggio T, Bizzi E. Generalization in vision and motor control. *Nature.* 2004; 431:768-74.
3. Indovina I, Sanes JN. On somatotopic representation centers for finger movements in human primary motor cortex and supplementary motor area. *Neuroimage.* 2001; 13:1027-34.
4. Engel AK, Fries P, Singer W. Dynamic predictions: oscillations and synchrony in top-down processing. *Nat Rev Neurosci* 2001; 2: 704-716
5. Comon P. Independent component analysis: a new concept? *Singal processing*, 1994; 36:287-314
6. Cichocki A, Amari S-I. Adaptive blind signal and image processing, 2002, J. Wiley & Sons
7. Barbati G, Porcaro C, Zappasodi F, Rossini PM, Tecchio F. Optimization of an Independent Component Analysis approach for artifact identification and removal in magnetoencephalographic signals. *Clinical Neurophysiology*, 2004 May; 115(5):1220-32.
8. Makeig S, Westerfield M, Jung TP, Enghoff S, Townsend J, Courchesne E, Sejnowski TJ. Dynamic brain sources of visual evoked responses. *Science.* 2002; 295(5555):690-4
9. Rossi S, Tecchio F, Pasqualetti P, Olivelli M, Pizzella V, Romani G-L, Passero S, Battistini N, Rossini PM. Somatosensory processing during movement observation in humans. *Clin Neurophysiol*, 2002; 113:16-24
10. Himberg J, Hyvarinen A, Esposito F. Validating the independent components of neuroimaging time series via clustering and visualization. *Neuroimage.* 2004; 22:1214-22
11. Meinecke F, Ziehe A, Kawanabe M, Muller KR. A resampling approach to estimate the stability of one-dimensional or multidimensional independent components. *IEEE Trans Biomed Eng.* 2002; 49:1514-25.
12. Tecchio F, Rossini PM, Pizzella V, Cassetta E, Romani GL. Spatial properties and interhemispheric differences of the sensory hand cortical representation: a neuromagnetic study. *Brain Res.* 1997 Aug 29;767(1):100-8.
13. Cruces-Alvarez S, Castedo-Ribas L, Cichocki A. Robust blind source separation algorithms using cumulants. *Neurocomputing* 2002;49: 87-118.

# Filtering-Free Blind Separation of Correlated Images

Frédéric Vrins, John A. Lee, and Michel Verleysen\*

Université catholique de Louvain, Machine Learning Group,  
Place du Levant, 3, 1380 Louvain-la-Neuve, Belgium  
{vrins, lee, verleysen}@dice.ucl.ac.be  
www.ucl.ac.be/mlg

**Abstract.** When using ICA for image separation, a well-known problem is that most often a large correlation exists between the sources. Because of this dependence, there is no more guarantee that the global maximum of the ICA contrast matches the outputs to the sources. In order to overcome this problem, some preprocessing can be used, like e.g. band-pass filtering. However, those processings involve parameters, for which the optimal values could be tedious to adjust. In this paper, it is shown that a simple ICA algorithm can recover the sources, without any other preprocessing than whitening, when they are correlated in a specific way. First, a single source is extracted, and next, a parameter-free postprocessing is applied for optimizing the extraction of the remaining sources.

## 1 Introduction

In the recent past years, Independent Component Analysis (ICA) has been a fast growing research topic; many algorithms have been developed to solve the ICA problem. Though they share the same goal, all algorithms approach the problem differently and may have different performances on specific applications. This explains why new ICA contrasts are still developed nowadays. For instance, the support width measure (SWM) has been recently suggested as cost function for ICA; its main advantages are its theoretical convexity for bounded sources, its geometrical interpretation and its simplicity [1].

A particular application of ICA is the blind separation of mixed images. This application does not entirely fulfill the assumptions of the canonical ICA problem, since natural images can be highly correlated, i.e. the sources are not independent anymore. Consequently, it can be expected that usual ICA algorithms would fail to recover the source images given only linear mixtures of them. Indeed, even if exceptions seem to exist [2], two mixed images can be

---

\* M.V. is a Senior Research Associate of the Belgian F.N.R.S. The authors are grateful to A. Cichocki from BSI RIKEN for discussion on MDS-ICA. This work was partially supported by the Belgian F.M.S.R. (project 3.4590.02).

more independent than the dependent sources. However, several tools have been designed to address this issue. The most efficient one seems to be filtering. In that case, it is assumed that a frequency band exists, for which the source images are statistically independent. By filtering the raw image mixtures outside this frequency band, new mixtures are obtained and processed by a usual ICA algorithm. Once the latter has converged, the computed unmixing matrix is used for separating the initial (unfiltered) mixed images [3].

Even if the previous method looks very efficient, additional parameters appear, like the cutoff frequencies and the order of the filter, which may be difficult to adjust. For instance, finding the frequency band that makes unknown images fully independent, starting from mixtures of them may be a tedious task. To our knowledge, no simple and automatic method exists to solve this problem.

In this work, we propose to use the SWICA algorithm [4] for optimizing the SWM cost function and for solving image separation problems without any other preprocessing than whitening. The method is tested on a simple example, involving two correlated source images (landscapes). SWICA is first applied to extract a single source image. Next, it is modified to optimize the separation of the second source image. Paradoxically, it is an apparent weakness of the SWM criterion that allows us to separate correlated images without any filtering.

The remainder of this paper is organized as follows. Section 2 introduces the SWM contrast and the corresponding algorithm (SWICA). Section 3 deals with additional theoretical issues, whereas experiments are detailed in Section 4. Finally, conclusions are drawn in Section 5.

## 2 ICA by Support Width Minimization

In order to make this paper self-contained, this section summarizes the SWM criterion [1] and the related ICA algorithm [4] (SWICA).

### 2.1 The SWM Cost Function

Consider the linear mixing model  $\mathbf{x} = \mathbf{A}\mathbf{s}$ , where  $\mathbf{s} = [s_1, \dots, s_n]^T$  is a source vector made of  $n$  independent and zero-mean random variables,  $\mathbf{A}$  is a square mixing matrix and  $\mathbf{x} = [x_1, \dots, x_n]^T$  is the vector of the mixtures. In general, the mixtures are dependent and correlated, but it is easy to find a decorrelation transformation  $\mathbf{V}$  such that  $\mathbf{z} = \mathbf{V}\mathbf{x}$  satisfies the whiteness condition  $E\{\mathbf{z}\mathbf{z}^T\} = \mathbf{I}_n$ . Next, if the sources are white, then an ICA algorithm can be run on the whitened mixtures  $\mathbf{z}$  in order to recover the sources. More formally, the ICA algorithm identifies the orthogonal matrix  $\mathbf{W}$  in the unmixing model  $\mathbf{s} \approx \mathbf{y} = \mathbf{W}\mathbf{z}$ . The symbol ‘ $\approx$ ’ means that the transfer matrix  $\mathbf{C} \triangleq \mathbf{W}\mathbf{V}\mathbf{A}$  is equal to  $\mathbf{P}\mathbf{D}$ , where  $\mathbf{P}$  and  $\mathbf{D}$  are permutation and scaling matrices, respectively.

Several methods exist to find  $\mathbf{W}$ . One of them consists in minimizing the support width  $\Omega(y_i)$  of each estimated source  $y_i$ , provided the sources are bounded (i.e. the support of the source distribution  $\Omega(s_i)$  is finite). In this case, it has been shown that if the number of sample points is large enough, the SWM criterion is convex. In other words, each local minimum of  $\Omega(y_i)$  satisfies  $\mathbf{c}_i = \pm \mathbf{e}_k$ ,

where  $\mathbf{c}_i$  is the  $i$ -th row of  $\mathbf{C}$  and  $\mathbf{e}_k$  the  $k$ -th row of  $\mathbf{I}_n$ . This can be understood by looking at the theoretical expression of  $\Omega(y_i)$ , expressed as a function of the source supports:

$$\Omega(y_i) = \sum_{j=1}^n \mathbf{c}_i(j) \Omega(s_j) , \quad (1)$$

where the source supports  $\Omega(s_1), \dots, \Omega(s_n)$  are positive and finite. The function  $\Omega(y_i)$  can have a minimum only if  $\mathbf{c}_i = \pm \mathbf{e}_k$ , i.e. when  $y_i$  is proportional to a source  $s_j$ .

The support minimization of the output is illustrated in Fig. 1. The first angle  $\phi$  correspond to the rotation of the source resulting from  $\mathbf{V}\mathbf{A}$  while angle  $\varphi$  is associated with the unmixing matrix  $\mathbf{W}$ . The width  $D$  is shown when  $\mathbf{W}$  has not converged yet to a satisfactory unmixing matrix, i.e.  $\Omega(y_1) = \Omega(\mathbf{w}_1\mathbf{z})$  is not minimized yet. After convergence of the algorithm, we observe  $\Omega(y_1) = D'$ , which corresponds to  $y_1 \propto s_1$ ; this minimum is global since in this example,  $\Omega(s_1) < \Omega(s_2)$ . A local minimum of  $\Omega(y_1)$  would be obtained if  $y_1 \propto s_2$ .

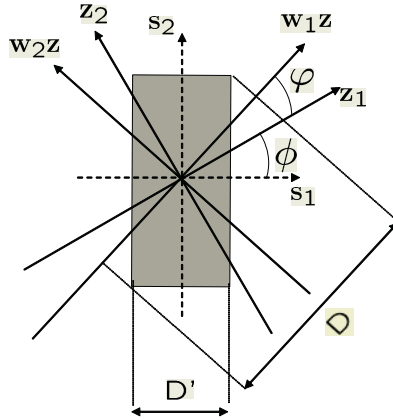


Fig. 1. SWM: principle of ICA by finding directions with minimum support width

## 2.2 The SWICA Algorithm

SWM is a single-unit ICA criterion. For this reason, when several sources have to be extracted, an algorithm based on a deflation approach must be used. In addition, it must be mentioned that the SWM contrast ( $-\Omega$ ) is not differentiable, thus making all traditional optimization procedures (fixed point, gradient ascent, etc.) unusable. The algorithm we propose takes as input the whitened mixtures and extracts the sources one after the other, by determining the corresponding row of  $\mathbf{W}$ . In order to keep  $\mathbf{W}$  orthogonal, rows of  $\mathbf{W}$  are seen as directions and are updated accordingly. For this purpose, angular variations of the current row  $\mathbf{w}_i$  towards another row  $\mathbf{w}_j$  are defined and noted as

$$\mathbf{w}_{i\uparrow j} = \cos(\alpha) \mathbf{w}_i + \sin(\alpha) \mathbf{w}_j \quad \text{and} \quad \mathbf{w}_{i\downarrow j} = \cos(\alpha) \mathbf{w}_i - \sin(\alpha) \mathbf{w}_j . \quad (2)$$

Such angular variations allow comparing the current value of the contrast ( $-\Omega(\mathbf{w}_i\mathbf{z})$ ) to surrounding values ( $-\Omega(\mathbf{w}_{i\uparrow j}\mathbf{z})$  or  $-\Omega(\mathbf{w}_{i\downarrow j}\mathbf{z})$ ). As the contrast is not differentiable, a very simple optimization procedure is proposed. Briefly put, for each row  $\mathbf{w}_i$ , the algorithm looks at the contrast value in each perpendicular direction ( $\mathbf{w}_j$  with  $i + 1 < j < n$ ), for both positive and negative angular variations. Then it updates  $\mathbf{W}$  by rotating both  $\mathbf{w}_i$  and  $\mathbf{w}_j$  according to the highest contrast value. Consequently, the algorithm keeps  $\mathbf{W}$  orthogonal.

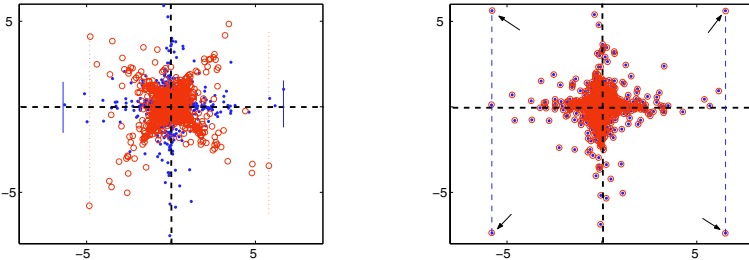
The only parameters of the algorithm are the convergence rate and the number of iterations. By construction, the algorithm is monotonic: the contrast is either increased or kept constant. Similar algorithms using more sophisticated techniques, like discrete gradient approximations (based on a  $2^{nd}$ -order Taylor expansion) have been tried too. Unfortunately, they lead to worse results than the simple proposed algorithm. In addition, they involve a larger number of parameters that are tedious to adjust. More details can be found in [4].

### 3 SWM: An Extreme Statistics Contrast

Maximizing the contrast requires an estimation of  $-\Omega(y_i)$ . If the number of observations is large enough, the following estimator measures the support width of a random variable  $u$ :

$$\hat{\Omega}(u) = \max(u) - \min(u) . \tag{3}$$

This estimator works best for abruptly bounded variables. When tails of the distribution are longer and less dense, as for platykurtic variables, the estimator may fail to give a good approximation of the support, as illustrated in Fig. 2(a),



(a) Essential points for the estimation of  $\Omega(\mathbf{y}_1)$  are missing: the source extraction failed

(b) Four points have been added at the borders of the source JPDF (located by the arrows); these 'artificial' points allow SWICA to extract the sources

**Fig. 2.** SWICA applied on super-Gaussian signals: scatter plots of the source signals (dots) and of the outputs (circle)

because there are not enough observations in ‘critical areas’ (the ‘corners of the square’ in the figure). In this case, SWICA may be completely misled: SWICA has minimized the support of  $y_1$ , but this output does not correspond to a source. Of course, if four well-chosen observations are available (see arrows in Fig. 2(b)), the problem disappears. This means that the SWM contrast may be very sensitive to a small number of observations. Fortunately, the distributions of the pixel intensities in an image are usually abruptly bounded, due to particular implementation choices (small encoding range) and image properties (because neighboring pixels often have similar values, there are usually few outliers).

## 4 Separation of Correlated Images

In this section, we show results of three ICA algorithms (FastICA [5], JADE [6] and SWICA) used to separate two correlated images (correlation: 26%). The source images are two gray-level landscapes (Fig. 3(a) and 3(d)); pixel intensities range from 0 to 255. A random linear mixing matrix and a decorrelation transformation are then applied; mixed images are shown in Fig. 3(b) and 3(e) (after translation and scaling to map the mixtures in the full range [0, 255], for readability purposes).

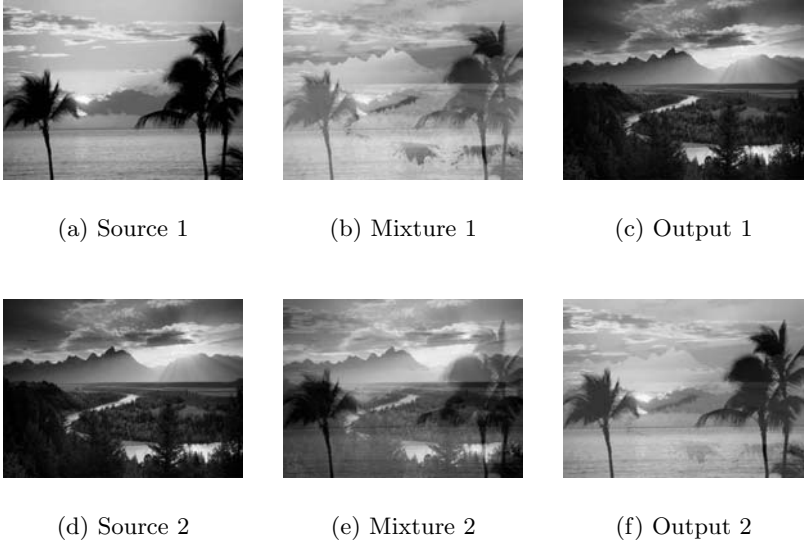
When looking at the scatter plot of the source images in Fig. 5(a), it becomes clear that they are not independent because the joint probability density function cannot be factorized (for instance, look at several horizontal – or vertical – conditional pdfs in the scatter plot: they are not equal to each other). As could be feared, both JADE and FastICA fails to recover the source images (see Fig. 4(b) and 4(c)).

In order to assess the extraction quality of the  $i$ -th source  $s_i$ , the ‘ $i$ -th performance index’ is defined as  $PI(i) \triangleq \frac{\sum_{j=1}^n \mathbf{c}_i(j)^2}{\max_j \mathbf{c}_i(j)^2} - 1$ ; a zero  $PI(i)$  indicates that  $y_i$  is proportional to a source, while a high  $PI(i)$  means that  $y_i$  results from the superimposition of several sources.

### 4.1 Extraction of the First Source

SWICA behaves rather differently than JADE and FastICA and recovers one of the sources (Fig. 4(d)). The output scatter plot is a parallelogram and two of its edges are parallel to the vertical axis: values of  $y_1$  computed by SWICA nearly equal those of  $s_1$ , since both marginal pdfs (initial sources and estimated ones) along the horizontal axis coincide. Unfortunately, the two other edges of the parallelogram are not parallel to the horizontal axis, meaning that  $y_2$  does not correspond to the second source.

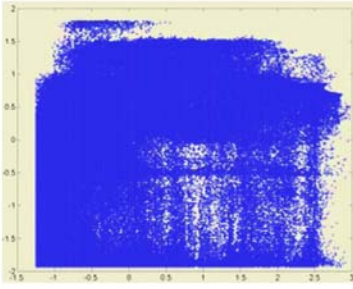
Understanding why both JADE and FastICA fail in this example is straightforward. Because source images are correlated, estimating independent sources amounts to extracting their common but independent components. In the case of two landscapes, these components are not the source images but new images (e.g. comp. 1 could account for the shared soil/sky contrast whereas comp. 2 could account for varying trees, mountain and clouds).



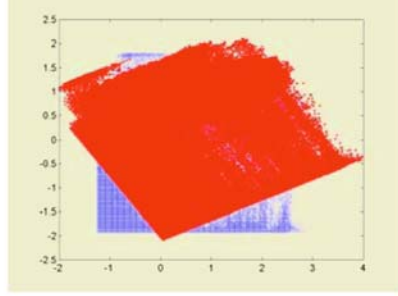
**Fig. 3.** Example of images separation using SWICA ( $\text{cov}=0.26$ ); source images (a,d), rescaled mixed images (b,e), rescaled extracted images (c,f)

Contrarily to other ICA contrasts, the SWM extracts a very limited piece of information out of the marginal pdf of the currently estimated source  $y_1$ : the bounds. In our image application, bounds are particularly interesting parts of the image distribution. Indeed, images may be assumed to involve three more or less important parts: (i) a global shared shape, (ii) local independent details and (iii) encoding techniques. The global shared shape leads to highly correlated and dense spots in the scatter plot. On the other hand, local independent details contributes to fill the scatter plot in a uniform but very sparse way. Finally, encoding technique generally produce saturation effects (towards full white and/or black), which are independent (source images are independently encoded). Usual ICA contrasts are especially sensitive to the global shape, which is dominating in correlated images, and thus try to make the images independent. On the other hand, SWM focuses on the bounds of the scatter plot: these bounds are generally well drawn due to parts (ii) and/or (iii) of the images and contains most of the independent features of the images. Of course, if parts (ii) and (iii) in the image are negligible, the edge of the scatter plots disappear and SWICA is likely to fail.

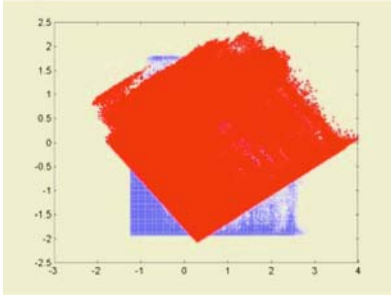
This explains why SWICA can match its first output  $y_1$  to one of the source (see Figure 1). But why does SWICA fail to recover the second source image? Actually, as for all ICA orthogonal contrasts, we whiten the mixtures beforehand and then constrain the unmixing matrix  $\mathbf{W}$  to be orthogonal. Unfortunately, this constraint is too restrictive in our case and amounts to recovering sources that are not correlated. More precisely, on one hand we know that



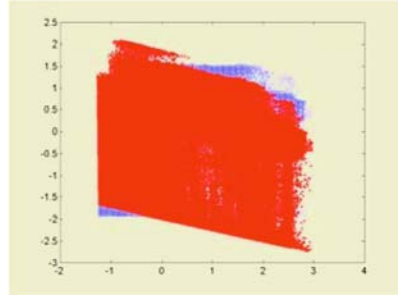
(a) Normalized sources



(b) FastICA ( $y_1$  vs  $y_2$ );  $PI(1) = 0.246$  and  $PI(2) = 0.449$



(c) Jade ( $y_1$  vs  $y_2$ );  $PI(1) = 0.497$  and  $PI(2) = 0.687$



(d) Swica ( $y_1$  vs  $y_2$ );  $PI(1) = 1.3 \times 10^{-4}$  and  $PI(2) = 0.063$

**Fig. 4.** Scatter plots between normalized images. In (b)-(d), the plots of  $y_1$  vs  $y_2$  (dark) are superimposed to the plot of the normalized sources (light gray)

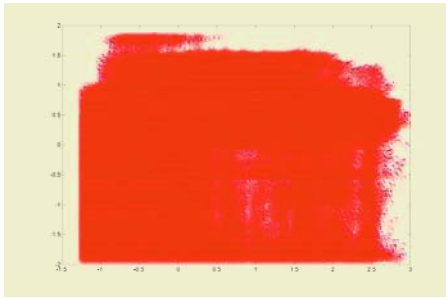
$E\{\mathbf{y}\mathbf{y}^T\} = \mathbf{W}E\{\mathbf{z}\mathbf{z}^T\}\mathbf{W}^T = \mathbf{W}\mathbf{W}^T = \mathbf{I}_n$  because of the whiteness property. On the other hand, we know that  $E\{\mathbf{s}\mathbf{s}^T\}$  is not diagonal, which is contradictory.

### 4.2 Extraction of the Second Source

The above-mentioned arguments explain why one source can be recovered, whereas the other ones cannot be recovered when the source images are correlated and  $\mathbf{W}$  is constrained to be orthogonal.

In the easy case where  $n = 2$ , the second line of the  $\mathbf{W}$  matrix given by SWICA must be modified. Therefore, we minimize  $\hat{\Omega}(y'_2) = \mathbf{w}'_2\mathbf{z}$ , where  $\mathbf{w}'_2$  is not constrained to be orthogonal to  $\mathbf{w}_1$  anymore. In order to avoid converging to  $y_1$ , we take  $\mathbf{w}_2$  as first guess for  $\mathbf{w}'_2$ . This procedure is applied only on the second output, without changing the first one, and allows separating the second source, as shown in Fig. 5. This procedure can be extended for a larger number





(a) Swica ( $y_1$  vs  $y_2'$ );  $PI(1) = 1.3 \times 10^{-4}$   
and  $PI(2) = 1.4 \times 10^{-4}$



(b) Second output  $y_2'$  after mapping to (0, 255)

**Fig. 5.** Results after extraction of the second source image: the low  $PI(\cdot)$  indicate that both sources are recovered correctly

of source images, by deriving a deflation algorithm to correct the bias due to the orthogonality constraint.

## 5 Conclusion and Future Work

In this paper, the SWM contrast is used to solve the ICA problem involving correlated images. This approach is motivated by the unsatisfying results of JADE and FastICA for the same problem, when no other preprocessing than whitening, like filtering is used.

Regarding the separation of two correlated images, SWICA succeeds in recovering one of the images but prewhitening of the mixtures settles on  $\mathbf{W}$  an orthogonality constraint that jeopardizes the retrieval of the second source image. To circumvent this problem, the orthogonality constraint is relaxed after the first deflation step. In this case, both source images are recovered correctly.

Future work will compare the proposed approach to MSD ICA (multiresolution subband decomposition), a method that preprocesses the image mixtures using filters. A second target is the development of a new version of SWICA, without orthogonality constraint.

## References

1. Vrins, F., Verleysen, M.: Swm : A class of convex contrasts for source separation. In: proceedings of ICASSP'05 (accepted), Philadelphia (USA) (2005)
2. Yang, H., Amari, S.I.: Adaptive on-line learning algorithms for blind separation-maximization entropy and minimum mutual information. *Neural Computation* **9** (1997) 1457–1482
3. Cichocki, A., Georgiev, P.: Blind separation algorithms with matrix constraints. *IEICE Trans. fundamentals* **E86-A** (2003) 522–531

4. Lee, J., Vrins, F., Verleysen, M.: A simple ica algorithm for non-differentiable contrasts. In: proceedings of EUSIPCO'05 (submitted), Antalya (Turkey) (2005)
5. Hyvärinen, A.: Fast and robust fixed-point algorithms for independent component analysis. *IEEE Trans. on Neural Networks* **10** (1999) 626–634
6. Cardoso, J.F.: Source separation using higher order moments. *Proceedings of the IEEE, Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP'89)* (1989, Glasgow, England) 2109–2112

# Robust Blind Image Watermarking with Independent Component Analysis: A Embedding Algorithm

Juan José Murillo-Fuentes and Rafael Boloix-Tortosa

Dept de teoria de la Señal y Comunicaciones, Universidad de Sevilla,  
Paseo de los Descubrimientos sn, Sevilla 41092, Spain\*

[murillo@esi.us.es](mailto:murillo@esi.us.es)

<http://viento.us.es/~murillo>

**Abstract.** The authors propose a new solution to the blind robust watermarking of digital images. In this approach we embed the watermark into the independent components of the image. Since independent components are related to the edges of the image, this method has a little perceptual impact on the watermarked image. Besides, we exploit the orthogonality of independent components and spread-spectrum generated watermarks in the blind extraction of the watermark. As extraction algorithm we use a simple matched filter. We also improve this novel method with standard techniques such as perceptual masking and holographic properties. Some experiments are included to illustrate the good performance of the algorithm against compression, cropping, filtering or quantization based attacks.

## 1 Introduction

Robust Watermarking (RW) of digital images [1] is one common solution to protect owners rights. It consists of embedding another signal or mark into the to be protected host image. We aim the watermark to be detected after severe attacks. In addition, our watermark is designed to be transparent to the user and we do not use the host image at detection, i.e., this is a invisible blind RW approach.

In some RW approaches we embed the mark in the spatial domain. On the other hand, we have methods working in a transform domain, such as the DCT or the DWT. ICA has been recently applied to digital watermarking following two main approaches. On the one hand, we have those approaches based on the mixture of the host image, or some transform domain coefficients, and the watermark [2, 3, 4]. In these methods, ICA is applied at detection to extract the watermark. On the other hand, based on the original results in [5], we have methods based on ICA as a transform domain where to embed the watermark

---

\* Thanks to Spanish government for funding TIC-2003-03781.

[6, 7]. The authors in [6] develop a non-blind approach focusing mainly in the detection stage by using non-linear techniques. In this paper we focus on simple techniques in the embedding stage [7] such as spread-spectrum watermark, perceptual masking and holographic methods, to greatly improve the blind method in [5].

## 2 ICA in Image Processing

### 2.1 Independent Component Analysis

Independent component analysis (ICA) [8] consists of projecting a set of components onto another statistically independent set. In the simple ICA, the  $l$  entries of a sample  $t$  of a column vector sequence  $\mathbf{x}_t$  are projected into a space of  $l$  components  $\mathbf{y}_t$  as statistically independent as possible. This projection is represented by an  $l \times l$  matrix  $\mathbf{B}$ .

$$\mathbf{y}_t = \mathbf{B}\mathbf{x}_t \quad (1)$$

Very much literature has been devoted to ICA algorithms. We will use here batch algorithms that minimize the marginal entropies ME of the outputs [9]. These algorithms have a good performance and are easy to use compared to gradient based methods.

### 2.2 Application to Image Processing

There are two common applications of ICA to image processing. On the one hand, we may assume we have  $l$  linear mixtures of  $l$  images. Therefore, we simply need to reshape each mixture of images into a vector and then apply ICA to separate them as in equation (1). The methods in [2, 3, 4] are based on this approach. On the other hand, we have other approaches where only one image is involved. These methods first decompose an image into components  $\mathbf{x}_t$  to later apply ICA [10]. Afterwards, any image processing technique may be applied to these, so computed, independent components (IC) [11]. Notice that each row of  $\mathbf{B}$  provides one independent component (an entry of vector  $\mathbf{y}_t$ ). Therefore, if we reshape each row of  $\mathbf{B}$  into a matrix, we obtain a set of 2-dimensional basis functions. These basis functions, also regarded as patches or features, are closely related to well-localized and oriented Gabor filters [8]. Some other authors suggest these basis functions to be the edges of the image [10], [12], or even to model the receptive fields of the primary visual cortical neurons [8]. An analysis of the image ICA components shows that many independent components are sparse distributed and that only some basis functions are needed to represent the image. Besides, the probability of independent components having small amplitudes is high, but large amplitudes occurs as well [8]. In [11] these features are used to compress or encode an image. Basic compression algorithms exploit these ideas as they retain only the independent components with larger energy. In addition, the authors in [11, 13] show that groups of images with similar features may be restored from a common set of basis (rows of matrix  $\mathbf{B}$ ). Particularly, the

projections  $\mathbf{B}$  computed for one image can be used in the processing of another one of the same class (text images, natural scenes,...).

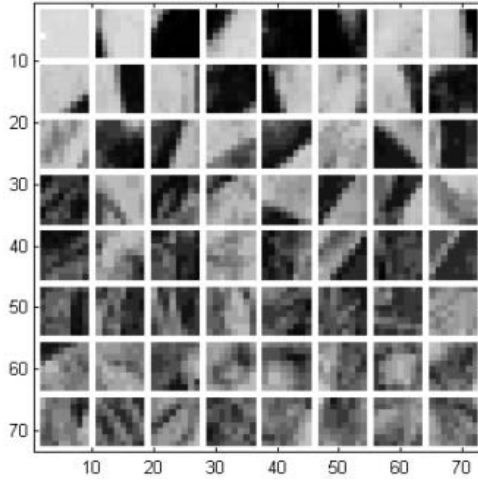
The IC of an image  $\mathbf{I}$  can be computed as follows. Assume matrix  $\mathbf{I}$  be a gray-scale image of size  $n \times m$ . This matrix can be divided into  $k \times k$  blocks or patches [8]  $\mathbf{C}_{p,q}$  to reshape them into vectors  $\mathbf{x}_t^I$  where  $t = (p - 1) \cdot m/k + q$ . The rows of  $\mathbf{x}_t^I$  may be then projected onto  $l = k^2$  independent components,

$$\mathbf{y}_t^I = \mathbf{B}\mathbf{x}_t^I \quad t = 1, \dots, mn/k^2 \quad (2)$$

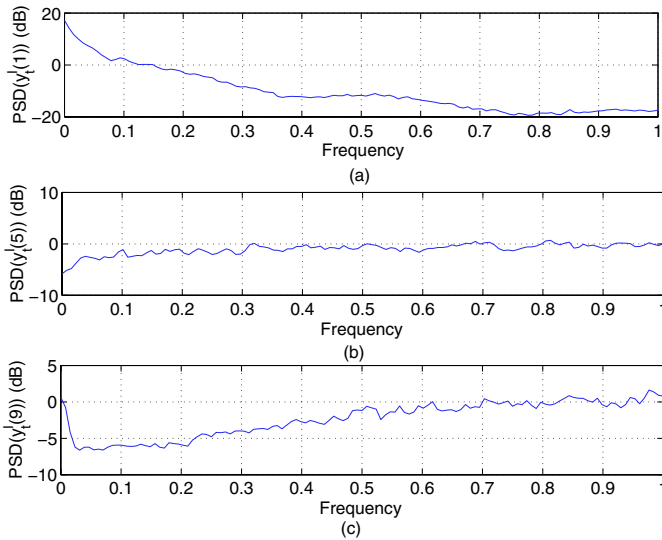
In Fig. 1 we show the spatial basis functions computed for  $k = 8$  and Lena Image ( $256 \times 256$ ). Each row matrix of the separating matrix  $\mathbf{B}$  was reshaped into a  $8 \times 8$  image. They were arranged row wise in descending order of energy, i.e., those basis functions (rows of  $\mathbf{B}$ ) providing independent components with larger variance are located at the top rows. The top-left corner basis allows to represent the DC component of every  $8 \times 8$  patch of the image. Notice also that the first rows are the basis functions to build the borders of the image. Besides, the last ones provide low energy components, details of the image.

### 3 Watermarking with ICA

Having the previous ideas in mind, we propose a new algorithm as follows. Firstly, the edges of the images are the candidate areas where to embed the watermark if we aim it to be imperceptible. Since some of the ICA basis functions are the edges of the images, by embedding the watermark in the associate IC we improve the invisibility of the mark. Hence, at a given PSNR, our watermark will be more imperceptible than the one embedded by using, e.g., DCT coefficients. Secondly, we may use a common public set of basis or rather use, following the ideas in [11, 13], our own private ICA projection. Hence, we fulfill one of the Kerckhoffs [14] principles, even if the attackers know we embedded the watermark using ICA they still do not know the exact projections. In addition, this privacy of the embedded watermark can be improved if we recall about IC tending to be sparse. In Fig. 2 we depict the power spectral density (PSD) for the independent components number 1 (Fig 1a) , 5 (Fig 1b) and 9 (Fig 1c) computed for the image of Lena with  $k = 3$ . As the IC have been arranged in descending order of energy, we have the DC component in Fig. 2a. IC number 5 in Fig. 2b has a white noise-like frequency response. Finally, the last IC has a high frequency response. In this paper we propose to use spread-spectrum watermark, with flat frequency response, to be added to the middle IC. This way we improve the robustness against any frequency based watermark filtering and most important, we ensure a low cross correlation between the IC and the watermark allowing blind detection by using a simple matched filter. Finally, the spread-spectrum watermark can be generated using a circular convolution with the bits of copyright information. This way we embed every bit into every pixel of the image, improving the robustness against some attacks such as cropping. We next describe the algorithm in detail.



**Fig. 1.** Example of basis functions for  $k = 8$



**Fig. 2.** Power spectral density for independent components number 1 (a), 5 (b) and 9 (c) of the Lena image for  $k=3$

### 3.1 Embedding

In Fig. 3 we include a scheme for a ICA based RW algorithm. We first will describe the basic steps of the embedding method to later discuss on its particular features. The embedding method yields

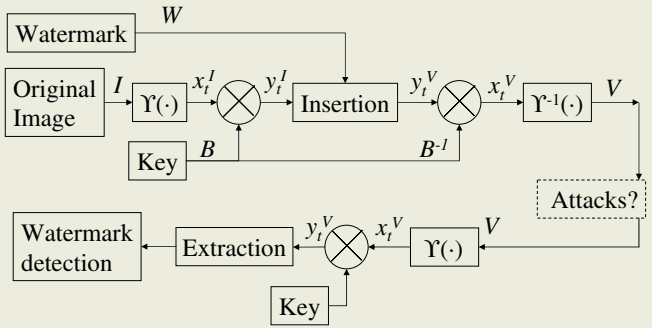


Fig. 3. ICA based robust watermarking algorithm

Algorithm 1: Embedding.

- 1 *Image to column vectors.* Compute the components  $\mathbf{x}_t^I$  of the  $n \times m$  host image  $\mathbf{I}$  using  $k \times k$  blocks.
- 2 *ICA components.* Compute its IC,  $\mathbf{y}_t^I = \mathbf{B}\mathbf{x}_t^I$ , using an ICA projection  $\mathbf{B}$ , the key of the insertion method.
- 3 *Insertion.* Compute the IC of the marked image,  $\mathbf{y}_t^V$ , by updating  $\mathbf{y}_t^I$  with the watermark,  $\mathbf{W}$ .
- 4 *Restoration.* Restore the (watermarked) image  $\mathbf{V}$  from components  $\mathbf{x}_t^V = \mathbf{B}^{-1}\mathbf{y}_t^V$ .

As already discussed, we propose the ICA projection  $\mathbf{B}$  computed for an image of the same kind as a private key. We use this matrix  $\mathbf{B}$  in the watermarking of a group of (e.g. same owner’s) images. Regarding the watermark, in [5] we used another image as watermark. In this paper we embed a spread spectrum mark, i.e., a message (the copyright information) “modulated” by means of spread spectrum techniques (SS) [1], hiding every bit of the message over the entire image (“holographic” property). Hence we endow the method with robustness against cropping and better synchronization properties [15]. We propose to design the watermark to have the size of one component,  $n/k \times m/k$ . This watermark is computed as the circular convolution of a key-dependent pseudorandom image  $\mathbf{P}$  and an image containing the bits of the message  $\mathbf{Q}$

$$\mathbf{W} = \mathbf{P} \otimes \mathbf{Q} \tag{3}$$

Let’s  $\mathbf{M}$  be a  $p \times p$  matrix whose pixels are the bits of the message. We define matrix  $\mathbf{Q}$  as follows

$$\mathbf{Q}(i, j) = \sum_{rs} M(r, s)\delta(i - r \cdot n_r/2, j - s \cdot n_c/2) \tag{4}$$

where  $n_r = n/(k \cdot p)$  and  $n_c = m/(k \cdot p)$ . Therefore, matrix  $\mathbf{Q}$  is a zero valued matrix except for the bits of the message, located at the center of each  $n_r \times n_c$  block. Once we have the watermark, we perform a perceptual masking based on

edge detection [16] to improve the invisibility of the watermark. Now we have the watermark ready to be embedded.

By arranging the IC components  $y_t(i)$ ,  $i = 1, \dots, k^2$  in descending order of magnitude we have the low frequency, the medium frequency and the very high-frequency coefficients, in this ordering. Next, we reshape the watermark into a row vector,  $y_t^W$ , and add it to the first  $r$  host image IC,  $\mathbf{y}_t^V$ . Similarly to other frequency transform watermarking algorithms [1], we propose to embed the watermark in the  $r$  most significant IC, excluding the first one. The first IC is important since it is the low pass component and it is the more robust one to compression. However, since it is not orthogonal to spread-spectrum watermarks we cannot easily blindly extract the mark by using simple detectors. We update the host IC as follows

$$\mathbf{y}_t^V(h) = y_t^I(h) + \alpha_h y_t^W \quad h = 2, \dots, r \quad (5)$$

where  $\alpha_h$  is a scaling factor to control the perception of the watermark. Other techniques such as the multiplicative or exponential approaches are possible [1]. In [5] we proposed a replacement of high-frequency components instead. However, these components can be easily removed by a simple, e.g., compression of the watermarked image.

### 3.2 Detection

In the detection of the watermark  $\mathbf{W}$  from the (attacked) watermarked image  $\mathbf{V}$  we go back on the steps of the embedding Algorithm 1, as illustrated in Fig. 3. The watermark detection yields

*Algorithm 2:* Detection.

- 1 *Watermarked image to column vectors.* Compute the components  $\mathbf{x}_t^V$  of the watermarked image  $\mathbf{V}$  by dividing it in  $k \times k$  patches.
- 2 *ICA components.* Compute the IC,  $\mathbf{y}_t^V$ , of the image as  $\mathbf{y}_t^V = \mathbf{B}\mathbf{x}_t^V$ .
- 3 *Extraction.* Extract the watermark from  $\mathbf{y}_t^V$ .
- 4 *Detection.* Estimate the message and the probability of watermark detection.

Since we use a SS watermark, detection can be easily achieved by simple correlation, i.e., by using a matched filter. We first compute the IC of the watermarked image  $\mathbf{y}_t^V = \mathbf{B}\mathbf{x}_t^V$ . Then we average all components  $h : \alpha_h \neq 0$ , improving the signal (watermark) to noise (image+attacks) ratio,

$$\hat{\mathbf{y}}_t^W = \sum_{h:\alpha_h \neq 0} \mathbf{y}_t^V(h) \quad (6)$$

and reshape the resulting vector into matrix  $\hat{\mathbf{W}}$ . Finally, we estimate the copyright message by computing matrix  $\mathbf{Q}$  in (4) as

$$\hat{\mathbf{Q}} = \mathbf{P} \otimes \hat{\mathbf{W}}^S \quad (7)$$



**Table 1.** Averaged bit error rate for different attacks and images performed on ICA and DCT watermarked images, PSNR=41 dB

| Attack                    | BER ICA |
|---------------------------|---------|
| AWGN                      | 0.0243  |
| Quantization $2^2$ levels | 0       |
| Median (5×5)              | 0.0104  |
| JPEG 20 %                 | 0.0747  |
| Cropping 10%              | 0.0278  |

where  $\mathcal{S}$  denotes symmetry ( $\mathbf{W}^{\mathcal{S}}(i, j) = \mathbf{W}(j, i)$ ). The values of the peaks of this convolution are the bits of the message. The probability of detection can be easily estimated by comparing these peaks to the rest of pixels modeled as zero mean Gaussian noise.

## 4 Experimental Results

We next include an example of robust watermarking applied to nine  $512 \times 512$  intensity images. We first computed  $\mathbf{x}_t^I$  with  $k = 3$  and then the IC of the image as  $\mathbf{y}_t^I = \mathbf{B}\mathbf{x}_t^I$ , where matrix  $\mathbf{B}$  was the one obtained for another image. The watermark was generated as the spread version of a 2-dimensional message of  $8 \times 8$  bits. The watermark was added to the IC of the image number  $h = 2, 3, 4, 5, 6$ . The final peak signal-to-noise ratio (PSNR) was 41 dB. At this point we must emphasize that for 41 dB the visual perception of the watermark in the DCT approach was much more significant than in the ICA method. Besides, it as been further improved by means of a perceptual mask.

We performed the following attacks and obtain the averaged bit error rate (BER) included in Tab. 1. We first added white Gaussian noise with standard deviation  $\sigma = 0.15$ , we requantized the image to  $2^2$  levels, then we applied a  $3 \times 3$  median filter, the image was also JPEG compressed to 20% of its original size and finally we cropped the 90% of the image.

## 5 Conclusions

In this paper we propose a new blind robust image watermarking algorithm, where we embed the watermark into the independent components of the image. The orthogonality between spread spectrum signals and IC is exploited in this novel blind design. This method has a little perceptual impact compared to the DCT approach. Besides, we propose a double-key algorithm, where the ICA projection and the spreading codes are needed in the retrieval of the watermark. We show that by introducing some useful and simple features in the embedding

stage we greatly improve the performance of the method. These are the use of a perceptual mask and the holographic approach. In the experiments included we illustrate the good performance of this method.

## References

1. Cox, I., Kilian, J., Leighton, T., Shamoon, T.: Secure spread spectrum watermarking for multimedia. *IEEE Transactions on Image Processing* **6** (1997) 1673–1687
2. Noel, S.E., Szu, H.H.: Multimedia authenticity with ICA watermarks. In: *Proc. SPIE Vol. 4056*, p. 175–184, *Wavelet Applications VII*, Harold H. Szu; Martin Vetterli; William J. Campbell; James R. Buss; Eds. (2000) 175–184
3. Yu, D., Sattar, F.: A new blind watermarking technique based on independent component analysis. In: *IWDW. (2002)* 51–63
4. Liu, J., Zhang, X., Sun, J., Lagunas, M.A.: A digital watermarking scheme based on ICA detection. In: *Proc. ICA2003*, Nara, Japan (2003) 215–220
5. Murillo-Fuentes, J., Molina-Bulla, H., González-Serrano, F.: Independent component analysis applied to digital image watermarking. In: *Proc. ICASSP'01. Volume III.*, Salt Lake City, USA (2001) 1997–2000
6. Bounkong, S., Toch, B., Saad, D., Lowe, D.: Ica for watermarking digital images. *J. Mach. Learn. Res.* **4** (2003) 1471–1498
7. Murillo-Fuentes, J.J.: Independent component analysis in the watermarking of digital images. In: *Proc. ICA2004*, Granada, Spain (2003)
8. Hyvriinen, A., Karhunen, J., Oja, E.: *Independent component analysis*. John Wiley and Sons (2001)
9. Murillo-Fuentes, J.J., Gonzalez-Serrano, F.J.: A sinusoidal contrast function for the blind separation of statistically independent sources. *IEEE Trans. on Signal Processing.* **52** (2004) 3459–3463
10. Bell, A.J., Sejnowski, T.J.: Edges are the independent components of natural scenes. In *Mozer, M.C., Jordan, M.I., Petsche, T., eds.: Advances in Neural Information Processing Systems. Volume 9.*, The MIT Press (1997) 831
11. Lee, T., Lewicki, M., Sejnowski, T.: Unsupervised classification with non-gaussian mixture models using ICA. In: *Advances in Neural Information Processing Systems. Volume 11.*, Cambridge, MA, The MIT Press (1999) 58–64
12. S Hornillo-Mellado, R Martn-Clemente, J.I.A., Puntonet, C.G.: Application of independent component analysis to edge detection and watermarking. In: *Proc. IWANN'03*, Mahn, Spain (2003) 270–280
13. Bugallo, M.F., Dapena, A., Castedo, L.: Image compression via independent component analysis. In: *Learning*, Leganés (2000)
14. Kerckhoffs, A.: La cryptographie militaire. *Journal des Sciences Militaires*, IX:5–38 (Jan), 161–191 (Feb) (1883)
15. Mora-Jimenez, I., Navia-Vazquez, A.: A new spread spectrum watermarking method with self-synchronization capabilities. In: *Proc. ICIP2000*, Vancouver, BC, Canada (2000)
16. Wolfgang, R., Podilchuk, C., Delp, E.J.: Perceptual watermarks for digital images and video. *Proceedings of the IEEE* **87** (1999) 1108–1126

# Adaptive Load Frequency Control with Dynamic Fuzzy Networks in Power Systems

Yusuf Oysal<sup>1</sup>, Ahmet Serdar Yilmaz<sup>2</sup>, and Etem Koklukaya<sup>3</sup>

<sup>1</sup> Dept. of Computer Engineering, Anadolu University, Eskisehir, Turkey  
yoysal@anadolu.edu.tr

<sup>2</sup> Dept. of Electrical and Electronics Eng. Kahramanmaraş Sutcu Imam University,  
K.Maras, Turkey  
asyilmaz@ksu.edu.tr

<sup>3</sup> Dept. of Electrical and Electronics Eng., Sakarya University, Sakarya, Turkey  
ekaya@sakarya.edu.tr

**Abstract.** This paper proposes a new controller based on neural network and fuzzy logic technologies for load frequency control to allow for the incorporation of both heuristics and deep knowledge to exploit the best characteristics of each. A “Dynamical Fuzzy Network (DFN)” that contains dynamical elements such as delayers or integrators in their processing units is used in the adaptive controller design for load frequency control. A DFN is connected between the two area power systems. The input signals of the DFN are the ACEs and their changes. The outputs of the DFN are the control signals for the two area load frequency control. Adaptation is based on adjusting parameters of DFN for load frequency control. This is done by minimizing the cost functional of load frequency errors. The cost gradients with respect to the network parameters are calculated by adjoint sensitivity. In this paper, it is illustrated that this control approach is more successful than conventional integral controller for load frequency control in two area systems.

## 1 Introduction

Load frequency control (LFC) is the one of major requirements providing reliable and quality operation in multi-area power systems. For satisfactory operation, constant frequency and active power balance must be provided. As frequency is a common factor throughout the systems, any change in active power demand/generation at power systems is reflected throughout the system by a change in frequency.

Two important technologies, which provide good results in load frequency control in power system, are neural networks and fuzzy logic control/decision systems that are excellent at developing human-made systems that can perform the same type of information processing that our brain performs. In the literature many neural networks and fuzzy logic investigations for load frequency control in power systems have been presented. In [1], fuzzy logic is used for gain scheduling of PI controllers to control the area load frequency. The control scheme adopts a formulation for the area control error, which always guarantees zero steady state time error and inadvertent inter-

change. This study was developed in [2] based on the refined genetic algorithm (RGA) to tune the membership functions and rule sets for the fuzzy control of a two-area interconnected power system. The work [3] used a two-layer fuzzy controller for efficient load frequency control. The first one is the pre-compensator, to generate and update the reference value of area control error (ACE), while the second one is a feedback fuzzy controller, to make the ACE decay to zero at steady state. The study [4] aims to develop a nonlinear neural network controller to control the deviations in load frequency of a power system and to overcome the problems of simple neural networks such as long training times and requirements large number of neurons.

This paper proposes a new controller based on neural network and fuzzy logic technologies for load frequency control to overcome these drawbacks and may also allow for the incorporation of both heuristics and deep knowledge to exploit the best characteristics of each. A “Dynamical Fuzzy Network (DFN)” [5-7] that contains dynamical elements such as delays or integrators in their processing units is used in the adaptive controller design for load frequency control.

## 2 Two Area Load Frequency Control

In interconnected power system with two or more areas, the generation within each area has to be controlled so as to maintain scheduled power interchange [8]. Load frequency control scheme have to be on two main control actions: i) Primary control and ii) Secondary control.

Primary control is achieved by turbine-governing system. In this level, only active power is balanced and steady state frequency errors can occur. In multi-area power systems, frequency must be equal and must be hold a rated value at all areas. So, the second control action is realized in large power systems that include two or more areas. Active power is controlled at the tie line between neighbor areas. Especially, large power plants at these areas join to frequency control. Nowadays computer-aided controllers realize this action. Fig.1 shows primary and secondary control action.

Simulated system consists of an interconnection of two power areas. For the simulations, linearized mathematical model given in Fig.2 is used. This model includes classical integral secondary controller and its parameters are given in Appendix.

## 3 The Architecture of a Dynamic Fuzzy Network

DFN model has been used in the meaning of a network that has unconstrained connectivity and has dynamical elements in the fuzzy processing units. The processing unit in the network is called “feuron” [6,7]. The feuron represents a single dynamic neuron with fuzzy activation function. The dynamic feuron resembles the biological neuron model. This model fires if the inputs of feurons are excited enough. The firing procedure is done through a lag dynamics such as Hopfield dynamics. Fuzzy activation function  $\phi$  behaviors as biological neurons which have receptive field units in the visual cortex, in part of the cerebral cortex and in outer parts of the brain [9-11].



$$\dot{x}_i = f_i(x_i, p) = \frac{1}{T_i} [-x_i + \sum_{j=1}^n w_{ij}y_j + \sum_{j=1}^L p_{ij}u_j + b_i]; x_i(0) = x_{i0}, i = 1, 2, \dots, n \quad (3)$$

In general, there are L input signals, which can be time-varying, n dynamic feuron units, n bias terms, and M output signals. The units have dynamics associated with them, and they receive input from themselves, bias term and from all other units. The output of a unit  $y_i$  is a fuzzy activation function  $\phi(x_i)$  of a state variable  $x_i$  associated with the unit. The output of the overall network is a linear weighted sum of the unit outputs. The bias term  $b_i$  is added to the unit inputs.  $p_{ij}$  is the input connection weights from  $j^{th}$  input to  $i^{th}$  feuron,  $w_{ij}$  is the interconnection weights from  $j^{th}$  feuron to  $i^{th}$  feuron and  $q_{ij}$  is the output connection weights from  $j^{th}$  feuron to  $i^{th}$  output.  $T_i$  is the dynamic constant of  $i^{th}$  feuron and  $b_i$  is the bias (or polarization) term of  $i^{th}$  feuron.  $\pi$  is the parameters of fuzzy activation function which are centers  $c$ , spreads  $\sigma$  and output centers  $a$ . The initial conditions on the state variables  $x_i(0)$  must be specified.

This model (DFN) approximates physical dynamic nonlinear systems [6,7]. An example of DFN open diagram with two-input/two-output and two-feuron is shown in Fig.3.

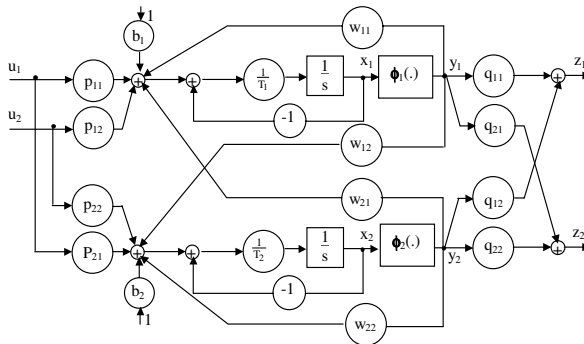


Fig. 3. The state diagram of a DFN with two-feuron two-inputs/two-outputs

### 4 Adaptive Load Frequency Control

Adaptive load frequency control is done by a DFN connected between two area power systems as seen in Fig.4. The input signals to the DFN are the ACEs and their changes. The outputs of the DFN model are the control signals for the two area load frequency control. Adaptation is based on the DFN training that is adjusting parameters of DFN for load frequency control in power systems with Broyden-Fletcher-Golfarb-Shanno (BFGS) gradient algorithm [12]. The cost gradients with respect to the network parameters are required for this algorithm. The focus in this paper has been adjoint sensitivity analysis for calculating the cost gradients with respect to all DFN parameters.

As a performance index or cost structure is selected in the simple quadratic form as follows:

$$E = \frac{1}{2} \int_0^t [z(t) - z^d(t)]^T [z(t) - z^d(t)] dt \tag{4}$$

where  $e(t)=z(t)-z^d(t)$  is error function for the load frequencies of the power system.  $z(t)$  is DFN model response (output), and  $z^d(t)$  is desired load frequency responses. We need to compute the cost sensitivities for various parameters such as:

$$\frac{\partial E}{\partial w}, \frac{\partial E}{\partial p}, \frac{\partial E}{\partial q}, \frac{\partial E}{\partial T}, \frac{\partial E}{\partial b}, \frac{\partial E}{\partial c}, \frac{\partial E}{\partial \sigma}, \frac{\partial E}{\partial a} \tag{5}$$

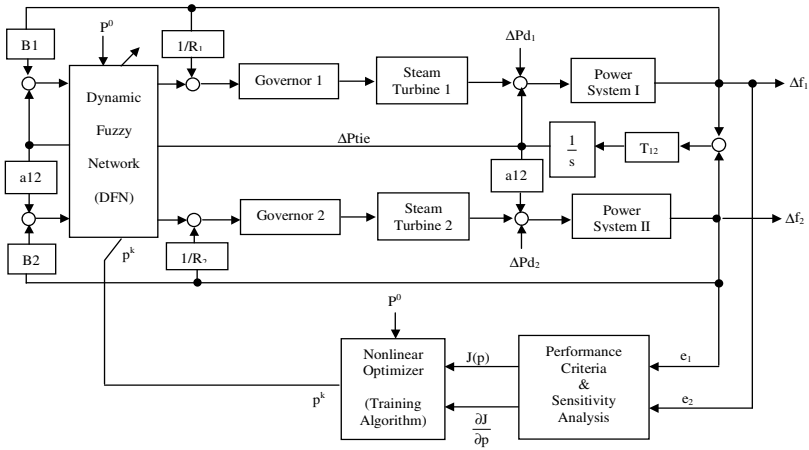


Fig. 4. Block diagram of adaptive LFC of two area power system with DFN model

“Adjoint” method is used for sensitivity computation [5-7,13,14] based on a new dynamical system defined with adjoint state variables  $\lambda_i$  :

$$-\dot{\lambda}_i = -\frac{1}{T_i} \lambda_i + \frac{1}{T_i} \sum_j w_{ij} y'_j \lambda_j + e_i(t) \sum_j q_{ij} y'_j, \quad \lambda_i(t_f) = 0 \tag{6}$$

$$y'_j = \frac{\partial \phi_j(x_j)}{\partial x_j} = \frac{\sum_{k=1}^{R_j} (\phi_j - a_{jk}) \exp\left(-\frac{1}{2} \left(\frac{x_j - c_{jk}}{\sigma_{jk}}\right)^2\right) \left(\frac{x_j - c_{jk}}{\sigma_{jk}^2}\right)}{\sum_{i=1}^{R_j} \exp\left(-\frac{1}{2} \left(\frac{x_j - c_{jk}}{\sigma_{jk}}\right)^2\right)} \tag{7}$$

The size of adjoint vector is  $n$  and is independent of network parameters. There are  $n$  quadratures for computing the sensitivities. The integration of the differential equations must be performed backwards in time, from  $t_f$  to 0. Let  $p$  be a vector containing all network parameters. Then, the cost gradients with respect to parameters are given by the following quadratures:

$$\frac{\partial E}{\partial p} = \int_0^{t_f} \left( \frac{\partial f}{\partial p} \right)^T \lambda dt \tag{8}$$

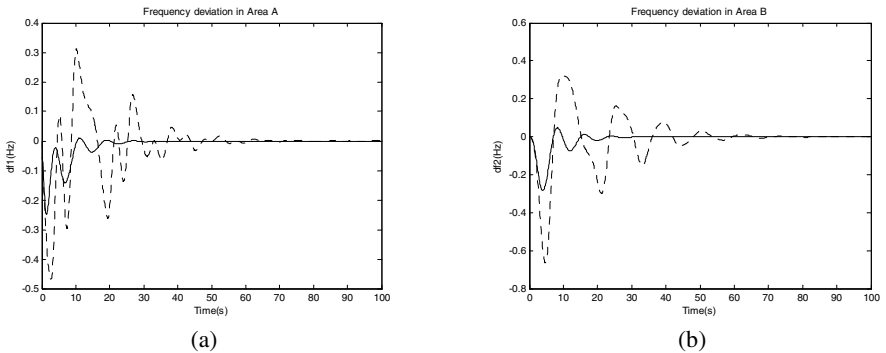
### 5 Simulation Results

In simulation examples a DFN with two feurons (Fig.3) were trained for load frequency control. In this level, a step disturbance of 5% is applied to system power input of linearized block diagram shown in Fig.4. Under these conditions, DFN was trained offline and interconnection parameters and time constants of the DFN were obtained as:

$$w = \begin{bmatrix} 0.01 & 1.1 \\ 0.02 & 0.11 \end{bmatrix}, p = \begin{bmatrix} -2.03 & 0 \\ 0 & -1.01 \end{bmatrix}, q = \begin{bmatrix} 0.01 & 0.946 \\ 0.01 & -0.55 \end{bmatrix}, r = \begin{bmatrix} -7.07 \\ 0.103 \end{bmatrix}, T = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

During the disturbances, frequency variations of all areas and changing in power flow at tie line for conventional and proposed controller after training DFN are illustrated in Fig.5 and Fig.6, respectively.

In the second case, parameter adjustment of the DFN was done online. To show the efficiency of this approach, we only update the parameter value of  $q_{12}$  in the simulations for the case of 10% increase in input power of area A. After training, the value of  $q_{12}$  is increase form 0.946 to 1.034. As we see from Fig.7, the control application is again successful.



**Fig. 5.** Frequency Change in a) Area A b) Area B (5% step change, DFN solid line, Integral dotted line)



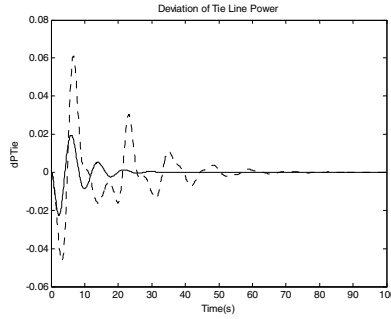


Fig. 6. Deviation of Tie Line Power (5% step change, DFN solid line, Integral dotted line)

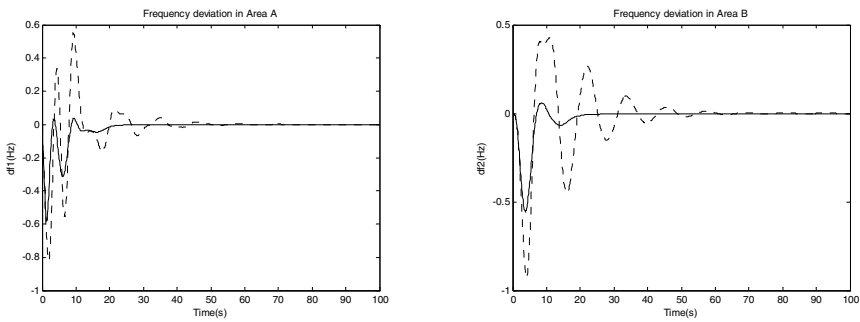


Fig. 7. Frequency Changes in a) Area A, b) Area B (10% step change, DFN solid line, Integral dotted line)

## 6 Conclusion

An alternative and successful controller for load frequency control is proposed. Simulation results prove the robustness and effectiveness of the proposed adaptive LFC scheme over only the conventional integral control system due to fast response with less overshoot and/or undershoot. It is also seen from the simulations that, the proposed controller causes less frequency drop than integral controller and oscillations in frequency rapidly damp out. Proposed networks can be easily trained and adapted to different disturbance cases online. There is one other important motivation for investigating dynamic fuzzy networks for load frequency control in power systems is that they suggest a simple hardware implementation. It is the simple and plausible VLSI implementation for a continuous-time dynamic fuzzy network [10,15].

## References

1. Chang, C.S., Weihui, F.: Area Load Frequency Control Using Fuzzy Gain Scheduling of PI Controllers. *Electric Power System Research*, Vol. 42 (1997) 145-152
2. Chang, C.S., Weihui, F., Fushuan W.: Load Frequency Control Using Genetic Algorithm Based Fuzzy Gain Scheduling of PI Controllers. *Electric Machines and Power Systems*, Vol. 26 (1998) 39-52

3. El-Sherbiny, M.K., El-Saady, G., Yousef, A.M.: Efficient Fuzzy Logic Load Frequency Controller. *Energy Conversion and Management*, Vol. 43 (2002) 1853-1863
4. Chaturvedi, D.K., Satsangi, P.S., Kalra, P.K.: Load Frequency Control: A Generalized Neural Network Approach. *Int. Journal of Electrical Power Systems*, Vol. 21 (1999) 405-415
5. Konar, A.F., Becerikli, Y., Samad T.: Trajectory Tracking with Dynamic Neural Networks, *Proc. of the 1997 IEEE Int. Sym. on Intelligent Control (ISIC'97)*, Istanbul/Turkey (1997)
6. Oysal, Y., Feuro Modeling and Optimal Fuzzy Control, Ph.D. Thesis, Sakarya University, Sakarya/Turkey, (2002)
7. Becerikli, Y., Oysal, Y., Konar, A.F.: Trajectory Priming With Dynamic Fuzzy Networks In Nonlinear Optimal Control. *IEEE Transaction on Neural Networks*, Vol. 15 (2004) 383- 394
8. Kundur, P.: *Power System Stability and Control*. IEEE Press Mc Graw Hill (1994)
9. Hopfield, J.J.: Neural Networks and Physical Systems with Emergent Collective Computational Abilities. *Proc. of the National Academy of Sciences*, Vol. 79 (1982) 2554-2558
10. Hopfield, J.J.: Neurons with Graded Response Have Collective Computational Properties Like Those of Two-State Neurons. *Proc. of the National Academy of Sciences*, Vol. 81 (1984) 3088-3092
11. Passino, K.M., Yurkovich, S.: *Fuzzy Control*. Addison-Wesley, MenloPark, Cal. (1998)
12. Edgar, T.F., Himmelblau, D.M.: *Optimization of Chemical Processes*, McGraw-Hill (1988)
13. Becerikli, Y., Konar, A.F., Samad, T.: Intelligent Optimal Control with Dynamic Neural Networks. *Neural Networks*, Vol. 16 (2003) 251-259
14. Konar, A.F., Samad, T.: *Dynamic Neural Networks*. Technical Report SSDC-92-I 4152-2, Honeywell Technology Center, 3660 Technology Drive, Minneapolis, MN 55418 (1992)
15. Anderson, J.A., Rosenfeld: *Neurocomputing: Foundation of Research*, Cambridge, MA: MIT Press (1988)

## Appendix

### Mathematical Model for Turbine-Governor Systems

$$G_{TUR}(s) = \frac{1}{1+sT_T} \frac{1+sK_T.T_{TT}}{1+sT_{TT}} \quad (A-1)$$

$$G_{GOV}(s) = \frac{1}{1+sT_G} \quad (A-2)$$

For simulated systems, time constants are as follows :

$T_{T1}=T_{T2}=0.3$  s,  $T_{TT1}=T_{TT2}=20$  s,  $K_{T1}=K_{T2}=0.3335$ ,  $T_{G1}=T_{G2}=0.2$  s,  $R_1=R_2=2.43$ ,  $B_1=B_2=0.425$ ,  $K_{P1}=K_{P2}=120$ ,  $T_{P1}=T_{P2}=20$ s.

# Fault Fuzzy Rule Extraction from AC Motors by Neuro-fuzzy Models

G.I. Sainz\*, R. García, and M.J. Fuente

Department of Systems Engineering and Control, School of Industrial Engineering,  
University of Valladolid, Paseo del Cauce s/n,  
47011 Valladolid, Spain  
Phone: +34 83 423000 Ext. 4401  
gresai@eis.uva.es

**Abstract.** In this paper the knowledge extraction from neural and fuzzy models, and the quality and the explanation capacities of this knowledge are tracked. Nowadays the application of algorithms and methodologies based on artificial neural networks and fuzzy logic are very usual in most of the scientific and technical areas in order to generate models driven by data. But sometimes to obtain a good model by these techniques is not enough when some explanations about the model behaviour are mandatory, and these models are very near, most of the cases, to "black boxes" or its explanatory capacity is very poor. In literature, several methods are been published in order to extraction, simplification and interpretability of the knowledge stored in these types of models.

In this paper a real problem is involved: to model an AC motor on several functioning modes (faults) by several neural/fuzzy approaches, making a comparison on the knowledge extracted from each one: Feed-forward network + Backpropagation, Subtractive Clustering + ANFIS, FasArt and FasBack.

## 1 Introduction

The subject of this paper is to study the knowledge extracted from some networks and neuro-fuzzy systems by a set of fuzzy rules about fault modes of an AC motor, that have been learnt by these systems in the learning stage. In accordance with this, the motor behaviours have been stored in the structure and weights set of each neuro/fuzzy system employed. If this information is extracted and interpreted then the neural networks would not be considered as black boxes because they could explain that has been learnt, discovering in a "friendly" way the information structure hidden in the data.

If a compact and interpretable fuzzy rule set was possible from data then the application of softcomputing techniques on several problems will be improved,

---

\* Corresponding author. This work has been supported by the national research agency of Spain (CICYT) through the project DPI2003-09373.

even will be factible. A process from a numeric level to a symbolic level is involved using neuro-fuzzy systems and a knowledge extraction is made.

This process can involve two tasks: extraction and refine on the fuzzy rules. In literature there are a lot of approaches for both tasks. The refinement is very relevant when “fuzzy modelling” is involved due to this approach permits to reach a good accurate models but their transparency and interpretability are very low, opposite to the “linguistic modelling”.

In this paper several known artificial, neuro and neuro-fuzzy networks have been used to studying the quality of their learnt and extracted knowledge by themselves, when they have been applied to the functioning modes of a real AC motor:

- Subtractive Clustering + ANFIS (Chiu, 1994; Jang, 1993), this system works as a Sugeno fuzzy system.
- Feedforward network (Haykin, 1999), that is the well-known multilayer perceptron training with backpropagation algorithm (Castro *et al.*, 2002).
- FasArt (Cano Izquierdo *et al.*, 2001; Sainz Palmero *et al.*, 2000) and FasBack (Cano Izquierdo *et al.*, 2001), that can be considered similar to a fuzzy logic system with fuzzification by simple point, inference by product and defuzzification by average of fuzzy set centers.

These models can be interpreted as fuzzy logic system but in the feedforward case, here the method proposed in (Castro *et al.*, 2002) has been used in order to obtain a fuzzy rule base.

The paper is organized as follows: first a introduction to knowledge extraction and its main concepts is made. Then the description of the AC motor plant is introduced and the main results obtained are discussed. Finally, the most relevant conclusions obtained from this work are set out.

## 2 Knowledge Extraction

One of the most interesting, and in some cases mandatory, aspects of the neural networks is the extraction, refinement and interpretation of the knowledge learnt and stored during the training stage by the neural network. In (Mitra and Hayashi, 2000) and (Andrews *et al.*, 1995) a survey about the rule extraction methods from neural networks and neuro-fuzzy systems is made.

The term rule generation usually includes *rule extraction + rule refinement*:

- Rule extraction, the way or algorithm to get some expression type about the knowledge from the neural networks using the information about them: weight sets, units, links, etc...
- Rule refinement, this extracted knowledge can be very rough, containing redundancies, incoherencies, etc. so a ”deputation” process has to be considered.

Some popular criteria can be taking in account to evaluate this type of methods (Taha and Ghosh, 1999):

- Granularity, details about "taking decisions" of the system.
- Comprehensiveness: amount of embedded knowledge captured by the rules.
- Compressibility: number of rules and number of premises in each extracted rule.
- Transparency of the extracted rules: how well the decisions or conclusions are explained.
- Portability of the extraction method.
- Modifiability, updating of the rules from a retrained network.
- Refinement capability.
- Stability and robustness.
- Complexity and scalability: computational issues.

The interpretability of the rules is based on the completeness and distinguishability that permit to assign a clear meaning to each fuzzy set (Jin and Sendhoff, 2003). These two conditions are expressed in:

$$\gamma_1 < \textit{Similarity}(A_i, A_{i+1}) < \gamma_2 \quad (1)$$

if  $\gamma_1$  is small but no zero then the completeness is guarantee, and if  $\gamma_2$  is different enough than 1 then a good distinguishability is reached.

In general most of the methods are focused on particular networks and problems. Here, the extraction of rules from the considered neuro/fuzzy systems have been made by descompositional (Mitra and Hayashi, 2000) approach, and the criteria have been:

- Quality: accuracy, rule number, etc.
- Interpretability and expressive power
- Refinement capability
- Method complexity.

### 3 Extraction Fault Rules from AC Motor Models

The AC motor has been modelled on several functioning modes in order to generate some rule sets, fuzzy rules mainly, describing these motor modes by a comprehensible and interpretable way. The above mentioned neuro/fuzzy systems have been used to make this propose and to test the capacities from each.

**Laboratory Plant.** The motor laboratory plant is composed by two Leroy-Somer motors of induction with threes phases, 4 poles, 28 rotor bars and 36 stator slots and a power of 5.5 Kw. The power supply frequency is 50 Hz and a delta connection is employed in the motor. Several sensors have been applied to each motor in order to achieve as fine monitoring as possible: voltage, current, temperature, magnetic flux, optical encoders, etc.

**Table 1.** RMS Error and rule number from each neural/fuzzy system and each AC motor mode

| Mode | S.C. + ANFIS |          | FasArt |          | FasBack |          | Backp. |          |
|------|--------------|----------|--------|----------|---------|----------|--------|----------|
|      | Error        | N. Rules | Error  | N. Rules | Error   | N. Rules | Error  | N. Rules |
| NF   | 0.00         | 7        | 0.01   | 9        | 0.01    | 7        | 0.01   | 6        |
| RV1  | 0.01         | 5        | 0.01   | 4        | 0.01    | 4        | 0.03   | 5        |
| RV2  | 0.01         | 5        | 0.01   | 5        | 0.02    | 5        | 0.04   | 5        |
| RV3  | 0.01         | 5        | 0.01   | 7        | 0.01    | 7        | 0.03   | 5        |
| VS1  | 0.00         | 12       | 0.02   | 6        | 0.04    | 4        | 0.42   | 5        |
| VS2  | 0.02         | 4        | 0.01   | 4        | 0.02    | 4        | 0.36   | 5        |
| VS3  | 0.01         | 4        | 0.02   | 4        | 0.02    | 8        | 0.31   | 5        |
| CS1  | 0.00         | 6        | 0.03   | 4        | 0.03    | 4        | 0.14   | 5        |
| CS2  | 0.00         | 13       | 0.02   | 4        | 0.03    | 6        | 0.06   | 5        |
| CS3  | 0.00         | 10       | 0.02   | 4        | 0.03    | 4        | 0.04   | 5        |
| PS1  | 0.00         | 12       | 0.02   | 6        | 0.04    | 7        | 0.08   | 5        |
| PS2  | 0.01         | 5        | 0.03   | 8        | 0.03    | 9        | 0.20   | 5        |
| PS3  | 0.01         | 8        | 0.05   | 8        | 0.10    | 9        | 0.04   | 5        |

The experiments of this work have involved only non destructive faults due to economic costs of the destructive ones. Faults in the plant and faults in the sensors have been considered, in total 13 types of faults have been generated:

- Plant:
  - Normal functioning (NF).
  - Important unbalanced power supply in each of the three phases (PS1, PS2, PS3).
  - Resistor stator variation ( $\Delta$ ) (RV1, RV2, RV3).
- Sensors:
  - Fault in the angular speed encoder.
  - Fault in voltage sensor in each phase (VS1, VS2, VS3).
  - Fault in current sensor in each phase (CS1, CS2, CS3).

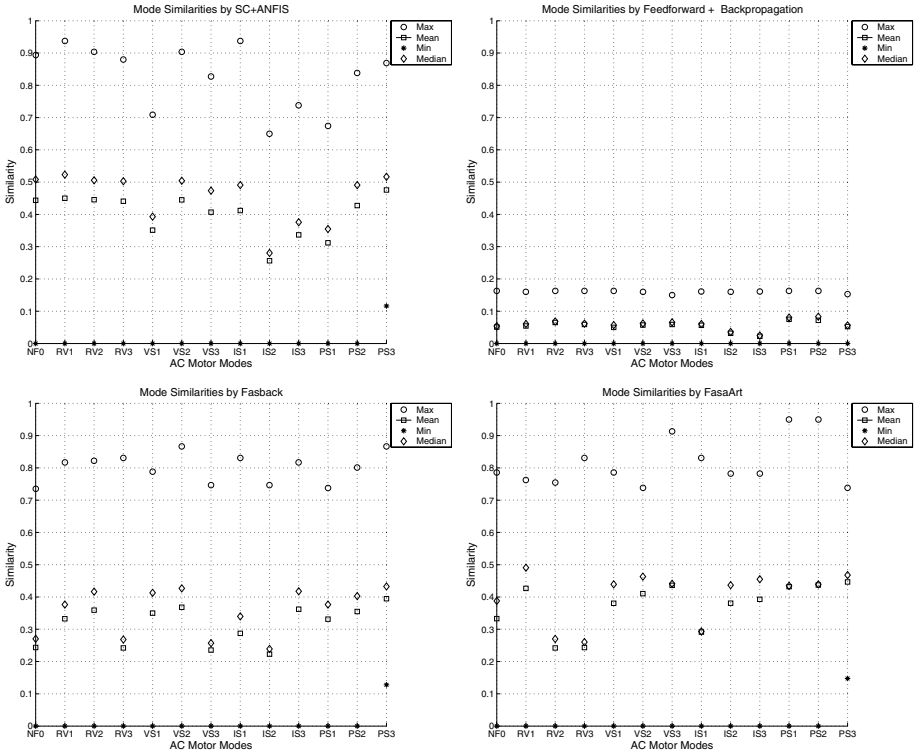
The input variables employed are: the 3 phase currents ( $I_1, I_2, I_3$ ), the 3 phase power ( $V_1, V_2, V_3$ ) and the out power (torque). These variables were chosen due to be the most direct, simple and known features so more intuitive ones.

The data were acquired by a data card whose sampling frequency can get until 250 KHz. In each data set contains 35000-40000 data vector of the selected variables, and for each fault o functioning type 2-4 data sets were recorded.

### 3.1 Experimental Results

In order to make the experiments each involved neural/fuzzy system was training with data sets containing only information about each motor mode involved.

The best model for each motor mode was chosen by trial and error, varying the system parameters and using the root mean square error (RMS) of the output.

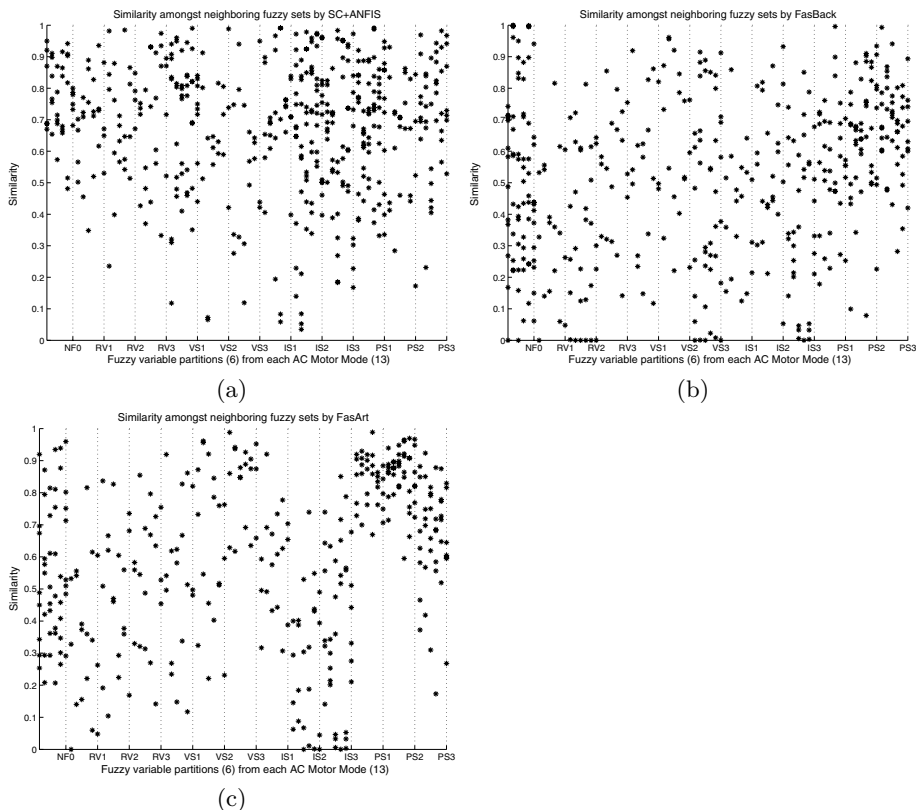


**Fig. 1.** Similarity values (maximum, minimum, mean a median) for each rule antecedent

The first case is Subtractive Clustering and ANFIS, here the only tuning parameter is the clustering rate:  $0.1 \leq r \leq 1$ . In this case, the hard work is made by Subtractive Clustering, being ANFIS a refinement method with poor improvement. When FasArt and FasBack were training the range for their tuning parameters was:  $0.1 \leq \rho \leq 1$  and  $1 \leq \gamma \leq 20$ . Finally, in the feedforward network with backpropagation the number of units from the hidden layer was the tuning parameter.

The results show in Table 1 reflect a good accuracy for all cases, being the best one SC+ANFIS. Observing the rule number generated in each case by each system, in general this is small in most of the cases, so compact enough rule sets have been generated but this does not mean that a refinement process not is needed. If a rank is made, the best case, and the most homogeneous is the Feedforward one, next FasArt, FasBack and finally, the worst one is SC+ANFIS.

**Similarity.** In order to test the quality of the rules for each case, the similarity among them must be considered for detecting either redundancies or incoherencies so a reduction and simplification process could be taken in account. In (Setnes *et al.*, 1998; Chao *et al.*, 1996) a similarity criterion is formulated:



**Fig. 2.**  $S(A_i, A_{i+1})$ ,  $A_i$  and  $A_{i+1}$  are neighboring fuzzy sets, for each partition of each mode

$$S(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{\sum_i^j [\mu_a(x_j) \wedge \mu_b(x_j)]}{\sum_i^j [\mu_a(x_j) \vee \mu_b(x_j)]} \tag{2}$$

In Figure 1 the similarities (maximum, minimum, mean and median values) for the antecedent rule of each considered mode are shown. In general, the best case is the feedforward one, next the FasArt and FasBack systems. Anyone generates redundancies/incoherencies (similarity = 1), but the ANFIS and FasBack cases have similarities near 1 in several modes that indicate a rule simplification could improve/simplify the fuzzy rule set.

**Interpretability: Completeness and Distinguishability.** This is other interesting quality goal: to try that the knowledge contained in the fuzzy system (fuzzy rules) was accessible enough to “experts” in the problem domain. This can be achieved by studying the interpretability of the generated rules, that is implemented by the completeness and distinguishability concepts (Jin and Sendhoff, 2003): any element in each universe of discourse is covered by some fuzzy



**Table 2.** Knowledge quality from neural and neurofuzzy systems

|                          | SC+Anfis | Backp. | FasArt | Fasback |
|--------------------------|----------|--------|--------|---------|
| Accuracy                 | best     | good   | good   | good    |
| Compactness: Rule number | worst    | best   | good   | good    |
| Completeness             | best     | -      | bad    | bad     |
| Distinguishability       | regular  | -      | good   | good    |
| Refinement               | yes      | -      | yes    | yes     |
| Complexity method        | low      | high   | low    | low     |

set but being different enough these fuzzy sets:  $S(A_i, A_{i+1})$  must be different than 0 and lower enough than 1.

The Figure 2 shows the similarities between neighboring fuzzy sets ( $S(A_i, A_{i+1})$ ) in each fuzzy partition (6) of each model (13) generated by SC+Anfis (a), FasBack (b) and FasArt (c), in the feedforward networks this is not possible due to the extraction method employed.

In general, if  $S(A_i, A_{i+1})$  is considered, the overlapping for each one of these set pairs is not small, so an "adequated" degree of completeness could be achieved, but this measure is not enough for estimating the completeness: it could have universe parts not covered by any fuzzy set, so a:

$$\sum_i^n \mu_{ij}(x) \quad x \in U_i \nabla i(\text{fuzzyset}), j(\text{universe}). \quad (3)$$

must be made, but this last aspect can depend of the problem involved.

According to this, the only neuro system that generates every mode models with complete partitions is SC+ANFIS, FasArt only produces 3 models with complete fuzzy partitions for every variable, and 5 models with all their fuzzy partitions not complete, and Fasback 2 and 5 models respectively.

On the other hand, if the upper limit of  $S$  is considered, the similarity values are lower than 1 but some cases for the three systems these values are near 1 that indicates low interpretability. In general, for FasArt and FasBack is possible to achieve "acceptable" distinguishability with some exceptions, the worst case is feedforward. This indicates refinement/simplification process could be improved the compactness and interpretability of the fuzzy rule set.

## 4 Conclusions

This paper described a little study about the capacity of generating knowledge by fuzzy rules from well-known neural and neuro-fuzzy systems. Several criteria described in literature for this goal have been employed and some qualitative results are summarized in table 2 when the modelling of an AC motor is involved.

In general all systems involved permit obtain fuzzy rule sets with small number of them, being backpropagation case and the extraction algorithm used the

worst case. Interpretability can be improved in all system in general by a simplification process that will permit achieve a better distinguishability and completeness, in special for FasArt and Fasback cases, so to improve the interpretability and the expression power of the knowledge extracted.

The complexity of the extraction is low for FasArt, FasBack and Anfis cases due to their interpretability as fuzzy systems, but in the case of feedforward network is needed to applied an algorithm that permits to obtain the fuzzy rules so the complexity arises too much, and several alternatives for this task are possible.

On other hand, FasBack and FasArt permit to generate Mamdani fuzzy rules that is a very interesting aspect when expressive power is taken in account. The rest of models generate TSK rules.

## References

- Andrews, R., J. Diederich and A. B. Tickle (1995). Survey and critique of techniques for extracting rules from trained artificial neural networks. *Knowledge-Based Systems* **8**(6), 373–389.
- Cano Izquierdo, J. M., Y. A. Dimitriadis, E. Gómez Sánchez and J. Coronado López (2001). Learning from noisy information in FasArt and fasback neuro-fuzzy systems. *Neural Networks* **14**(4-5), 407–425.
- Castro, J. L., C. J. Mantas and J. M. Benitez (2002). Interpretation of artificial neural networks by means of fuzzy rules. *IEEE Transactions on Neural Networks* **13**(1), 101–116.
- Chao, C. T., Y. J. Chen and C. C. Tang (1996). Simplification of fuzzy-neural systems using similarity analysis. *Transactions on Systems, Man and Cybernetics:Part B* **26**(2), 344–354.
- Chiu, S. L. (1994). Fuzzy model identification based on cluster estimation. *Journal of Intelligent and Fuzzy Systems* **2**, 267–278.
- Haykin, S. (1999). *Neural Networks, Second Edition*. Prentice Hall. New Jersey.
- Jang, J. S. R. (1993). ANFIS: Adaptive-network-based fuzzy inferences systems. *IEEE Transactions on Systems, Man and Cybernetics* **23**, 665–687.
- Jin, Y. and B. Sendhoff (2003). Extracting interpretable fuzzy rules from RBF networks. *Neural Processing Letters* **17**(2), 149–164.
- Mitra, A. and Y. Hayashi (2000). Neuro-fuzzy rule generation: Survey in soft computing framework. *IEEE Transactions on Neural Networks* **11**(3), 748–768.
- Sainz Palmero, G.I., Y. Dimitriadis, J.M. Cano Izquierdo, E. Gómez Sánchez and E. Parrado Hernández (2000). ART based model set for pattern recognition: FasArt family. In: *Neuro-fuzzy pattern recognition* (H. Bunke and A. Kandel, Eds.). pp. 147–177. World Scientific Pub. Co.
- Setnes, M., R. Babuska and H. B. Verbruggen (1998). Ruled-based modeling: Precision and transparency. *IEEE Transactions on Systems, Man, and Cybernetics Part C: Applications and Reviews* **28**(1), 165–169.
- Taha, I. A. and J. Ghosh (1999). Symbolic interpretation of artificial neural networks. *IEEE Transactions on Knowledge and Data Engineering* **11**(3), 448–463.

# Adaptive Power System Stabilizer Using ANFIS and Genetic Algorithms

Jesús Fraile-Ardanuy and Pedro J. Zufria

Grupo de Redes Neuronales, ETSI Telecomunicación, UPM, Spain  
jefar@caminos.upm.es, pzz@mat.upm.es  
<http://www.mat.upm.es/>

**Abstract.** This paper presents an adaptive Power System Stabilizer (PSS) using an Adaptive Network Based Fuzzy Inference System (ANFIS) and Genetic Algorithms (GAs). Firstly, genetic algorithms are used to tune a conventional PSS on a wide range of operating conditions and then, the relationship between these operating points and the PSS parameters is learned by the ANFIS. The ANFIS optimally selects the classical PSS parameters based on machine loading conditions. The proposed stabilizer has been tested by performing nonlinear simulations using a synchronous machine-infinite bus model. The results show the robustness and the capability of the stabilizer to enhance system damping over a wide range of operating conditions and system parameter variations.

## 1 Introduction

Power system control requires a continuous balance between electrical generation and a varying load demand, while maintaining system frequency and voltage levels. The use of high performance excitation systems is essential for maintaining steady state and transient stability of modern synchronous generators and provides fast control of the terminal voltage. However, these fast acting exciters with high gains can contribute to oscillatory instability in the power system. This type of instability is characterized by low frequency oscillations which can persist or even grow in magnitude [1, 2]. Several real examples have been recorded and studied [3, 4].

In order to avoid this effect, supplementary stabilizing signals have been proposed in the excitation systems through lead/lag power system stabilizers [5] or PI - PID power system stabilizers [6]. The computation of the fixed parameters of these stabilizers is based on the linearized model of the power system around a nominal operating point.

The operating condition does change as a result of load variation and major disturbances, making the dynamic behaviour of the power system to become different, at different operating points. Thus, if the parameters of the stabilizer are kept fixed, PSS performance is degraded whenever the operating point changes.

Therefore, a good PSS design must consider a trade-off between: adaptability to the changes of the dynamics of the plant and easy design.

From the seventies, developments in digital technology have made possible to implement new controllers using adaptive control techniques [7]-[10]. These stabilizers provide better dynamic performance over a wide range of operating conditions, but they suffer from the major drawback of requiring parameter model identification, state observation and feedback gain calculations in real-time. If there is some error in parameter identification, it can lead to generate incorrect control signals, reducing the robustness.

In last decade, Fuzzy Logic Controllers (FLCs) and Artificial Neural Network Controllers (ANNCs) being used as power system stabilizers, have been developed and tested [11]-[18]. Unlike other classical control methods, FLCs and ANNCs are model-free controllers; i.e they do not required an exact mathematical model of the controlled system. Moreover, speed and robustness are the most significant properties in comparison to the other classical schemes. But these controllers present some disadvantages. There are not practical systematic procedures for the Fuzzy PSS (FPSS) design, so the rules and the membership functions of the controller are tuned subjectively, making the design laborious and a time-consuming task. With respect to ANNCs, they have the capability of learning and adaptation, but they work like a 'black-box' and it is difficult to understand the behaviour of the network.

In order to make the design simpler, Genetic Algorithms (GAs) have been successfully applied to PSS design [19]-[22]. GAs define a global optimization technique based on the mechanics of natural selection and survival-of-the-fittest.

In the research reported in this paper, a conventional PSS for a single machine infinite bus system is tuned by an Adaptive Network based Fuzzy Inference System (ANFIS) trained from the input-output data generated by a GA. The advantages of this design are:

- The Genetic Algorithm uses a simple objective function which does not depend on the mathematical model of the electric power system.
- The ANFIS combines the advantages of FLCs and ANNCs, avoiding their problems.
- The PSS parameters are modified on-line, in order to get a better response in the entire range of operation.

## 2 Genetic Algorithms and ANFIS

### 2.1 Genetic Algorithms

Genetic Algorithms copy the process of the natural evolution and their principles were firstly published by Holland [23]. In GAs, the features characterizing an individual are often binary coded in bits and concatenated as a string. The string package made from different combinations of bits is referred to as a point in the solution space. The fundamental operations of a GA, detailed in [24], are:

*Initialization.* The GA work with a population of strings, which evolves iteratively by generating new individuals taking the place of their parents. In our application, each string does code a set of PSS parameters.

*Objective Function.* The performance of each structure is evaluated according to its 'fitness', which is defined as the non-negative figure of merit to be maximized.

*Genetic Operations.* With the evaluation of the fitness function of all individuals, the GA generates a new and improved population from the old one. Most commonly used operations are the following: *Reproduction*, *Crossover* and *Mutation*.

### 2.2 ANFIS

ANFIS was proposed by [25]. In this neural fuzzy control system, the consequents of the Takagi-Sugeno (TS) fuzzy rules are linear combinations of their preconditions.

A more detailed information about ANFIS can be found in [24].

## 3 Proposed Adaptive PSS

The proposed PSS is shown in Fig. 1, where the ANFIS system adjusts the PSS parameters processing the operating points data defined by the active power output,  $P_t$ , reactive power output,  $Q_t$ , and the terminal voltage,  $V_t$  of the generator. A widely used conventional lead-lag PSS is considered in this paper. The time constant of its wash-out filter,  $T_w$ , is set to be 2.0 seconds.

ANFIS outputs are the optimal PSS parameters generated by the genetic algorithm in this particular operating point defined by the inputs. The PSS

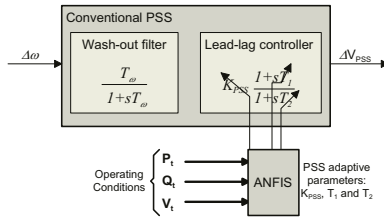


Fig. 1. Proposed Adaptive PSS

design is divided in two different parts:

- Tuning the conventional PSS using GA in different operating points, storing the PSS optimal parameters and the operating conditions.
- Training the ANFIS system with the stored data.

### 3.1 Collecting the Training Data

Firstly, a GA is used to adjust the PSS parameters on a wide spectrum of operating conditions, i.e., the generator power output ranging from 0.1 p.u. to

1.1 p.u., and the power factor ranging from 0.7 lead to 0.1 lag. Similarly, a wide range of possible disturbances is used for the training. These disturbances are: reference voltage in the range of -0.1 to 0.5 p.u., input mechanical torque variation from -0.2 p.u. to 0.2 p.u., one transmission line outage and three phase fault on one of the double circuit transmission lines connected to the generator.

Starting in a particular operating point, the GA searches the PSS parameters that optimize the fitness function:

$$f_k = \frac{1}{1 + \int_{t_1}^{t_2} t |\Delta\omega(t)| dt} \quad (1)$$

where  $t_1$  and  $t_2$  are the study time limits and  $\Delta\omega(t)$  represents the speed deviation of the generator. The PSS parameters  $K_{PSS}$ ,  $T_1$  and  $T_2$  are selected so as to maximize the objective function  $f_k$ . The advantage of the selected fitness function as opposed to other functions proposed in [20], [21] and [26], is that minimal dynamic plant information is needed. It is only necessary to measure the speed deviation of the generator instead of identifying on-line the electric power system model parameters, needed to design the PSS by a pole-placement technique. For a given operating point, the objective function is evaluated with an initial population randomly generated. Each individual in the initial population has an associated fitness function value. Using this information, GA operations are applied to produce the next generation. These two steps, evaluation of the objective function and generation of the new population, are repeated from generation to generation until the population does converge, producing the optimum PSS parameters (within GA limitations) for this particular operating point. Once the PSS parameters are tuned, these data are stored together with the loading condition and the algorithm starts again in a different operating point. All the data collected constitute the training set.

### 3.2 ANFIS Training

A total of 5000 input-output data pairs were obtained for the training of the ANFIS. The ANFIS transforms a fuzzy inference engine into an adaptive network that learns the relationship between inputs, defined by the operating conditions, and outputs, defined by the PSS parameters. These relationships are learned independently for each PSS parameter, using three ANFIS, in order to improve the converge speed of the ANFIS hybrid learning algorithm. Choosing the correct number of membership functions is a fundamental question often raised in these applications. Usually this number is determined experimentally in a similar way to choosing the number of neurons in the hidden layer of an artificial neural network. But there are other methods based on pruning or growing the network. Pruning algorithms start with a larger network and then prune it to desired size [27, 28]. On the opposite, growing algorithms start with a small network and gradually increase it to appropriate size [29]. In this paper, the number of membership functions for each input variable is determined by trial and error for simplicity. In this study, five linguistic variables for each input variable were use to get the desired performance.

## 4 Simulation Studies

To demonstrate the effectiveness of the proposed PSS whose parameters are adapted by the ANFIS, time domain simulations were performed for the generator under major disturbance conditions over a wide range of loading conditions. Test cases are similar to those proposed in [11]-[13] and [15]-[22], to verify the PSS behaviour. The considered system is a synchronous machine connected to an infinite bus through two parallel transmission lines as shown in Fig. 2.

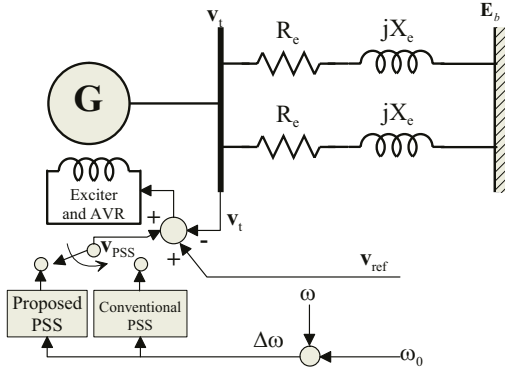


Fig. 2. Single machine infinite bus system with two transmission lines

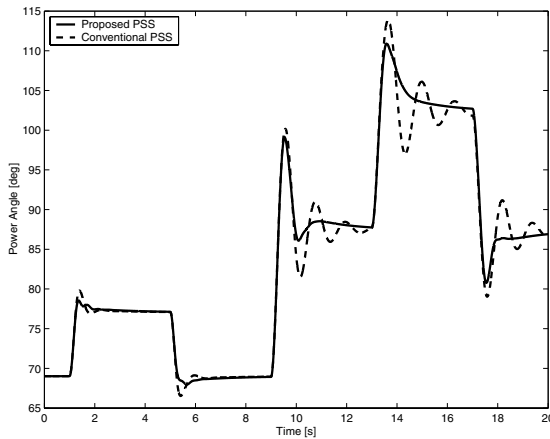
The nonlinear model equations and parameters used to simulate the generating unit, AVR and conventional PSS (CPSS) are given in [26]. The CPSS is adjusted to give optimal performance for the operating point of 1 p.u. generated power, and 0.97 power factor lag.

### 4.1 Switching Off One Line

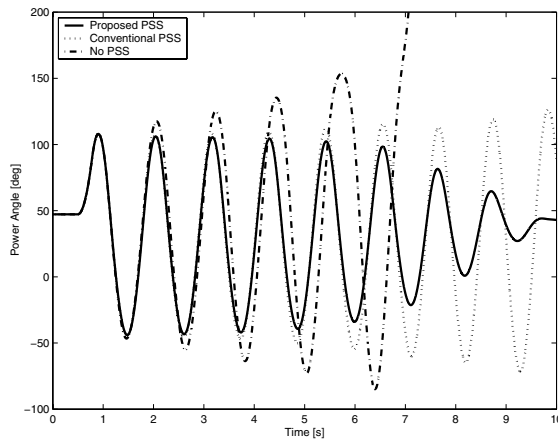
At an operating point of 0.9 p.u., 0.98 power factor lag, one circuit of the double circuit transmission line was switched off at  $t=9$  s. Before the line disconnection, 0.2 p.u. step increase in the input torque reference was applied at  $t=1$  s and removed at  $t=5$  s. A second 0.2 p.u. step increase in the input torque reference was applied at  $t=13$  s. and removed at  $t=17$  s. during the transmission line fault. The response of the system with both conventional PSS and the proposed PSS is shown in Fig. 3. The response with the proposed PSS shows less oscillations than the conventional PSS and demonstrates better performance.

### 4.2 Three Phase to Ground Fault

Three phase to ground fault locating at 50 % of the distance along line is applied at  $t=0.5$  s. and the fault is cleared at  $t=0.773$  s. Responses without stabilizer and with conventional PSS and proposed PSS are shown in Fig. 4. Systems without stabilizer and with conventional PSS are both unstable. The system with proposed PSS is highly oscillatory, but it is stable.



**Fig. 3.** Power angle for a  $\pm 0.2$  p.u. input torque reference step change at  $P=0.9$  p.u. and  $pf=0.98$  p.u. during the switching of one of the transmission lines



**Fig. 4.** Three phase to ground fault

## 5 Conclusions

An adaptive power system stabilizer based on an Adaptive-Network-Based Fuzzy Inference System and Genetic Algorithms is presented in this paper. The effectiveness of this scheme has been investigated through nonlinear simulations. The following conclusions are derived from the results:

- Since PSS parameters are adjusted by a genetic algorithm, minimal knowledge about the system is required and there is no need for system model linearization.
- Due to the selected fitness function, opposite to other functions proposed in other papers, there is no need to identify the power system model parameters.



So this PSS design, which is not based on a pole-placement technique, saves time and reduces the complexity of the design.

- The ANFIS combines the advantages of artificial neural networks and fuzzy logic controllers, that is, adaptation and robustness.
- Since the proposed design modifies the parameters of the installed classical PSS, there is no need to change the actual PSS in the generator.
- Test results for various operating conditions and disturbances show that the proposed stabilizer is able to provide good damping over a wide operating range and improves the overall system performance.

**Acknowledgments.** This work has been partially supported by projects BFM2003-02960 and MTM2004-05316 from the MCyT and MEyC, Spain, respectively.

## References

1. F. P. deMello and C. Concordia: Concepts of synchronous machine stability as affected by excitation control, *IEEE Transactions Power Apparatus and Systems*, Vol. 88, (1969), 316-329.
2. M. Klein, G. J. Rogers and P. Kundur: A fundamental study of inter-area oscillations in power systems, *IEEE Transactions on Power Systems*, Vol. 6, No. 3, (1991), 914-921.
3. E. L. Busby, J. D. Hurley, F. W. Keay and C. Raczkowski: Dynamic stability improvement at Monticello station - analytical study and field tests, *IEEE Trans Power Apparatus and Systems*, Vol. PAS-98, No. 34, (1979), 889-901.
4. J. E. Van Ness, F. M. Brash Jr., G. L. Landgren and S. T. Naumann: Analytical investigations of dynamic instability occurring at Powerton station. *IEEE Transactions Power Apparatus and Systems*, Vol. PAS-99, No. 4, (1980), 1386-1395.
5. E. V. Larsen and D. A. Swann: Applying power system stabilizers, parts I and II, *IEEE Transactions Power Apparatus and Systems*, Vol. 100, (1981), 3017-3046.
6. Y. Y. Hsu and C. Y. Hsu: Design of a proportional-integral power system stabilizer, *IEEE Trans Power Systems*, Vol. 2, No. 1, (1987), 92-100.
7. A. Gosh, G. Ledwich, O. P. Malik and G. S. Hope: Power system stabilizer based on adaptive control techniques, *IEEE Transactions Power Apparatus and Systems*, Vol. 103, No. 8, (1984), 1983-1989.
8. Y. Y. Hsu and K. L. Liou: Design of self-tuning pid power system stabilizers for synchronous generators, *IEEE Transactions on Energy Conversion*, Vol. 2, No. 3, (1987), 343-348.
9. G. P. Chen, O. P. Malik, G. S. Hope, Y. H. Qin and G. Y. Xu: An adaptive power system stabilizer based on the self-optimizing pole shifting control strategy, *IEEE Transactions on Energy Conversion*, Vol. 8, No. 4, (1993), 639-645.
10. M. L. Kotari, K. Bhattacharya and J. Nanda: Adaptive power system stabiliser based on pole-shifting technique, *IEE Proceedings on Generation, Transmission and Distribution*, Vol. 143, No. 1, (1996), 96-98.
11. A. Hariri and O. P. Malik: A fuzzy logic based power system stabilizer with learning ability, *IEEE Transactions on Energy Conversion*, Vol. 11, No. 4, (1986), 721-727.

12. M. A. M. Hassan, O. P. Malik and G. S. Hope: A fuzzy logic based stabilizer for a synchronous machine, *IEEE Transactions on Energy Conversion*, Vol. 6, No. 3, (1991), 407-413.
13. K. A. El-Metwally, G. C. Hancock y O. P. Malik: Implementation of a fuzzy logic PSS using a micro-controller and experimental test results, *IEEE Transactions on Energy Conversion*, Vol. 11, No. 1, (1996), 91- 96.
14. M. K. El-Sherbiny, G. El-Saadly and E. A. Ibrahim: Efficient incremental fuzzy logic for power system stabilization, *Electric Machines and Power Systems*, Vol. 25, (1997), 429-441.
15. Young-Moon Park, Myeon-Song Choi and Kwang Y. Lee: A neural network-based power system stabilizer using power flow characteristics, *IEEE Transactions on Energy Conversion*, Vol. 11, No. 2, (1996), 435-441.
16. Young-Moon Park, Seung-Ho Hyun and Jin-Ho Lee: A synchronous generator stabilizer design using neuro inverse controller and error reduction network, *IEEE Transactions on Power Systems*, Vol. 11, No. 4, (1996), 1969-1975.
17. D. Flynn, S. McLoone, G. W. Irwin, M. D. Brown, E. Swidenbank and B. W. Hogg: Neural control of turbogenerator systems, *Automatica*, Vol. 33, No. 11, (1997), 1961-1973.
18. M. M. Salem, A. M. Zaki, O. A. Mahgoub, E. Abu El-Zahab and O. P. Malik: Experimental verification of a generating unit excitation neuro-controller, *IEEE Power Engineering Society Winter Meeting*, Vol. 1, (2000), 585-590.
19. Y. L. Abdel-Magid, M. Bettayeb y M. M. Dawoud: Simultaneous stabilisation of power systems using genetic algorithms, *IEE Proc.-Gener. Transm. Distrib.*, Vol. 144, No. 1, (1997), 39-44.
20. M. A. Abido and Y. L. Abdel-Magid: A genetic-based power system stabilizer, *Electric Machines and Power Systems*, Vol. 26, (1998), 559-571.
21. Y. L. Abdel-Magid, M. A. Abido, S. Al-Baiyat and A. H. Mantawy: Simultaneous stabilization of multimachine power systems via genetic algorithms, *IEEE Transactions on Power Systems*, Vol. 14, No. 4, (1999), 1428-1439.
22. P. Lakshmi and M. Abdullah Khan: Stability enhancement of a multimachine power system using fuzzy logic based power system stabilizer tuned through genetic algorithm, *Electric Power and Energy Systems*, Vol. 22, (2000), 137-145.
23. Holland, J. H.: Outline for a logical theory of adaptive systems, *J. Assoc. Comput. Mach.*, Vol. 3, (1962), 297-314.
24. Lin, C-T. and C. S. G. Lee: *Neural Fuzzy Systems. A Neuro-Fuzzy Synergism to Intelligent Systems*. Prentice Hall, (1996).
25. J. Roger Jand: ANFIS: adaptive-network-based fuzzy inference system, *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 23, No. 3, (1993), 665-685.
26. Fraile-Ardanuy, J.: Design of an adaptive neuro-fuzzy PSS adjusted by a genetic algorithm. Ph.D Thesis. Polytechnic University of Madrid (Spain), (2003).
27. Karnin, E. D.: A simple procedure for pruning back-propagation trained neural networks, *IEEE Transactions on Neural Networks*, Vol. 1, No. 2, (1990), 239-241.
28. Sietsma, J. and R. J. F. Dow: Neural net pruning. Why and how?, *Proc. IEEE Int. Conference on Neural Networks*, Vol. 1, (1998), 325-333.
29. Hertz, J., A. Krogh and R. G. Palmer: *Introduction to the Theory of Neural Computation*. New York, Addison-Wesley, (1991).

# Use of ANN in a Research Reactor Power Fuzzy Controller

Jorge S. Benítez-Read<sup>1</sup>, Da Ruan<sup>2</sup>, Jorge A. Ruiz-Enciso,  
Régulo López-Callejas, and Joel O. Pacheco-Sotelo

<sup>1</sup> The Mexican National Institute for Nuclear Research (ININ),  
Km. 36.5 Carretera México-Toluca, Ocoyoacac, Edo. de México,  
52749, Toluca Institute of Technology (ITT), Mexico  
jsbr@nuclear.inin.mx

<sup>2</sup> Belgian Nuclear Research Centre (SCK•CEN), Boeretang 200, 2400 Mol, Belgium

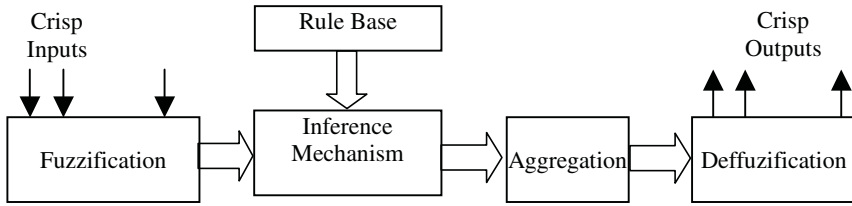
**Abstract.** Six artificial neural networks (ANN) were developed to compute the membership values of the consequents of a fuzzy rule evaluation table, which is part of a Mamdani type fuzzy controller. This controller is aimed to regulate the neutron power of a research nuclear reactor. The neural networks obtained were validated over a wide range of input data. These ANN offer the possibility of a parallel processing of the fuzzy inputs, thus reducing the response time of the controller.

## 1 Introduction

In TRIGA type reactors, the power control algorithms process plant variables such as the error between the desired power level and the current power, and the instantaneous reactor period value, which is a measure of the rate of change of the reactor power. The controller determines the amount of reactivity that is needed to increase or decrease power, as well as the corresponding motion of the control rods that introduce that required reactivity in the core. A particular control algorithm is derived from the control theory used. For instance, classical control methods such as proportional, integral, derivative, or modern methods such as pole placement, can be applied. More recent techniques, such as fuzzy logic based methods, are being proposed for power control [4]. In fuzzy logic based controllers, the input and output signals are real-valued quantities, as in any other controller. The main difference is the processing of the input data, which is based on the theory of fuzzy sets and fuzzy logic.

A fuzzy controller is a special expert system that utilizes a knowledge base, in turn represented in a series of fuzzy inference rules. The operation of a fuzzy controller of the Mamdani type is generally divided in the following stages: a) Fuzzification, b) Rule evaluation, c) Aggregation, and d) Defuzzification (see Fig. 1) [1].

The operating rules are usually of the type "If *antecedent*, then *consequent*". These rules describe the action that needs to be taken by the controller in response to various fuzzy inputs.



**Fig. 1.** Processing stages in a fuzzy controller

On the other hand, artificial neural networks (ANN) are composed by an adequate interconnection of nodes through which the input data is processed according to the specific application for which the ANN is devised [3].

These networks can be implemented either in hardware or software. Common applications of ANN are found in the areas of system identification and control.

When the amount of rules of the knowledge base in a fuzzy system is large, the number of operations in the stage of rule evaluation is increased. In systems where the operations are carried out sequentially, this increase in the number of operations results in longer controller's response time. This can lead to difficulties for its application in real time. Given that ANN are computing structures with the capability of parallel processing, the idea of this work is to design a series of ANN able to perform the functions of the rule evaluation stage of a fuzzy controller [2]. This controller, together with the ANN, will be tested in a point kinetic model of a research TRIGA reactor to regulate the reactor's power.

## 2 Fuzzy Control System

The operation of the reactor is monitored by the continuous measuring of two parameters, the reactor period and the normalized deviation of the current power level from the set point. Taking into account the safety requirement of maintaining the reactor period above the 3-second level, the fuzzy controller brings the neutron power from its source level to the desired point. A block diagram of the closed-loop fuzzy control system is shown in Fig. 2.

The reactor dynamics is modelled by a set of three nonlinear differential equations. The reactor's measured variables are the neutron power,  $n(t)$ , and the reactor period,  $T(t)$ . The control input  $\rho_{\text{ext}}(t)$  represents the external reactivity applied to the reactor by means of the insertion or withdrawal of the control rod [5].

The common crisp inputs to a two-input fuzzy controller are the reactor period and the normalized percent of the neutron power deviation from its set point, %PD. The controller input %PD is computed by the expression  $\%PD = \frac{n(t) - n_{\text{setpoint}}}{n_{\text{setpoint}} - n_0} \times 100\%$ ,

where  $n_0$  represents the initial stable neutron power.

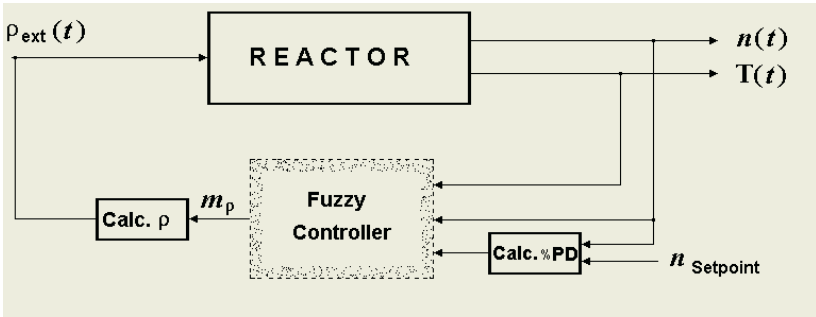


Fig. 2. Block diagram of the closed-loop fuzzy control system

Each set of crisp input variables to the fuzzy controller produces one crisp output variable,  $m_\rho$ , which represents the rate of change of external reactivity. Once  $m_\rho$  is defined, the external reactivity,  $\rho_{ext}(t)$  (control variable), is determined by the expression  $\rho_{ext}(t_i) = \rho_{ext}(t_{i-1}) + m_\rho(t_i - t_{i-1})$ , where  $t_i$  represents the current time and  $t_{i-1}$  the previous time at which the reactivity value of  $\rho_{ext}(t_{i-1})$  was applied to the reactor.

### 3 Methodology and Design of the ANN that Replace the Rule Evaluation Stage of the Fuzzy Controller

The design of the neural networks was based on the particular characteristics of the rule evaluation stage of the reactor’s power fuzzy controller. The methodology used for the design is explained in the following sections.

#### 3.1 Grouping of Rules Sharing the Same Consequent

Initially, it was proposed to design a single ANN whose inputs were the membership values of the fuzzy sets associated to the variables T (reactor period) and %PD (power error in percentage). This would result in 10 inputs to the ANN. The network would have, as outputs, the membership values of the four output fuzzy sets that would correspond to the ten input values. The output sets are associated to the control signal  $m_\rho$ , which represents the rate of change of the reactivity that is inserted into the reactor’s core.

Considering 10 samples for each of the 10 inputs to the ANN, an enormous number of patterns ( $10^{10}$ ) would be generated for the training of the network. The training of this single ANN with 10 inputs, 4 outputs, and the unmanageable number of training patterns, was foreseen extremely complicated and lengthy. This first approach was logically abandoned, giving place to the idea of using several ANN to accomplish the pretended functions.

The next and definite approach was to design one or two ANN for each consequent of the set of rules, where the output would be the membership value of this

consequent in its corresponding fuzzy set. Each set of ANN would have fewer inputs, the number of training patterns would be considerably reduced, and the convergence of the training process would hopefully be less complicated.

**3.2 Definition of the ANNs for Each Consequent**

The fuzzy associative memory of the fuzzy controller is shown in Table 1, where the labels assigned to the different fuzzy sets represent the following: a) For input T: CR, critical; PC, close to critical; NR, normal; GD, big; TI, tends to infinity; b) For input %PD: GN, negative big; MN, negative medium; PN, negative small; ID, ideal; PP, positive small; and c) For output  $m_p$ : NP, negative small; CE, zero; PQ, positive small; PG, positive big.

**Table 1.** Fuzzy associative memory of the controller

|     |     | T   |     |     |     |     |
|-----|-----|-----|-----|-----|-----|-----|
|     |     | ~CR | ~PC | ~NR | ~GD | ~TI |
| %PD | ~GN | ~CE | ~PQ | ~PG | ~PG | ~PG |
|     | ~MN | ~CE | ~PQ | ~PG | ~PG | ~PG |
|     | ~PN | ~CE | ~PQ | ~PQ | ~PQ | ~PQ |
|     | ~ID | ~CE | ~CE | ~CE | ~CE | ~CE |
|     | ~PP | ~CE | ~NP | ~NP | ~NP | ~NP |

$m_p$

Based on Table 1, six ANN were defined as follows: a) One ANN for the output NP; b) Two ANN for the output CE; c) Two ANN for the output PQ; and d) One ANN for the output PG. Two ANN were chosen for each of the outputs CE and PQ because the number of patterns was still too large if only one ANN for each output had been used.

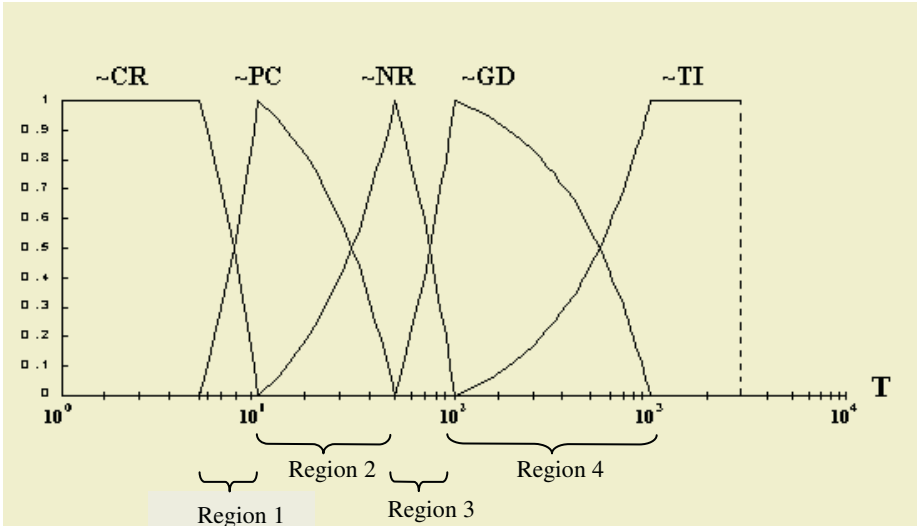
**3.3 Definition of Inputs and Training Data for Each ANN**

A given ANN only uses training patterns associated with the antecedents participating in the consequent associated to that ANN. For example, for the output set NP, Table 1 shows that the antecedents involved are: PC, NR, GD and TI for variable T and PP for variable %PD. The antecedents for the rest of the output fuzzy sets are determined in a similar manner, except that two ANN are used for each of the consequents CE and PQ. This approach drastically reduces the number of training patterns. The inputs and output for each ANN are shown in Table 2.

The definition of the sample values of the inputs to the ANN is described next. Consider the fuzzy sets associated to variable T, and shown in Fig. 3. Each region contains one line of one of the input fuzzy sets and another line of a contiguous set. Ten values for each of the lines in a given region are selected. Considering the four regions, as well as 10 samples per region, then the total number of samples is 40.

**Table 2.** Inputs and output for each ANN

| ANN   | ANN output | ANN inputs related to variable T | ANN inputs related to variable %PD |
|-------|------------|----------------------------------|------------------------------------|
| ANN 1 | ~NP        | ~PC ~NR ~GD ~TI                  | ~PP                                |
| ANN 2 | ~CE1       | ~CR                              | All                                |
| ANN 3 | ~CE2       | All                              | ~ID                                |
| ANN 4 | ~PQ1       | ~PC                              | ~GN ~MN ~PN                        |
| ANN 5 | ~PQ2       | ~PC ~NR ~GD ~TI                  | ~PN                                |
| ANN 6 | ~PG        | ~NR ~GD ~TI                      | ~GN ~MN                            |



**Fig. 3.** Fuzzy sets associated to variable T

The training patterns for each ANN are obtained with ease. For instance, if 40 samples are considered per each variable, T and %PD, the training patterns for the ANN associated to output NP are obtained from the membership values of the antecedents regions that participate in the rules having NP as their output. Thus, the number of training patterns would be as follows: 400 for ANN1 (NP), 400 for ANN2 (CE1), 800 for ANN3 (CE2), 600 for ANN4 (PQ1), 800 for ANN5 (PQ2), and 600 for ANN6 (PG). The output values of each ANN are determined by performing the inference method that corresponds to the rule evaluation stage for each of their corresponding training inputs.

**3.4 Selection of the ANN Structure**

Considering the range of values between 0 and 1 for both inputs and outputs, and the availability of input-output patterns for training, the backpropagation neural network was chosen. This type of network is widely used in pattern recognition problems. Every ANN will have one neuron in the output layer. The number of neurons in the

intermediate layer is modified during the training of the network according to an iterative process. This number will be modified up to attaining a pre-specified convergence. The initial number of intermediate neurons is 10. The number of inputs in a given ANN depends on the antecedents associated to the consequent of that ANN. Since the ANN output values lie between 0 and 1, an appropriate activation function to use is the sigmoidal function. Initially, with ten intermediate neurons, the total number of internal interconnections is shown in Table 3.

**Table 3.** Internal interconnections in the ANN

| Input neurons   | Interconnections |
|-----------------|------------------|
| 6 (CE1, CE2)    | 70               |
| 5 (NP, PQ2, PG) | 60               |
| 4 (PQ1)         | 50               |

### 3.5 ANN Convergence and Adjustment of Intermediate Neurons

The parameters for the training are: an error value of 0.02 and a total of 10000 epochs. When a training process converges, the final error, attained in 10000 epochs or less, should be less than 0.02. None of the six ANN converged in less than 10000 epochs. Under lack of convergence, the training program increases by one the number of intermediate neurons, and proceeds to a new training process. When convergence is attained, the weights  $W$ , bias  $b$ , number of epochs and final error, are available to implement the ANN together with the other fuzzy controller stages.

## 4 Results and Discussion

As mentioned in the previous section, with the initial 10 intermediate neurons per ANN, none of the six ANN converged to the pre-specified error after 10000 epochs of training. After several adjustments on the number of these neurons, convergence of all the ANN was obtained. The final configuration of the intermediate layers is shown in Table 4.

**Table 4.** Final configuration of the intermediate layers

| ANN | Number of neurons in the input layer | Number of neurons in the intermediate layers |
|-----|--------------------------------------|----------------------------------------------|
| CE1 | 6                                    | 26                                           |
| CE2 | 6                                    | 38                                           |
| NP  | 5                                    | 30                                           |
| PG  | 5                                    | 42                                           |
| PQ1 | 4                                    | 16                                           |
| PQ2 | 5                                    | 24                                           |



Matlab 5.0 was used for the training process. The membership values of the output fuzzy set NP obtained with the ANN and the fuzzy rule evaluation stage are shown in Fig. 4. The values are shown for 400 input patterns.

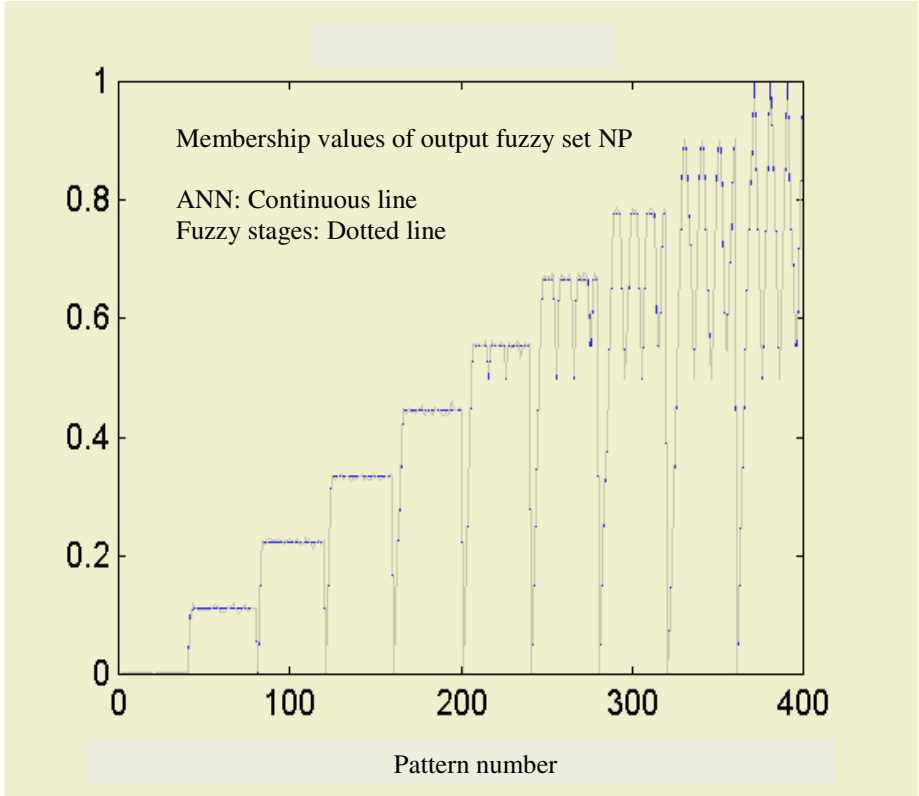


Fig. 4. Membership values for the output fuzzy set NP obtained with the ANN and the fuzzy stages

Similar results were obtained for the other consequents of the fuzzy rule table.

## 5 Conclusions

The values generated by the ANN are practically the same than those obtained by the fuzzy rule evaluation stage. Globally, the combined ANNs perform a greater number of operations compared to the operations realized with the traditional fuzzy algorithms for the evaluation of rules. However, considering the parallel computing capability of the ANN, the four ANN output values would be computed in less time. The longest sequential operation chain in the ANNs contains 57 operations whereas the fuzzy rule evaluation stage carries out 71 sequential operations.

For input data not considered in the training pattern but physically possible to occur, the output values obtained with the ANN are also very close to those obtained with the original rule evaluation algorithm. For instance, the input data vector [0.6333, 0.3667, 0, 0, 0, 0, 0, 0, 0.9444, 0.0556] was not used for the training of the ANN. The rule evaluation algorithm gives the output vector [ 0, 0, 0, 0.6333], whereas the ANN method delivers [ 0.0019, 0.0343, 0.0041, 0.6391].

As further work, these ANN are being integrated into the fuzzy control algorithm, and, due to government regulation, the performance in the regulation of power will be tested using a point kinetic model of the reactor.

## Acknowledgements

This work was partially funded by the Mexican National Council for Science and Technology (Conacyt) under grant 33797-A, and by the Council of the National System of Technological Education (Cosnet) under grant 463.01-PR.

## References

1. Benítez-Read, J.S. Sotuyo, A.I.: Análisis de las etapas de procesamiento difuso, Informe Técnico IT.AU-01-13, ININ, México (2001)
2. Benítez-Read, J.S. and Sotuyo, A.I.: Las técnicas de redes neuronales y lógica difusa en las distintas etapas de los controladores basados en conocimiento, Informe Técnico IT.AU-01-18, ININ, México (2001)
3. James A., David M.: Neural Networks. Algorithms, Applications and Programming Techniques. Addison-Wesley Publishing Company (1992)
4. Tsoukalas, L., Berkan, R.C., Upadhyaya, B.R., Uhrig, R.E., Kisner, R.A.: Expert system driven fuzzy control application to power reactors., Proceedings of the 2nd International IEEE Conference on Tools for Artificial Intelligence (1990) 445–450
5. Vélez-Díaz, D. and Benítez-Read, J.S.: Estudio del comportamiento del reactor Triga Mark III, basado en simulaciones, Informe técnico IT.ET.A-9514, ININ, México (1995)

# The Use of Bayesian Networks for Subgrouping Heterogeneous Diseases

Abdelaziz Ouali<sup>1,2</sup>, Amar Ramdane Cherif<sup>2</sup>, and Marie-Odile Krebs<sup>1</sup>

<sup>1</sup> Lab. INSERM E0117, 2 ter rue d'Alésia 75014 Paris, France  
{ouali, mokrebs}@broca.inserm.fr

<sup>2</sup> Lab. PRiSM, Univ. De Versailles, 45 Av. des Etats-Unis 78035, Versailles, France  
{oua, rca}@prism.uvsq.fr

**Abstract.** Schizophrenia is a frequent and devastating disorder beginning in early adulthood. Until now, the heterogeneity of this disease has been a major pitfall for identifying the aetiological, genetic or environmental factors. Age at onset or several other quantitative variables could allow for categorizing more homogeneous subgroups of patients, although there is little information on which are the boundaries for such categories. The Bayesian networks classifier approach is one of the most popular formalisms for reasoning under uncertainty. We used this approach to determine the best cut-off point for three continuous variables (i.e. age at onset of schizophrenia and neurological soft signs) with a minimal loss of information, using a data set including genotypes of selected candidate genes for schizophrenia.

## 1 Introduction

Schizophrenia is a frequent disorder (estimated prevalence around 1% of the community), that usually becomes apparent during adolescence or in early adulthood [1][2].

Compelling evidence support the involvement of genetic factors. For instance, family studies have shown that the risk of schizophrenia is higher in a first-degree relative of a proband (i.e. 9 percent for a sibling, 13 percent for a child of the affected person) [3]. Twin studies using pairwise concordance rates have revealed that the risk of developing schizophrenia for the second twin is approximately 60 percent in a monozygous pair compared to 20 percent in a dizygous pair [4]. Adoption studies also point to genetic factors in schizophrenia, with 11-19 percent of the adopted-away offspring of an affected parent later developing the disease compared to 0-11 percent of the adopted-away offspring of unaffected parents [5]. Despite recent convergent findings pointing to a few candidate genes, the identification of the genes responsible for schizophrenia still remains to be completed.

Nevertheless, there are still unanswered questions concerning the precise clinical variable that best predicts the actual age of onset of pathological processes. In particular, “age at onset” is often defined on the basis of “age at first psychotic episode”, i.e. first time when positive symptoms (hallucinations and delusions) present with a sufficient severity and duration. But some patients with early cognitive and adaptive

dysfunctions display delayed positive symptoms, leading to misclassification when using this classical definition of age at onset. We hypothesized that data mining analysis, taking into account clinical or biological attributes, could help to unravel the actual age at onset of the disease (unknown variable). We hypothesized that in each individual, the actual age of onset would be related to a particular pattern of clinical or biological variables. Thus, the variable that most reliably reflects the actual age at onset should be the variable that is best predicted by these variables.

In a previous paper, we examined how two variables related to “age at onset” could be predicted by clinical variables including comorbid diagnosis [6]. Using a neuronal network to predict age at onset with the clinical attributes, we showed that age of first contact (AFC) was more reliably classified than age of first episode (AFE). Nevertheless, we had to define cut-off for these two continuous variables and our results may have been dependent on those cut-off points. Indeed, we recoded AFC: 0 to 18 years coded “0”; “1”, otherwise and AFE: 0 to 20 years coded “0”; “1”, otherwise. There is, however, no true consensus in the literature concerning the best cut-off ages for classifying patients between early vs. late-onset, possibly because it depends on the clinical sample and/or on the variables that are examined. It is thus critical to obtain objective ways to define such cut-off. There is accumulating evidence suggesting that age at onset could be related to genetic background. In this paper, we use a Bayesian network algorithm on genotyping data as predictor variables and in order (i) firstly, to develop a more convincing method to choose the cut-off ages for the “age at onset” in schizophrenia, and (ii) secondly, to confirm that genetic characteristics influence the age at onset, further confirming the link between schizophrenia and genes.

## 2 Design of the Study

We carried out our research on data obtained from genetic studies conducted in the University Department of Psychiatry, Sainte Anne Hospital. There were 223 records in the data base, which contains healthy subject (controls) and patients. Within this dataset, there were heterogeneous clinical variables: different variables related to age at onset (for example for age at first contact, at first treatment, at first hospitalization, age when the positive symptoms of schizophrenia occurred) and also variables issued from the psychopathological or neuro-cognitive assessments. Among those variables, we have chosen three numeric variables as defined below:

**Age First Contact (AFC)** (with a practitioner for psychological reasons): is defined as the first time the subject has to seek help from a psychologist or practitioner because they are feeling psychological distress or experiencing unspecific symptoms as insomnia, anorexia, anxiety, loss of concentration....

**Age First Psychotic Episode (AFE)**: is defined as the first time delusions, hallucinations or disorganized behaviour become clinically significant, according to the widely used definition found in the literature.

**Score of Neurological Soft-Signs (NSS)**: Neurological soft signs are highly prevalent in patients with schizophrenia. Using a standardized scale defined in the labora-

tory [7], we have previously shown that NSS scores are related to other variables linked to abnormal development (i.e. minor physical abnormalities [8]) and to genetic risk [9].

Genotyping data were also available and used as predictor variables. The genotypes were related to genes considered as candidates for schizophrenia, either related to neurotransmission (dopaminergic, serotonergic or cannabinoid) or to neurodevelopment (brain derived neurotrophic factor (BDNF), REELIN, ENGRAILED2). Those variables were: the Bal-I polymorphism dopaminergic receptor type 3 DRD3, the 48 bp VNTR polymorphism of the dopaminergic receptor type 4 (DRD4), the 5HTTLPR polymorphism of the serotonin transporters, the ATT polymorphism of the cannabinoid receptor type 1 (CNR1), the GT and Val66met polymorphisms of brain derived neurotrophic factor (BDNF), 5'UTR polymorphism of the reelin gene's promoter, CA repeat polymorphism located in the 3' region of the homeogene engrailed 2 gene (EN2). All these genotypes are nominal variables corresponding to their respective alleles for each polymorphism. To simplify the text, we will call these genotypes: A, B, C, D, E, E', F and G, respectively. As indicated, E and E' concern the same gene and are in disequilibrium of transmission (i.e. their transmission is not independent). In addition, there are experimental data in the literature stressing that there are functional interactions of the products of DRD3 and BDNF genes as well as Reelin and BDNF genes.

### 3 Experimental Results

For this work, a dataset with 223 instances was selected. The three target attributes, initially numeric continuous variables (age at onset and NSS scores), were recoded into categories. In a binary prediction model (e.g. presence / absence, disease/normal) there are two possible prediction errors: false positives (FP) and false negatives (FN). The performance of a binary prediction model is normally summarized in a confusion or error matrix that cross-tabulates the observed and predicted + / - patterns.

**Table 1.** Confusion Matrix

|          | Predicted + | Predicted - |
|----------|-------------|-------------|
| Actual + | a           | c           |
| Actual - | b           | d           |

#### 3.1 Confusion Matrix Derived Measures

A variety of error or accuracy measures can be calculated from a confusion matrix. All of these measures assume that data are counts and not percentages (See Table 2).

These measures have different characteristics; in particular some are sensitive to the prevalence of positive cases. The remaining tables (Table 3, Table 4 & Table 5)

**Table 2.** Some derived measures of the Confusion matrix

| Measure                     | Calculation                                                                                      |
|-----------------------------|--------------------------------------------------------------------------------------------------|
| Prevalence                  | $(a + c)/N$                                                                                      |
| Overall Diagnostic Power    | $(b + d)/N$                                                                                      |
| Correct Classification Rate | $(a + d)/N$                                                                                      |
| Sensitivity                 | $a/(a + c)$                                                                                      |
| Specificity                 | $d/(b + d)$                                                                                      |
| False Positive Rate         | $b/(b + d)$                                                                                      |
| False Negative Rate         | $c/(a + c)$                                                                                      |
| Positive Predictive Power   | $a/(a + b)$                                                                                      |
| Negative Predictive Power   | $d/(c + d)$                                                                                      |
| Odds-ratio                  | $(ad)/(cb)$                                                                                      |
| Kappa (K)                   | $(a + d) - (((a + c)(a + b) + (b + d)(c + d))/N)$<br>$N - (((a + c)(a + b) + (b + d)(c + d))/N)$ |

illustrate how prevalence affects some of the prediction of Age of first contact. The effect of prevalence, on the predictive AFC using 3 hypothetical examples that assume different prevalence representing three levels of study. O+ and O- are the observed and P+ and P- the predicted presence and absence frequencies respectively.

**Table 3.** Prevalence = 0,75

**Table 4.** Prevalence = 0,53

**Table 5.** Prevalence =0,18

| 14 years | P+  | P- | Total | 18 years | P+  | P- | Total | 24 years | P+ | P-  | Total |
|----------|-----|----|-------|----------|-----|----|-------|----------|----|-----|-------|
| O+       | 152 | 14 | 166   | O+       | 80  | 33 | 113   | O+       | 0  | 42  | 42    |
| O-       | 52  | 5  | 57    | O-       | 46  | 52 | 98    | O-       | 14 | 167 | 181   |
| Total    | 204 | 19 | 223   | Total    | 126 | 85 | 211   | Total    | 14 | 209 | 223   |

**Table 6.** Summary Statistics of AFC derived from confusion matrix in Tables 3, 4 & 5

|                                 | Table 3 | Table 4 | Table 5 |
|---------------------------------|---------|---------|---------|
| <b>Prevalence</b>               | 0.75    | 0.53    | 0.18    |
| <b>Overall Diagnostic Power</b> | 0.25    | 0.46    | 0.81    |
| <b>Sensitivity</b>              | 0.91    | 0.70    | 0.00    |
| <b>Specificity</b>              | 0.08    | 0.53    | 0.92    |
| <b>PPP(ie.Precision)</b>        | 0.745   | 0.63    | 0.00    |
| <b>NPP(ie.Precision)</b>        | 0.26    | 0.61    | 0.8     |
| <b>Odds Ratio</b>               | 1.04    | 2.7     | 0.0     |
| <b>Kappa</b>                    | 0.0043  | 0.240   | -0.10   |

Only two measures given in Table 2 (odds-ratio, Kappa K) make full use of the information contained in the confusion matrix. The odds-ratio has the unfortunate characteristic of being infinite when either  $b$  or  $c$  are 0. Thus, it has the same value when the algorithm is perfect or lacks one type of error.

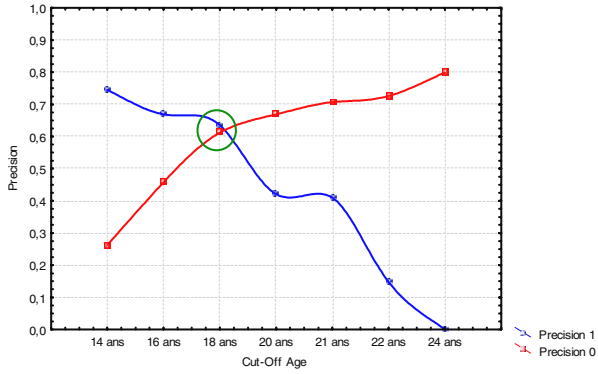
The measures described in Table 2 serve different purposes and a measure should be selected to reflect its intended use. If the aim is to assess the effectiveness of the classifier, a measure that assesses improvement over chance is appropriate, e.g. Kappa. This is important because it is possible to obtain high overall accuracy using trivial rules when, for example, prevalence is low. Indeed, overall accuracy, measured by the correct classification rate, is dependent on the prevalence ( $p$ ) since it can be rewritten as  $[(p \cdot \text{sensitivity}) + (1-p) \cdot \text{specificity}]$ . For example, if prevalence is 5% it is possible to achieve a 95% correct classification rate by labelling all cases as negative. Landis and Koch (1977) have suggested the following ranges of agreement for the Kappa statistic: poor  $K < 0.4$ ; good  $0.4 < K < 0.75$  and excellent  $K > 0.75$ .

### 3.2 Adjusting Thresholds

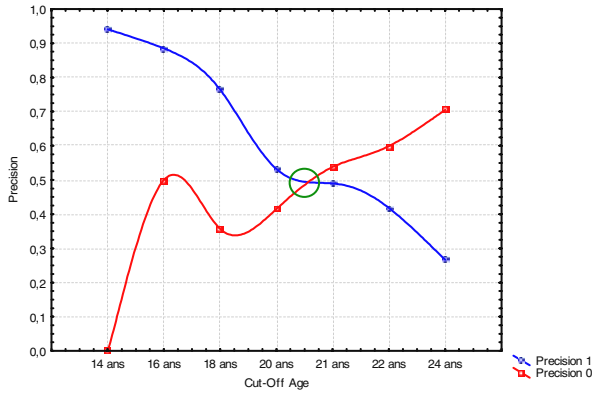
All of the measures described in this section depend on the values assigned to  $a, b, c$  &  $d$  in the confusion matrix. These values are obtained by application of a threshold criterion to a continuous variable generated by the classifier (Naïve Bayes & Bayesian Network in this case). There are a variety of reasons why the threshold value may need to be examined. For example, unequal group sizes (prevalence) can influence the scores for many of the classifier methods. This is particularly true for Naïve Bayes classifier which produces scores biased towards the larger group. Similarly, if we have decided that False Negative (FN) errors are more serious than False Positive (FP) errors the threshold can be adjusted to decrease the FN rate at the expense of an increased FP error rate. The effect of the threshold on three error rates is shown below (Fig 1 for AFC; Fig 2 for AFE and Fig 3 for NSS respectively). For example; in the first figure (AFC), at a cut-off point lower than 14 years every case is labelled as positive, while a cut-off point higher than 24 years labels every case as negative. As the cut-off point is moved from 14 to 24 years the false positive frequency falls while the false negative frequency increases. The point where these two curves cross is the point with the minimum overall error rate (assuming equal costs for false positive and false negative errors). For each of the three target variables, the most accurate statistics for the model vary among prevalence (unequal group sizes), prediction (precision), kappa, Odd-ratio...

In our data, three potential cut-offs are labelled to satisfy the different criteria and the correspondent's confusion matrix is showed in Table 7.

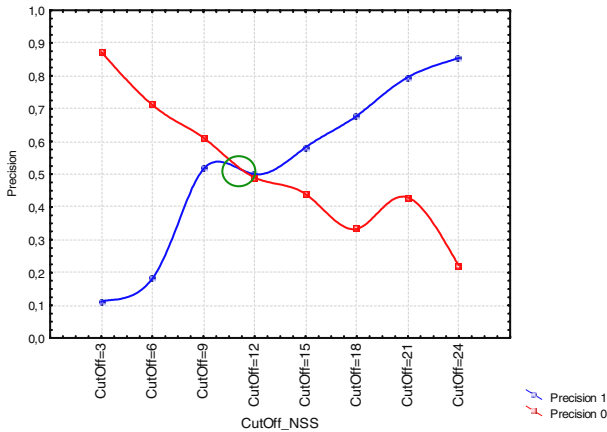
- 18 years is the 'best' cut-off for AFC (Prevalence  $\approx 0.5$ , Kappa= 0,24, odds-ratio=2,7).
- 21 years is the 'best' cut-off for AFE (Prevalence  $\approx 0.48$  & Precision for Class a and b  $\approx 0.52$ ).
- NSS=11 is the 'best' cut-off for NSS (Prevalence  $\approx 0.44$ , odds-ratio=1,41).



**Fig. 1.** Relative accuracy for different cut-Off of Age First Contact



**Fig. 2.** Relative accuracy for different cut-Off of Age First Episode



**Fig. 3.** Relative accuracy for different cut-Off of NSS



**Table 7.** Confusion matrix and Accuracy with 10 Cross Validation

|                 | Confusion Matrix                                                                                       | Correctly Classified Instances |
|-----------------|--------------------------------------------------------------------------------------------------------|--------------------------------|
|                 | a b <--classified as<br>X <sub>1</sub> X <sub>2</sub>   a = 1<br>X <sub>3</sub> X <sub>4</sub>   b = 0 |                                |
| AFC<br>18 years | a b <--classified as<br>80 33   a = 1<br>46 52   b = 0                                                 | 62.5592 %                      |
| AFE<br>21 years | a b ← classified as<br>51 55   a = 1<br>53 64   b = 0                                                  | 51.5695 %                      |
| NSS<br>≈ 11     | a b ← classified as<br>71 34   a = 1<br>49 34   b = 0                                                  | 55.8511 %                      |

## 4 Discussion

We used Bayesian networks in a set of genetics data (genotypes) to study the influence of the genotypes on clinical features (age at onset and neurological soft signs). More particularly, we used Bayesian networks to classify the three target variables *AFC*, *AFE* & *NSS* in subgroups with regards to the value of the clinical variables. Our results revealed that each of the chosen attributes influenced the classification for age at onset and neurological soft signs scores. This is in line with the fact that age at onset has been proposed as a valid clinical characteristic that helps to determine subgroups of patients that are more homogeneous regarding the origin of the disease (i.e. earlier age at onset, more genetic weight). This is also in accordance with the fact the neurological soft signs were found to follow the genetic risk [9] and that we and others have found that genotypes can be associated with age at onset [10] or *AFC* [11].

Furthermore, our results indicate that these attributes have an influence that is apparently independent since the best predictions were obtained for the naïve Bayesian models. Indeed, this model requires that attributes are (1) equally important; (2) statistically independent for a given class, meaning that the knowledge about the value of a particular attribute is not informative about the value of another attribute (if the class is known). This first result could appear to contradict the fact that we know that those two criteria are not met by the chosen attributes. Indeed, two attributes are closely related: the two polymorphisms within the same gene (*BDNF*; “E” and “E’ ”) are statistically related (i.e. in “linkage disequilibrium”). In addition, there have been reports of functional interactions between two polymorphisms (*DRD3* “A” and *BDNF* E” [11] or “E’ “ (Gourion et al, in revision). Actually, it has already been underlined that naïve Bayesian networks work surprisingly well even if independence assumption is clearly violated, since classification does not require accurate probability estimates as long as the maximum probability is assigned to the correct class [12][13][14]. Domingos and Pazzani [15] provide an explanation for the relatively good performance of Naive-Bayesian classifiers. They argue that even though Naive-

Bayesian classifiers do not estimate the underlying probability densities correctly, they provide good enough solutions in terms of zero-one loss (misclassification rate).

In a different data set and using totally independent attributes and different methodologies, we have confirmed our previous finding, that ‘age at first contact’ classification is better predicted than ‘age at first episode’ (which was itself better predicted than NSS scores). Because this result could depend on the chosen cutoff, we used a naïve Bayesian network in order to determine whether changing the cutoff for age at first episode (before or after 18 y.o) could improve the validity of classification. We chose the “best cutoff” as a methodological compromise (taking into account more different criteria) between the prevalence of classes in each data set, the percentage of accuracy (i.e. precision) and some statistical constraints (i.e. Odds-ratio & Kappa...), this approach could be applied with more complex diseases.

This study had certain methodological limitations. In particular, the information extracted from any set of data is highly dependent on the content of the initial database. The clinical data found in the present set of data are reliable information based on direct standardized “lifetime” interviews, ensuring a homogeneous and systematic manner of collecting information. In addition, data was collected from patients who have mostly been known in the department since the onset of their disease, and their case report was reviewed by a senior psychiatrist, using any available source of information (i.e. family, clinical case report). Nevertheless, age at first episode remains particularly difficult to determine, because it implies the determination of when psychotic symptoms first reached clinical significance. Though age of first contact clearly depends on accessibility of the care system and on the social or familial tolerance regarding the symptoms, it appears as a reliable variable when collecting information, since it does not rely on the rater’s opinion. Lastly, as for any assessment scale, the score on the Neurological Soft Signs Scale depends on the rater, although this scale has a good inter rater reliability between the few fully trained raters involved in the assessment of such patients.

Keeping in mind those possible limitations, our results show that “age at first contact”, more than “age at first episode”, is predictable by the genetic variation available in this data set. They further suggest that age at first contact could better reflect the actual age at onset of the pathological processes, which is underlain by genetic characteristics.

Interestingly, the attributes that were used in this study were all the available genotypes for the subjects, regardless if an association with age at onset or NSS scores was previously reported. For instance, we previously reported an association of both BDNF genotypes and age at onset, or DRD3 Bal I polymorphism was also reported to be associated with age at onset. In addition, the UTR polymorphism in the Reelin gene and a different polymorphism of the gene encoding engrailed was found to be associated with autism, a childhood onset psychiatric disease that might share some biological mechanisms with early onset forms of schizophrenia. In contrast, no association regarding age at onset was ever reported with the polymorphisms of the CB1, the 5HTT or DRD4 genes.

## 5 Conclusion

The methodology presented here allowed us to bring new information based on the present data set to the debated question of the determination of cut-offs for classifying subgroups in schizophrenia. In particular, it allowed us to propose objectively based cut-offs for age at onset and another continuous variable related to the neurodevelopmental origin of the disease.

Our results further suggest that age at first contact may better reflect the actual age at onset of the pathological processes, underlain by genetic characteristics. Since the relative weight of each individual attribute varies between AFC and AFE, our results also suggest that the three variables (AFC, AFE and NSS) could reflect different pathological processes related to different aetiological backgrounds, although all three are thought to be in relation to neurodevelopmental features. Altogether, this work supports the value of data mining methodology in complex disorders such as psychiatric disease. It also highlights the necessity of a close relationship with practitioners in order to both provide reliable information and accurate interpretations.

## References

1. Mueser K.T, McGurk S.R. Schizophrenia. *Lancet*, 363, 63-72, 2004.
2. McClure R.K, Lieberman J.A. Neurodevelopmental and neurodegenerative hypotheses of schizophrenia: a review and a critique. *Curr Opin Psychiatry*, 16 (Suppl 2):S15-S28, 2003.
3. I.I.Gottesman, Schizophrenia Genesis, W.H.Freeman, New York, 1991.
4. E.F.Torrey, Are we overestimating the genetic contribution to schizophrenia?, *Schizophr. Bull.*18, 159-170, 1992.
5. P.McGuffin et al, Genetic basis of schizophrenia, *Lancet* 346, 678-682, 1995.
6. A. Ouali et al. Phenotype Analysis in Schizophrenia using Neural Networks. IASTED International Conference on B-E, Innsbruck, Austria. February 16-18, 2005.
7. Krebs et al. Validation and factorial structure of a standardized neurological examination assessing NSS in schizophrenia. *Elsevier Schizophrenia Research* 45, 245-260, 2000.
8. D. Gourion et al. Minor physical anomalies in patients with schizophrenia and their parents. *Psychiatry Research* .Vol 125, Issue 1, 21-28, 2004.
9. D.Gourion et al, Neurological and morphological anomalies and the genetic liability to schizophrenia: a composite phenotype. *Schizophrenia Res*, 67:23-31, 2004.
10. MA. Crocq, Mant R et al. Association between schizophrenia and homozygosity at the dopamine D3 receptor gene. *J Med Genet*. Vol 29, 858-60, 1992.
11. Krebs MO et al, Brain derived neurotrophic factor (BDNF) gene variants association with age at onset and therapeutic response in schizophrenia. *Mol Psychiatry*, 5:558-62, 2000.
12. A. McCallum and K. Nigam. A comparison of event models for naive bayes text classification. In *AAAI-98 Workshop on Learning for Text Categorization*, 1998.
13. D. Lewis. Naive (bayes) at forty: The independence assumption in information retrieval. In *Tenth European Conference on Machine Learning*, 1998.
14. Y. Yang and C.G. Chute. An example-based mapping method for text categorization and retrieval. *ACM Transactions on Information Systems*, 12(3), 1994.
15. P. Domingos and M. Pazzani. Beyond independence: conditions for the optimality of the simple Bayesian classifier. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pp105-112, 1996. Morgan Kaufman Publishers.

# Graph Partitioning via Recurrent Multivalued Neural Networks

Enrique Mérida-Casermeiro and Domingo López-Rodríguez

Department of Applied Mathematics,  
University of Málaga, Málaga, Spain  
{merida, dlopez}@ctima.uma.es

**Abstract.** In this work, the well-known Graph Partitioning (GP) problem for undirected weighted graphs has been studied from two points of view: maximizing (MaxCut) or minimizing (MinCut) the cost of the cut induced in the graph by the partition. An unified model, based on a neural technique for optimization problems, has been applied to these two concrete problems. A detailed description of the model is presented, and the technique to minimize an energy function, that measures the goodness of solutions, is fully described. Some techniques to escape from local optima are presented as well. It has proved to be a very competitive and efficient algorithm, in terms of quality of solutions and computational time, when compared to the state-of-the-art methods. Some simulation results are presented in this paper, to show the comparative efficiency of the methods.

## 1 Introduction

In classical literature, the MinCut (MaxCut) problem is defined as follows: Given an undirected weighted graph  $G = (V, E)$ , where  $V = \{v_i\}$  is the set of  $N$  vertices and  $E$  is the set of  $n_e$  edges, and edge weights are given by matrix  $C = (c_{i,j})_{i,j=1,\dots,N}$  (meaning that the weight or cost of the edge joining nodes  $i$  and  $j$  is  $c_{i,j} \geq 0$ ), find a *minimum (maximum) cut* of  $G$ , i.e., a partition of  $V$  into two sets that minimizes (maximizes) the total cost of the edges with endpoints in different sets.

These problems arise in the resolution of many practical or theoretical situations. Some examples include:

- For MinCut: network reliability theory (if  $c_{i,j}$  is the probability of a network edge to fail, the minimum cut ensures the minimum risk of network disconnection) [17, ?], design of compilers (communication costs must be minimized in order to reduce the swap with memory) [4, 8].
- For MaxCut: pattern recognition, clustering, statistical physics and the design of communication networks, VLSI circuits and circuit layout [2].

So, these problems are well-known in literature. Due to their wide applicability, many variants of these problems have been formulated, placing restrictions on the original formulation.

The original problems, with all the variants, are known to be NP-complete [6], making their resolution computationally intractable, but in the case of planar graphs they belong to  $P$ , that is, there exists a solution in polynomial time. So, many algorithms have appeared to tackle GP problems in the general case.

Originally, MinCut was believed to be a variant of the  $s$ - $t$  minimum cut problem, which adds the restriction of being  $s$  and  $t$  in different sets of the partition. In the early 60's, Gomory and Hu showed that a minimum cut in  $G$  can be estimated with  $N - 1$   $s$ - $t$  minimum cut computations, see [7]. So, most of the algorithms to solve MinCut are based in the max-flow/min-cut theorem [5], which implies that a  $s$ - $t$  max-flow solution induces a  $s$ - $t$  min-cut solution. The efforts were then focused, for a long time, in solving max-flow problems.

In 1989, Nagamochi and Ibaraki [16] presented an algorithm that did not make use of max-flow computations. In 1993 and later in 1996, Karger et al. [9, 10] presented a class of randomized algorithms, based on the notion of edge contraction, that can find all minimum cuts with probability  $1 - \frac{1}{N}$ .

In the recent years, METIS [11] has become one of the most powerful algorithms for this problem, achieving the best results when compared to other methods, and in a very low computational time, as proved by [20]. To our knowledge, no neural algorithm has been presented to tackle MinCut.

In 1997, Alberti et al. presented a type-Hopfield neural model for MaxCut [1], but its performance is worse than the presented by Bertoni et al [3]. Takefuyi and his colleagues [18] developed a powerful neural model named 'maximum' and it proved to perform better than the rest of algorithms in solving a wide range of combinatorial optimization problems. Recently, Galán-Marín et al. proposed a new neural model named OCHOM which obtains much more efficient solutions than 'maximum'. Moreover, it can be used for many problems and it also has the advantage of fast convergence to a valid solution without tuning any parameter. In order to make OCHOM escape from local minima, Wang et al.[19] have recently proposed a stochastic dynamics for OCHOM, permitting temporary decreases of the objective function.

In this work, we want to present a neural model, based on a recurrent network, that has been proved to get very good results in some combinatorial optimization problems, see for example [12, 13, 14, 15], allowing  $K$ -partitioning of a graph.

Note that there exists very few bibliographic references for  $K$ -partitioning (most of the references is focused in bipartition). For MinCut, only METIS and the algorithm proposed in [8] consider that possibility, no one for MaxCut.

In the next section, we will give a detailed description of GP problem, and the two variants studied in this work, MinCut and MaxCut.

## 2 Formal Description of the Problem

Let  $G = (V, E)$  be an undirected graph without self-connections.  $V = \{v_i\}$  is the set of vertices and  $E$  is the set of  $n_e$  vertices. For each edge in  $E$  there is a weight  $c_{i,j} \in \mathbb{R}^+$ . All weights can be expressed by a symmetric real matrix  $C$ , with  $c_{i,j} = 0$  when it does not exist an arc with endpoints  $v_i$  and  $v_j$ .

**The Minimum Cut Problem (MinCut):** consists in finding a partition of  $V$  into two subsets  $A_1$  and  $A_2$ , such that  $\sum_{v_i \in A_1, v_j \in A_2, i > j} c_{i,j}$  is minimum.

**Generalization of the MinCut Problem ( $K$ -MinCut):** It looks for a partition of  $V$  into  $K$  disjoint sets  $A_i$  such that the sum of the weights of the edges from  $E$  that have their endpoints in different elements of the partition is minimum. So, the function to be minimized is

$$\sum_{v_i \in A_m, v_j \in A_n, i > j} c_{i,j} \quad (1)$$

With this formulation, the trivial solution is  $A_1 = V$ ,  $A_i = \emptyset$  for  $i = 2, \dots, K$ . So, some constraints have to be made in order to make this problem more interesting. In this work, we have considered a restriction to the cardinality of the subsets  $A_i$ : the number of nodes in each group  $A_i$  is constrained to be  $N_i$ , such that  $\sum_{i=1}^K N_i = N$ .

**MaxCut** and  **$K$ -MaxCut** are defined in a similar way: find a partition ( $K$ -partition) of  $V$  such that the cost given by the expression in (1) is maximum. Contrary to  $K$ -MinCut, there are no need for constraints in the definition of  $K$ -MaxCut.

### 3 The Neural Model

In order to solve the GP problem, we have used the MREM neural model since this model has been successfully used for other combinatorial optimization problems [12, 13, 14, 15].

**The MREM neural model:** It consists in a series of multivalued neurons, where the state of  $i$ -th neuron is characterized by its output ( $s_i$ ) that can take any value in any finite set  $\mathcal{M}$ . This set can be a non numerical one, but, in this paper, the neuron outputs only take value in  $\mathcal{M} \subset \mathbb{Z}^+$ .

The state vector  $\mathbf{S} = (s_1, s_2, \dots, s_N) \in \mathcal{M}^N$  describes the network state at any time, where  $N$  is the number of neurons in the net. Associated with any state vector, there is an energy function  $E : \mathcal{M}^N \rightarrow \mathbb{R}$ , defined by the expression:

$$E(\mathbf{S}) = \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N w_{i,j} f(s_i, s_j) \quad (2)$$

where  $W = (w_{i,j})$  is a matrix,  $f : \mathcal{M} \times \mathcal{M} \rightarrow \mathbb{R}$  is usually a similarity function since it measures the similarity between the outputs of neurons  $i$  and  $j$ . At each step, the state vector will be evolving to decrease the energy function.

To solve the GP problem with this neural net, we need as many neurons as number of nodes  $N$  in the graph. Each neuron taking value  $s_i \in \mathcal{M} = \{1, 2, \dots, K\}$  points to the subset of the partition where the  $i$ -th node is assigned to.

The cost function of the  $K$ -MinCut and  $K$ -MaxCut problems, given by (1), must be identified with the energy function of (2). So, for the general GP,  $w_{i,j} =$

$c_{i,j}$ , and  $f(x, y) = 1 - \delta_{x,y}$  (Krönecker delta function) for the  $K$ -MinCut and  $f(x, y) = \delta_{x,y}$  for the  $K$ -MaxCut, since it is equivalent to maximize the cost of the edges cut by the partition and to minimize the cost of the edges whose endpoints lie within the same group of the partition.

Initially, the state of the net is randomly selected from a subset  $\mathcal{F} \subset \mathcal{M}^N$ . At any time, the net is looking for a better solution than the current one, in terms of minimizing the energy function. To this end, multiple dynamics can be defined for the net, and we will discuss them in the next section.

## 4 Neural Implementation for GP Problem

In this work, a simple dynamics, named best-2, has been firstly implemented, and then it has been combined with two methods to improve solutions: best-3 and the shake phase.

**best-2:** It consists in getting the greatest decrease of the energy function just by changing the state of only two neurons at each time. So, a set of neighboring states must be defined. If neurons to be changed are  $p$  and  $q$ , this set will be named  $\mathcal{N}_{p,q}$ . Then, if  $\mathbf{S}(t)$  is the state of the net at time  $t$ ,  $\mathbf{S}(t+1)$  will be the vector from a  $\mathcal{N}_{p,q}$  that maximizes the decrease of energy,  $-\Delta E$ .

An expression for the decrease of energy is here given in order to reduce the computational cost of the model. Suppose that neurons  $p$  and  $q$  are going to be changed, and that we denote  $s_i(t) = s_i$  and  $s_i(t+1) = s'_i$  for all  $i$ . Then, the decrease of energy is given by:

$$U_{p,q} = -\Delta E = \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N w_{i,j} (f(s_i, s_j) - f(s'_i, s'_j)) = \sum_{i=1}^N (\Delta_{i,p} + \Delta_{i,q}) - \Delta_{p,q} \quad (3)$$

(provided the symmetry of function  $f$ ), where  $\Delta_{i,j} = w_{i,j} (f(s_i, s_j) - f(s'_i, s'_j))$ .

So, the dynamics best-2 can be summarized as follows:

1. A state for the net is initially randomly assigned.
2. Repeat until no change in state vector:
  - (a) The scheduling selects a value  $d \in \{1, \dots, \lfloor \frac{N}{2} \rfloor\}$ . For  $d > \lfloor \frac{N}{2} \rfloor$ , all of the following computations are made twice, and this way we can save some computational effort.
  - (b) The following can be made parallel: every neuron  $p$  studies all possibilities of changing neurons  $p$  and  $q = (p + d) \bmod (N)$ , with  $0 < q \leq N$ , i.e.,  $p$  computes the potential associated to the possible changes, it is stored as a vector  $\mathbf{u}_p$  whose components are the decrease of energy associated to any vector in  $\mathcal{N}_{p,q}$ , by applying (3).
  - (c) Neuron  $p$  computes  $\alpha(p) = \max \mathbf{u}_p$ , associated to a state  $\tilde{\mathbf{S}}_{p,q} \in \mathcal{N}_{p,q}$ .
  - (d) The scheduling selects the next state of the net,  $\mathbf{S}(t+1) = \tilde{\mathbf{S}}_{p,q}$  for which  $p = \arg \max \alpha$ .

In order to achieve better solutions, a pair of techniques has been developed.

**best-3:** It is an extension of best-2, allowing changes of three neurons outputs. In this case, if the neurons to be changed are  $p$ ,  $q$  and  $r$ , a neighborhood  $\mathcal{N}_{p,q,r}$  must be defined, and the associate expression for the decrease of the energy is:  $U_{p,q,r} = U_{p,q} + \sum_{i=1}^N \Delta_{i,r} - (\Delta_{p,r} + \Delta_{q,r})$ . And the dynamics of best-3 can be easily derived from best-2.

So, the scheme of the complete process is as follows: iterate best-2 until achieving a solution, then iterate best-3 once. If the solution has been improved, start with best-2 again and repeat the process.

**Shake phase:** Given an estimated solution, a good solution for MaxCut (MinCut) usually has the following property: “*High(Low)-weighted arcs must have their endpoints in different subsets*”.

So, we can study the high(low)-weighted arcs. Let  $A$  be the set of arcs with weights greater (lower) than a threshold and endpoints in the same group. Let  $V^* \subset V$  be the set of endpoints of arcs in  $A$ . Then the current solution is saved and the shake phase begins. It consists in:

- Selecting the nodes that are endpoints of arcs in  $A$  and their neighbors:  $H = \{v_i / \exists v_j \in V^*, e_{i,j} = 1\}$ .
- Nodes in  $V - H$  are clamped to their current values, while nodes in  $H$  are randomly assigned.
- With this new initial state vector, the network evolves with the usual dynamics (best-2), but only nodes in  $H$  will be selected in order to be modified, until a new stable state is reached.
- This new solution is compared with the previous saved one and the best one is selected.

Although the shake method can be used to improve the solutions of both  $K$ -MaxCut and  $K$ -MinCut, in this work it has been tested for  $K$ -MaxCut, as an example.

So, we must concrete the definitions of  $\mathcal{F}$  and the neighbors  $\mathcal{N}_{p,q}$  and  $\mathcal{N}_{p,q,r}$  for each problem ( $K$ -MinCut and  $K$ -MaxCut):

– **For  $K$ -MinCut:**

1.  $\mathcal{F} = \{\mathbf{S} \in \mathcal{M}^N : \text{ exactly } N_i \text{ of its components are equal to } i, \forall i = 1, \dots, K\}$  is the set of feasible solutions for this problem. Therefore, the net must begin in a feasible state and must be kept inside  $\mathcal{F}$  in any time.
2. **best-2:** The only alternative for changing the states of neurons  $p$  and  $q$ , and to remain in  $\mathcal{F}$  is swapping their outputs. So, if  $S$  is the current state vector of the net, we have  $\mathcal{N}_{p,q} = \{\mathbf{S}, \overline{\mathbf{S}}\}$ , where  $\overline{\mathbf{S}}(p) = \mathbf{S}(q)$ ,  $\overline{\mathbf{S}}(q) = \mathbf{S}(p)$ , and  $\overline{\mathbf{S}}(m) = \mathbf{S}(m)$  for all  $m \notin \{p, q\}$ . Note that in this case, we have  $\Delta_{p,q} = w_{p,q} (f(s_p, s_q) - f(s_q, s_p)) = 0$ .

**best-3:** Something similar occurs when dealing with  $\mathcal{N}_{p,q,r}$ : only some permutations of the outputs of neurons  $p$ ,  $q$  and  $r$  are allowed because many of them are included in some  $\mathcal{N}_{p,q}$ . This gives  $\mathcal{N}_{p,q,r} = \{\mathbf{S}, \mathbf{S}_1, \mathbf{S}_2\}$ , where  $\mathbf{S}(m) = \mathbf{S}_1(m) = \mathbf{S}_2(m)$  for  $m \notin \{p, q, r\}$ , and  $\mathbf{S}_1(r) = \mathbf{S}_2(q) = \mathbf{S}(p)$ ,  $\mathbf{S}_1(p) = \mathbf{S}_2(r) = \mathbf{S}(q)$  and  $\mathbf{S}_1(q) = \mathbf{S}_2(p) = \mathbf{S}(r)$ .



– **For  $K$ -MaxCut:**

1. Now,  $\mathcal{F} = \mathcal{M}^N$ , there are no restrictions in this case, so any state is feasible.
2. The neighboring states are now defined as in the most general case: if current state is  $S$ ,  $\mathcal{N}_{p,q}$  and  $\mathcal{N}_{p,q,r}$  include all possible states from  $\mathcal{M}^N$  that differ from  $S$  only in the outputs of neurons  $p$  or  $q$  (or both), and  $p, q$  or  $r$ , respectively. There are  $K^2$  vectors in  $\mathcal{N}_{p,q}$ , and  $K^3$  in  $\mathcal{N}_{p,q,r}$ .

Some experimental results for the dynamics herein proposed are shown in the next section.

## 5 Simulation Results

In this work, we have tested our algorithms with the two problems exposed in the text.

A test set was formed by 240 random graphs depending on two parameters,  $N \in \{20, 50, 80, 100\}$  (the cardinality of the set of vertices), and  $\rho \in \{0.05, 0.15, 0.25\}$  (the density of edges in the graph, meaning that  $n_e \approx \rho \frac{N(N-1)}{2}$ ). Weights for edges were integers randomly chosen in  $[0, 5]$ . For this set to be complete, the values for the parameters were chosen to cover a wide range of graphs.

For  $K$ -MinCut, we have compared our model with METIS [11], every algorithm implemented in MatLab on a Pentium IV (3.06 Ghz). In Table 1, column labelled BEST-2 shows the results of applying only the dynamics best-2, and the column labelled BEST-3, presents the results of applying the composition of best-2 and best-3, as exposed in Sec. 4.

It can be verified that our model outperforms METIS in many cases. Note that METIS is a heuristic that always produces the same solution, while the repetitive use of best-2, best-3 always obtains good solutions that can be improved with new executions.

**Table 1.** Best and average performance on test set over 10 runs

| $N$ | $\rho$ | BEST-2 |         |        | BEST-3 |        |        | METIS    |        |
|-----|--------|--------|---------|--------|--------|--------|--------|----------|--------|
|     |        | Best   | Av.     | t      | Best   | Av.    | t      | Best/Av. | t      |
| 20  | 0.05   | 0      | 0.405   | 0.0006 | 0      | 0.395  | 0.0111 | 0        | 0.016  |
| 20  | 0.15   | 6.3    | 8.945   | 0.0007 | 6.05   | 8.21   | 0.012  | 7.15     | 0.0005 |
| 20  | 0.25   | 18.2   | 21.75   | 0.0007 | 17.75  | 19.965 | 0.0132 | 19.4     | 0.0005 |
| 50  | 0.05   | 9.35   | 14.145  | 0.0021 | 8.9    | 13.98  | 0.0928 | 9        | 0.0005 |
| 50  | 0.15   | 86.1   | 95.64   | 0.0019 | 81.75  | 91.435 | 0.1283 | 90.8     | 0      |
| 50  | 0.25   | 180.25 | 194.025 | 0.0023 | 174.5  | 186.38 | 0.1428 | 187.75   | 0.002  |
| 80  | 0.05   | 42.5   | 51.665  | 0.0032 | 40.8   | 50.35  | 0.2974 | 47.95    | 0.0005 |
| 80  | 0.15   | 269.95 | 286.61  | 0.0039 | 263.2  | 279.24 | 0.3998 | 283.05   | 0.0015 |
| 80  | 0.25   | 535.7  | 556.09  | 0.003  | 526.2  | 544.54 | 0.4663 | 551.6    | 0.0035 |
| 100 | 0.05   | 82.8   | 95.66   | 0.0046 | 80.3   | 93.725 | 0.5178 | 86.55    | 0      |
| 100 | 0.15   | 452.1  | 472.58  | 0.0046 | 440.95 | 461.68 | 0.7451 | 472.6    | 0.003  |
| 100 | 0.25   | 869.5  | 898.165 | 0.0047 | 856.75 | 883.09 | 0.8182 | 902.5    | 0.004  |

**Table 2.** Best and average performance on test set

| $N$ | $\rho$ | Wang |       |       | OCHOM |        |       | MREM |        |       | MREM-shake |        |       |
|-----|--------|------|-------|-------|-------|--------|-------|------|--------|-------|------------|--------|-------|
|     |        | Best | Av.   | t     | Best  | Av.    | t     | Best | Av.    | t     | Best       | Av.    | t     |
| 20  | 0.05   | 26   | 21.8  | 0.003 | 26    | 24.4   | 0.001 | 26   | 25.4   | 0.024 | 26         | 25.4   | 0.023 |
| 20  | 0.15   | 67   | 29.0  | 0.002 | 69    | 66.5   | 0.001 | 69   | 68.0   | 0.027 | 69         | 68.0   | 0.027 |
| 20  | 0.25   | 80   | 63.6  | 0.002 | 86    | 78.5   | 0.001 | 86   | 84.4   | 0.024 | 86         | 84.4   | 0.025 |
| 50  | 0.05   | 144  | 113.4 | 0.016 | 142   | 137.4  | 0.005 | 149  | 143.5  | 0.243 | 149        | 143.5  | 0.265 |
| 50  | 0.15   | 278  | 248.8 | 0.015 | 273   | 264.6  | 0.005 | 284  | 276.7  | 0.234 | 284        | 277.0  | 0.369 |
| 50  | 0.25   | 460  | 397.9 | 0.012 | 476   | 448.8  | 0.006 | 469  | 460.6  | 0.244 | 472        | 463.9  | 0.482 |
| 80  | 0.05   | 270  | 238.0 | 0.031 | 266   | 258.6  | 0.011 | 279  | 271.5  | 0.713 | 279        | 271.5  | 1.025 |
| 80  | 0.15   | 715  | 702.5 | 0.034 | 739   | 712.4  | 0.014 | 754  | 735.3  | 0.943 | 754        | 742.2  | 1.954 |
| 80  | 0.25   | 1100 | 878.2 | 0.034 | 1106  | 1080.5 | 0.016 | 1117 | 1091.0 | 0.857 | 1117       | 1095.1 | 1.717 |
| 100 | 0.05   | 400  | 323.7 | 0.048 | 407   | 390.2  | 0.017 | 418  | 406.0  | 1.539 | 418        | 406.6  | 2.374 |
| 100 | 0.15   | 1071 | 843.6 | 0.068 | 1060  | 1029.1 | 0.023 | 1081 | 1062.3 | 1.629 | 1084       | 1068.5 | 3.257 |
| 100 | 0.25   | 1697 | 834.4 | 0.043 | 1728  | 1682.3 | 0.025 | 1741 | 1702.7 | 1.323 | 1741       | 1714.8 | 2.407 |

With respect to  $K$ -MaxCut simulations, we have compared our proposed algorithms to OCHOM and Wang's. All of them have been implemented and tested in MATLAB, on the same conditions as above. More specifically, Wang's network has been tested with its default parameter  $\lambda = 30$ . In the proposed model, the set  $A$  was built by including every edge  $e_{i,j}$  whose cost  $c_{i,j} > \bar{c} + 3\sigma$ , where  $\bar{c}$ ,  $\sigma$  are respectively the mean and the standard deviation of  $c_{i,j}$ . So,  $A$  is forced to include exclusively high-weighted edges.

Both best and average solutions obtained are shown in Table 2. So, we can verify that the proposed algorithm outperforms others, not only giving the best results, but even on average.

## 6 Conclusions

The aim of this work has been to present a neural model for the resolution of combinatorial optimization problems. In particular, it has been proved to be a good optimizer for some NP-complete problems, as seen in [12, 13, 14, 15].

Contrary to heuristics, producing always the same solution to the problem, with the neural approach developed in this work, improvement of solutions is feasible, because the initial state of the net can be changed in each execution, and the search for the optimum begins from a different point in the search space, avoiding some local optima.

Another important feature that is present in the model is that it allows the  $K$ -partitioning of graphs, while some other techniques are based in the bipartition. So, this model is applicable to more general situations than some other methods.

Two important techniques to escape for local optima have been exposed in this paper. When combined to the original dynamics, best-2, they improve substantially the quality of the achieved solution.

To end with, the parallelism included in the computation dynamics is a powerful tool to achieve very good results with very little time consumption.

## References

1. A. Alberti, A. Bertoni, P. Campadelli, G. Grossi and R. Posenato. A neural algorithm for MAX-2SAT: performance analysis and circuit implementation, *Neural Networks* **10-3** (1997), 555-560.
2. F. Barahona, M. Grotschel, M. Junger and G. Reinelt, An Application of combinatorial optimization to statistical physics and circuit layout design. *Operat. Research* **36** (1988), 493-513.
3. A. Bertoni, P. Campadelli and G. Grossi, An approximation algorithm for the maximum cut problem and its experimental analysis. *Proceedings: Algorithms and experiments. Trento*, **9-11** (1998), 137-143.
4. S. Chatterjee, J. R. Gilbert, R. Schreiber, and T. J. Sheffler. Array Distribution in Data-Parallel Programs. In *Languages and Compilers for Parallel Computing*, Lecture Notes in Computer Science series, Springer-Verlag, **896** (1996), 76-91.
5. P. Elias, A. Feinstein, and C. E. Shannon. Note on Maximum Flow Through a Network. *IRE Transactions on Information Theory*, **IT-2** (1956), 117-199.
6. M.R. Garey and D.S. Johnson, *Computers and Intractability. A guide to the theory of NP-Completeness*. W. H. Freeman and Company, New York (1979).
7. R. Gomory and T. C. Hu, Multi-terminal network flows, *J. SIAM*, **9** (1961), 551-570.
8. E. J. L. Johnson, A. Mehrotra and G. L. Nemhauser. Min-cut clustering. *Mathematical Programming*, **62 (1)** (1993), 133-151.
9. D. R. Karger. Minimum cuts in near-linear time. *Proc. 28th Annual ACM Symposium on Theory of Computing*, (1996) 56-63.
10. D. R. Karger and C. Stein. A new approach to the minimum cut problem. *J. Assoc. Comput. Mach.*, **43(4)** (1996), 601-640.
11. G. Karypis and V. Kumar. Multilevel  $k$ -way partitioning scheme for irregular graphs. *Journal of Parallel and Distributed Computing*, **48(1)** (1998), 96-129.
12. D. López-Rodríguez and E. Mérida-Casermeiro. Matrix Bandwidth Minimization: A Neural Approach, in *Proceedings of International Conference of Computational Methods in Science and Engineering, ICCMSE*, **1** (2004), 324-327.
13. E. Mérida-Casermeiro, G. Galán-Marín and J. Muñoz-Pérez. An Efficient Multivalued Hopfield Network for the Traveling Salesman Problem. *Neural Processing Letters* **14** (2001), 203-216.
14. E. Mérida-Casermeiro, J. Muñoz-Pérez and R. Benítez-Rochel. Neural Implementation of Dijkstra's Algorithm. *LNCS* **2686** (2003), 342-349.
15. E. Mérida-Casermeiro and D. López-Rodríguez. Multivalued Neural Network for Graph MaxCut Problem, in *Proceedings of International Conference of Computational Methods in Science and Engineering, ICCMSE*, **1** (2004), 375-378.
16. H. Nagamochi and T. Ibaraki. Computing Edge-Connectivity in Multigraphs and Capacitated Graphs. *SIAM J. Disc. Meth.*, **5** (1992), 54-66.
17. A. Ramanathan and C. Colbourn. Counting Almost Minimum Cutsets with Reliability Applications. *Math. Prog.*, **39** (1987) 253-261.
18. Y. Takefuyi and J. Wang, *Neural computing for optimization and combinatorics*. Singapore, World Scientific, **3**, (1996).
19. Jiahai Wang and Zheng Tang. An improved optimal competitive Hopfield network for bipartite subgraph problems. *Neurocomputing* (In press).
20. D. Yang, Y. Chung, C. Chen and C. Liao. A Dynamic Diffusion Optimization Method for Irregular Finite Element Graph Partitioning, *The Journal of Supercomputing*, Kluwer Academic Publishers, **17**, (2000), 91-110.

# Dynamical Random Neural Network Approach to a Problem of Optimal Resource Allocation

YongJun Zhong<sup>1</sup>, DonChuan Sun<sup>2</sup>, and JianJun Wu<sup>3</sup>

<sup>1</sup> School of Business Administration, South China University of Technology,  
510640 Guangzhou, China

becoo@126.com

<sup>2</sup> The Zhuhai College of Jinan University, 519070 Zhuhai, China

dchsun@jnu.edu.cn

<sup>3</sup> The Department of Computer Science, Hexi University,  
734000 Zhangye, China

zyszwj@163.com

**Abstract.** Dynamical Random Neural Network (DRNN) has been suggested as tools for the solution of optimization problems [1, 2]. Here DRNN method is applied to solve the problem of optimal resource allocation with both minimum and maximum activation levels and fixed cost. The problem is NP-hard. The conclusion shows that the DRNN method provides results of the optimal resource allocation problem better than those given by [3].

## 1 Introduction

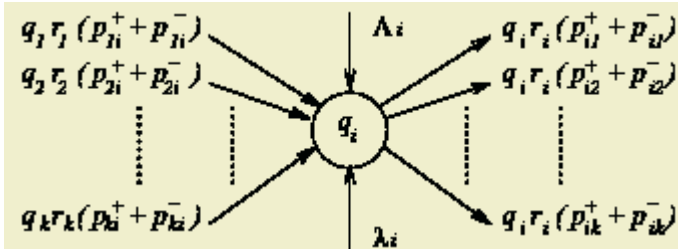
This paper analyzes a problem of optimal resource allocation. A pre-determined quantity of a particular resource can be allocated to a number of alternative projects. In comparison with other simpler allocation problems, additional constraints are considered in order to impose restrictions on the minimum and maximum activation levels for the projects to be undertaken. Moreover, both the resource requirements and the projects are assumed to a fixed part that is independent of the activation of the project level and a variable part that is proportional to this level. The problem consists in maximizing the total profit by satisfying simultaneously all the constraints and is presented as a mixed integer linear programming problem [3].

In this paper, the Dynamical Random Neural Network (DRNN) is approached to the recourse allocation problem, which originates with the Random Neural Network (RN) model. As a result, a very satisfactory solution has been obtained.

In the sequel, first the RN model will briefly be recalled in Section 2. Then, Section 3 introduces the basic ideas related to the DRNN method. Section 4 describes a resource allocation model. Section 5 is devoted to a presentation and discussion of the DRNN solution to the resource allocation problem. Finally, conclusion is drawn in Section 6.

## 2 Random Neural Network Model

In the random neural network model signals in the form of spikes of unit amplitude circulate among the neurons. Positive signals represent excitation and negative signals represent inhibition among a set of neurons. Each neuron's state is a non-negative integer called its potential. This increases when an excitation signal arrives at it and decreases when an inhibition signal arrives. Figure 1 shows a single neuron representation in RNN [4].



**Fig. 1.** Representation of a neuron in the RNN, Where  $r_i$  is the  $i$ -th neuron's firing rate when it is stimulated,  $p_{ji}^+$ ,  $p_{ji}^-$  are the probabilities that neuron  $j$  sends to neuron  $i$  excitatory(+) or inhibitory(-) signals:  $\Lambda_i, \lambda_i$  are the positive and negative external signal arrival rates to neuron  $i$

In random neural network model, firing occurs when the state of the neuron is strictly positive. The significant quantity in this model is the probability  $q_i$  that the neuron  $i$  is excited. It is computed from the positive and negative and negative signal arrival rates:

$$q_i = \frac{\Lambda_i + \sum_{j=1}^N q_j r_j P_{ji}^+}{r_i + \lambda_i + \sum_{j=1}^N q_j r_j P_{ji}^-}, i = 1, \dots, N \tag{1}$$

This is a frequency modulated model inspired by biophysical neuronal activity. The signals are spikes of unit amplitude, and the quantities which appear in the numerator and denominator of (1) are the frequencies at which excitation or inhibition spikes arrive at the neuron, as well as the characteristic or firing frequency  $r_i$  of the neuron.

## 3 The Dynamical Random Neural Network Model

The Dynamical Random Neural Network method consists to construct dynamical systems by using random neurons as basic components. DRNN method considers the excitation probability expression (1) as the instantaneous input-outputs relation of formal neurons. The variables  $A_i$  and  $q_i$  constitute respectively the inputs and the

outputs of the so-defined limit system. Indeed DRNN uses only positive inputs and inhibitory type interactions between neurons, *i.e.* for all  $i, j, \lambda_i = 0, P_{ji}^+ = 0$  (see Figure 2). Therefore interconnected random neurons can be output vector  $q$ .

In order to allow a dynamical behavior to the random neural network, the DRNN model assumes additional time-delayed feedbacks defined by a Cohen-Grossberg like equation:

$$\frac{d\Lambda_i}{dt} = A(q_i) \left[ B(q_i) - \frac{\partial F(q)}{\partial q_i} \right], i = 1, \dots, N \tag{2}$$

Where  $F(q)$  is a penalty function which will be optimized: it depends on the vector  $q=(q_1, \dots, q_N)$ , the  $q_i$  are the excitation probabilities of the neurons,  $A(q_i)$  is the gain function that control the convergence rate,  $B(q_i)$  is the decay function that allows us to place the attractors in appropriate positions of the state space.

The equation (2) shows that the dynamical variables of this network are the neuron inputs not as much as that of connectionist methods where the external inputs are kept constant once they had initially been set. The time variation of  $\Lambda_i$  depends on the vector  $\Lambda$  through an implicit form described by the non-linear transformation (1).

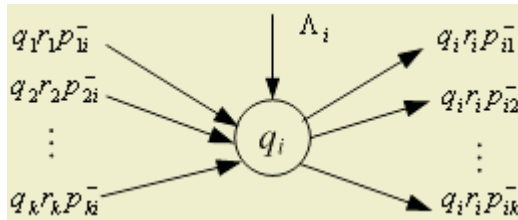


Fig. 2. Representation of a neuron in the DRNN, where  $\lambda_i = 0, P_{ji}^+ = 0$

### 4 Model

The following resource allocation problem is described in [3]. Suppose that a given resource is available at a level  $b$  and can be allocated to  $M$  projects. Assume that project  $r$  can be discarded or activated at a level  $a_r$  not lower than a minimum fraction  $\beta_r$ . Define the binary activation  $x_r, r=1, 2, \dots, M$ , which take value 1 if the  $r$ th project is activated and value 0 otherwise. Note that if  $\beta_r > 0$ , *i.e.*, if a minimum activation level is required to undertake project  $r$ , then  $0 \leq a_r < \beta_r$  entails the rejection of the project ( $x_r=0$ ).

Further assume that the resources need to activate project  $r$  is given by the sum of a fixed amount  $s_r \geq 0$  and a variable quantity  $t_r x_r$  that is proportional to the activation level using the coefficient  $t_r > 0$ . A budgetary constraint station that the allocated resources are not greater than  $b$  can be presented as follows ( $s_r=1$  whenever  $a_r > 0$ ):

$$\sum_{r=1}^M x_r (s_r + \alpha_r t_r) = \sum_{r=1}^M x_r s_r + \alpha_r t_r \leq b. \tag{3}$$

Moreover, let  $h_r + \alpha_r k_r$ , where  $h_r$  and  $k_r$  are real coefficients, are the profit obtained if the  $r$ th project is activated at a level  $\alpha_r$ , while if the project is not started the associated profit is equal to zero. In particular, if negative,  $h_r$  could represent the quantity of fixed costs incurred by project  $r$  irrespective of its activation level, while  $k_r$  is the variable part of the profit, proportional to the activation level. The total profit is defined as the sum of the profits of the activated projects:

$$\sum_{r=1}^M x_r (h_r + \alpha_r k_r) = \sum_{r=1}^M x_r h_r + \alpha_r k_r. \tag{4}$$

The problem consists in maximizing the total profit (4) subject to the budget constraint and to the additional constraints that guarantee the feasibility of the pairs  $(x_r, \alpha_r)$ :

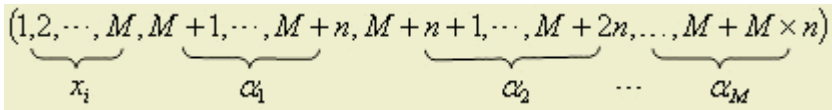
$$\max_{x_1, \dots, x_M, \alpha_1, \dots, \alpha_M} \sum_{r=1}^M x_r h_r + \alpha_r k_r, \tag{5}$$

Subject to

$$\begin{aligned} \sum_{r=1}^M x_r s_r + \alpha_r t_r &\leq b \\ \alpha_r - x_r &\leq 0, r = 1, 2, \dots, M, \\ \alpha_r - \beta_r + (1 - x_r) &\geq 0, r = 1, 2, \dots, M, \\ 0 \leq \alpha_r \leq 1, r &= 1, 2, \dots, M, \\ x_r \in \{0, 1\}, r &= 1, 2, \dots, M, \end{aligned} \tag{6}$$

### 5 The DRNN Approach to the Problem of Resource Allocation

In the problem of resource allocation, a set of  $M+M \times n$  neurons  $q$  is used to code the problem, where  $2^n > \max \{ \alpha_r \} / e + 1$ .  $e$  is the minimum error of  $\alpha_r$  which is requested. The code of neurons is as follows:



**Fig. 3.** The figure expresses the code of neurons about the problem of resource allocation, where the value of neuron  $i$  is  $d_i$

The pairs  $(x_r, \alpha_r)$  can be formulated as follows:

$$x_r = d_i, r=1, 2, \dots, M; \alpha_r = \sum_{i=1}^n d_{M+(n-1) \times r + i} 2^{-i}, r=M+1, \dots, M+M \times n \tag{7}$$

Furthermore, the firing rate  $r_i$  of neuron  $i$  depends on the input-output ratio of the project  $j$  where neurons  $i$  belong to the code of project  $j$ . It is as follows:

$$r_i = \begin{cases} \frac{s_i + \alpha_i t_i}{h_i + k_i \alpha_i}, i = 1, 2, \dots, M \\ \frac{s_{[(i-M)/n]} + \alpha_{[(i-M)/n]} t_{[(i-M)/n]}}{h_{[(i-M)/n]} + k_{[(i-M)/n]} \alpha_{[(i-M)/n]}}, i > M \end{cases} \quad (8)$$

Thus

$$p_{ij}^- = \begin{cases} 1/(r_i \times (M - 1)), j \leq M, j \neq i \\ 1/(r_i \times (M * (n - 1))), j > M, j \notin L \\ 0, otherwise \end{cases} \quad (9)$$

Where  $L$  denote a set of  $([(i-M)/n]+1, [(i-M)/n]+2, \dots, [(i-M)/n]+n)$ .

Equation (9) show  $p_{ij}^-$  is the function of  $r_i$ . If project  $j$  can provide more input-output ratio than other project, neurons that belong to the code of project  $j$  will have a smaller firing rate  $r_i$  and send stronger inhibitory signals to any other neurons. It is more efficient for find an optimum solution for resource allocation problem.

Excitatory links and negative inputs are null: ( $p_{ij}^+ = 0, \lambda_i = 0$ ). The parameter  $A(q_i)$  for the dynamics is chosen by:  $A(q_i) = 1 + q_i$ . On the other hand  $B(q_i)$  is chosen as a suited value so as to provide the appropriate sign in the derivative of equation (2), where it is chosen by:  $B(q_i) = -q_i \sum_{r=1}^M (h_r + a_r) / M$ . The penalty function  $F(q)$  with the objective and constraints weights  $\eta_1, \eta_2, \eta_3$  and  $\eta_4$  is chosen as follows:

$$F(q) = -\eta_1 \left( \sum_{r=1}^M x_r h_r + k_r \alpha_r \right) + \eta_2 \left( \max \left( 0, \sum_{r=1}^M x_r s_r + t_r \alpha_r - b \right)^2 \right) + \eta_3 \left( \sum_{r=1}^M \left( \min \left( 0, \alpha_r - q_r \right) \right)^2 \right) + \eta_4 \left( \sum_{r=1}^M \left( \max \left( 0, \alpha_r - \beta_r + (1 - x_r) \right) \right)^2 \right) \quad (10)$$

Where  $\eta_1 = \eta_2 = \eta_3 = \eta_4 = 1$ . Obviously, function (10) is a piecewise function which is differentiable.

The algorithm can be summarized as follows:

Initialize  $q_i^0, P_{ij}^-$ , set  $t=0$ ;

While not converge do

{ Calculate  $r_i, P_{ij}^-$  for all neurons  $q_i$ ;

$$q_i^{t+1} = \frac{\Lambda_i^t}{r_i + \lambda_i + \sum_{j=1}^N q_j r_j P_{ji}^-}, \forall i;$$



Calculate  $\Lambda_i^{t+1}$  from (2) (10),  $\forall i$  ;

T=t+1;}

In order to test the performance of the DRNN method approach to the resource allocation problem, the results have been obtained by solving 50 simulated instances of 10 projects (M=10). Coefficients of instance have been simulated as independent uniform random such that:  $-10 \leq h_r \leq 0$ ,  $0 \leq k_r \leq 50$ ,  $10 \leq t_r \leq 40$ ,  $s_r=0$ ;  $b=100$ ;  $\beta_r=0.35$ . For each instance, 4 successive tests were run. The performance of DRNN can be evaluated by the quality of the best solution ( $\varphi_k$  is the percentage of optimum solutions found by the DRNN up to  $k$ th trial where  $k=1,2,3,4$  ) and the average relative error ( $\varepsilon_k$ ) of the DRNN with respect to the optimum for the last solution obtained up to  $k$ th trial. The results are summarized in the Table 1 and 2.

**Table 1.** Optimal solution percentage for allocation problem with DRNN model

| Iterance | $\varphi_1$ | $\varphi_2$ | $\varphi_3$ | $\varphi_4$ |
|----------|-------------|-------------|-------------|-------------|
| 500      | 48.2        | 53.1        | 70.4        | 76.1        |

**Table 2.** Average relative error for allocation problem with DRNN model

| Iterance | $\varepsilon_1$ | $\varepsilon_2$ | $\varepsilon_3$ | $\varepsilon_4$ |
|----------|-----------------|-----------------|-----------------|-----------------|
| 500      | 1.3             | 1.1             | 0.6             | 0.4             |

## 6 Conclusion

The DRNN method is to assume an output dependent dynamic behavior for the input variables of Gelenbe’s stochastic neural network. It is efficient to solve NP-hard problem. In this paper, the Dynamic Random Neural network model to solve the resource allocation problem completely.

The result indicated that The DRNN model can approach to the optimal resource allocation problem which is NP-hard. Here DRNN used only positive inputs and inhibitory signal. Like the survival of the fittest in the biological evolution, the firing rate and the intensity of inhibitory signal depended on the input-output ratio of the project. As it improves, the DRNN model can quickly find an optimum solution.

## References

1. L.Weaver and T. Bossomaier. Evolution of Neural Networks to Play the Game of Dots-and-Boxes. In *Alife V: Poster Presentations*, May 16-18 (1996)
2. Gelenbe, E.; Koubi, V.; Pekergin, F. Dynamical Random Neural Network approach to the Traveling Salesman Problem. *Elektrik, Turkish Journal of Electrical Engineering & Computer Sciences*, Vol. 2, (1994)

3. Antomella Basso, Lorenzo A. Peccati. Optimal resource allocation with minimum activation levels and fixed cost. *European Journal of Operational Research* 131 (2001)
4. H. M. Abdelbaki, Random Neural Network Simulator for Use with Matlab, Technical Report, University of Central Florida, September, (1999)

# Biometric Hand Recognition Using Neural Networks

Francisco Martínez, Carlos Orrite, and Elías Herrero

Aragon Institute for Engineering Research,  
University of Zaragoza, María de Luna 1, 50018 Zaragoza, Spain  
{franjmar, corrite, jeliias}@unizar.es  
<http://www.cv.i3a.unizar.es>

**Abstract.** A new approach for personal identification using hand geometry based upon geometrical and shape features is presented. We propose a new pegless hand geometry verification system where the users are free to put their hand in arbitrary fashion. A Linear Discriminant Analysis is applied to the raw data in order to perform a best clustering of the feature space. The combination of three different neural network classifiers (unsupervised SOM, supervised SOM and LVQ) gives 0.35% FAR and 0.15% FRR. The method has been tested on a large size database of 1400 images for training and 1400 for test from 280 individuals suitable for medium and low security applications.

## 1 Introduction

Biometrics refers to automatic people recognition based on behavioural and/or physiological characteristics. A brief overview of the field of biometrics summarizing some of its advantages, disadvantages, strengths, limitations, and related privacy concerns can be consulted in [1]. Different biometric such as fingerprints, hand geometry, face, voice, etc., can be used for reliable authentication of individuals. Among all, hand geometry can be considered the most suitable modality for medium and low-level security applications [6].

There have been several hand geometry systems for biometric verification published in literature. These systems can be broadly classified into two categories: pegged systems where pegs are used to fix the placement of the hand; and pegless systems where the acquiring hand geometry process is carried out in absence of pegs and therefore users can place their hands in arbitrary fashion. Examples of pegged hand geometry system are for instance, the system developed by Raul Sanchez-Reillo et. al. [2]. They employed a feature selection and different pattern recognition techniques, concluding that Gaussian Mixture Models (GMMs) gives the best performance. Jain et al. [4] also developed a pegged hand geometry verification system used as prototype for web security applications. Later, Jain and Duta [7] developed another pegged verification system which perform post processing to remove the pegs and aligns finger contours and measure the mean alignment error between them.

Examples of pegless systems are given by Oden et. al. [8], who developed a system for identification and verification applications using implicit polynomials. Wong and Shi [10] proposed a feature-based framework for hand geometry recognition, based

upon matching of geometrical and shape features. They employed a GMM for the first group of features, followed by a distance metric classification of the second group of features if necessary. Kumar et. al. [3] developed a personal verification pegless system using palmprint and hand geometry features. Bulatov et. al. [6] developed a hand geometry recognition system for verification, classification, and identification of individuals. More recently, Chikkerur et. al. [9] have proposed a pegless hand geometry verification system through biometric hardening. This technique increases the entropy of the hand geometry template by using user specific random keys.

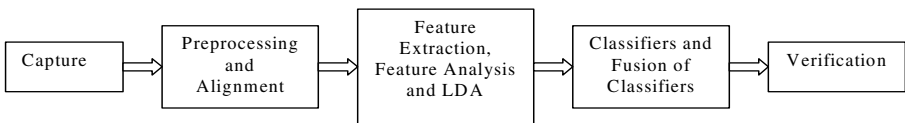
In this paper, we present a novel hand geometry recognition system that base upon a peg-free hand image acquisition and combining several neural network based classifiers over a transformed geometrical feature space by means of a Linear Discriminant Analysis.

## 2 System Overview

Most of the previous work on hand geometry recognition used pegs or similar mechanism to guide the hand in a consistent position. The method here proposed used images obtained from a document scanner or a digital camera, so users were free to put their hands anywhere. The only restriction was to keep their fingers separated.

To test the system we used a public database [5] of 5600 images obtained from 280 users with 10 sample of each hand per person. We reduced the biometric recognition only to the right hand. Off these 10 right-hand samples, 5 samples are used for training and 5 samples are used for testing purposes.

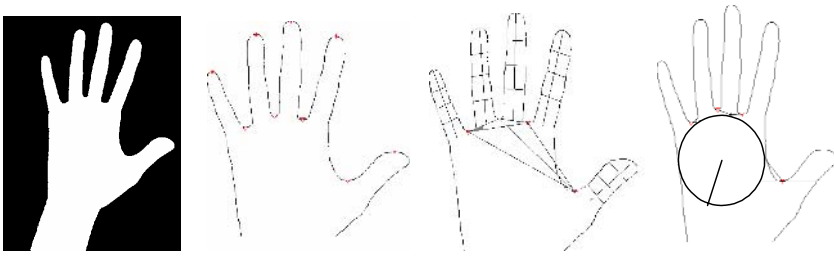
Fig. 1 shows the hand recognition diagram proposed. Every captured grey level image is binarized and aligned to extract hand geometry features. Then, we use a Linear Discriminant Analysis (LDA) to perform a best clustering of the features from the point of view of classification rather than feature reduction. Over this transformed space three different neural network-base classifiers are applied, and the combination of them allows improving the verification process.



**Fig. 1.** Hand recognition diagram

## 3 Feature Extraction

After the image is captured, some pre-processing is performed. The first step is to transform the RGB image into a grey-scale one. After that, the image thresholding operation is used to obtain a binary image. The threshold value is computed by Otsu's method [11]. With this image is easily to extract the hand boundary, and some geometry landmarks like the fingertips points and the valley points between adjacent fingers.



**Fig. 2.** Different steps in feature extraction: binarization; valley and fingertips extraction; geometrical features: widths, heights, distances and angles

These geometry landmarks are used to align the hand with respect to the vertical. The second and third valleys are used to form a line. The mid-point of this line and the middle fingertip form the reference axis from as a rotation angle is calculated and the transformation takes place following equation (1):

$$\begin{bmatrix} x_{new} \\ y_{new} \end{bmatrix} = \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} \tag{1}$$

Once the hand has been aligned a pattern of features is extracted. The different measurements are:

**Widths:** Each finger is measured in three heights. The greater and minor widths of the palm are also measured.

**Heights:** For each finger, the fingertip point and the middle point of its baseline determine its finger length. The baseline is a line formed by a valley point and the point which has the same distance from the fingertip to the other side of the boundary of the finger.

**Distances:** Distances between valleys and distances of each valley to the centre of mass of the hand (only until the minor width of the palm). Largest inscribed circle radius is also measured.

**Angles:** between the valleys points.

In this way, 34 features were extracted from 280 users with 5 photographs from each user. After that, a ratio F was calculated to determine the discriminability of the features. This ratio is defined as the quotient between the interclass and the intraclass variability.

$$F_j = \frac{\text{Interclass Variability}}{\text{Intraclass Variability}} = \frac{V\left(\frac{1}{N} \sum_{i=1}^N \overline{f_j^i}\right)}{\frac{1}{N} \sum_{i=1}^N V(f_j^i)} \tag{2}$$

where  $F_j$  is the ratio for the  $j$ th feature,  $V$  is the standard deviation function,  $N$  is the number of users,  $f_j^i$  is the  $j$ th feature of the  $i$ th user, and  $\overline{f_j^i}$  is the mean of the  $j$ th

features of the  $i$ th user. The features with great ratio are the most discriminating. The features with lowest ratios are eliminated. The resulting pattern has 28 features.

Principal Component Analysis has also been considered to reduce the dimensionality of the features. In both cases we use a Linear Discriminant Analysis (LDA) to perform a best clustering of the features from the point of view of classification rather than feature reduction. Applying LDA to the features patterns, the length of the pattern is reduced to 19 new features, using a variation percentage of 94.5 % (34 features vector) and 96.5 % (28 features vector, considering the discriminability rate).

## 4 Neural Networks-Based Classifiers

The feature pattern obtained has to be compared with the patterns stored in the data base to determine if the user is the person who says to be. The classifiers used to verify the identity of the user are 3 neural networks: Two Self- Organizing Maps (supervised and unsupervised) and a Learning Vector Quantization, previously trained with patterns of those same users.

### 4.1 SOM

The SOM was developed by the Finn Kohonen in the early 1980s. The main reasons for which method has been chosen are the great potential of useful applications that it has and its simple model of natural neuronal networks. Problems that can be solved are: grouping, pattern classification, vector quantification, reduction of dimensions and extraction of characteristics.

The basic idea of SOM is simple: every neuron  $i$  of the map is associated with an  $n$  dimensional codebook vector  $m_i = (m_{i1}, \dots, m_{in})^T$ . The neurons of the map are connected to adjacent neurons by a neighbourhood relation, which defines the topology of the map.

The network is trained by finding the codebook vector which is most similar to an input vector. This codebook vector and its neighbours are then updated so as to render them more similar to the input vector.

This training can be made in an unsupervised or a supervised fashion. Supervision is achieved by attaching information about class membership to the input vector in the training phase, while during the recognition phase, the class label is omitted. Supervised training is made with the goal to improve the discrimination between pattern classes, but the results shows that this is not always true because for every input vector, the SOM is updated globally. Hence, the processing of an input belonging to a class adds noise to the mapping of another input from another class. The effect worsens the greater the number of different classes.

For the training, we used 1400 patterns, 5 photographs from each user, for 280 users. The number of map units (neurons) used in the networks is 4000. For training the maps in batch mode and for the weight initialization, we used the functions available in the SOMPACK [12].

## 4.2 LVQ

Learning Vector Quantization (LVQ) performs supervised training, which uses a set of inputs with their correctly annotated outputs adjusting the model when an error is committed between the model outputs and the known outputs.

The LVQ algorithm is a classification method based on neural competitive learning, which allows the definition of a group of categories on the input data space by reinforced learning, either positive (reward) or negative (punishment).

The training patterns are the same that SOM, the number of neurons is 2800 (10 neurons for each user), the typical class percentage is the same for each user: 1/280, and the learning rate is 0.01.

## 5 On Combining Classifiers

Once each one of the neural networks has made a decision, these individual opinions can be combined to obtain a consensus decision. Different classifiers can offer complementary information about the patterns, and combining this information, we can obtain best results.

One of the measurements we consider is the posterior probability for each classifier. With these probabilities we can use the sum (3), product (4) and maximum (5) rules, following the next equations:

$$\sum_{i=1}^R P(w_j | x_i) = \max_{k=1}^m \sum_{i=1}^R P(w_k | x_i) \quad (3)$$

$$\prod_{i=1}^R P(w_j | x_i) = \max_{k=1}^m \prod_{i=1}^R P(w_k | x_i) \quad (4)$$

$$P(w_j | x_i) = \max_{k=1}^m P(w_k | x_i) \quad (5)$$

where R is the number of classifiers, m the number of classes,  $x_i$  the measurement vector used by ith classifier, and  $P(w_k|x_i)$  the probability the pattern  $x_i$  belongs to class  $w_k$ .

## 6 Results

Two main analyses with different feature patterns have been made to obtain the results. The first pattern has 34 features, and after applying LDA the feature vector length is 19. The second pattern has 28 features, the most discriminating, and reduces its length to 19, too. The methods used are SOM unsupervised and supervised and LVQ and the results obtained can be seen in Table 1. The results are given in False Acceptance Rate (FAR) and False Rejection Ratio (FRR). The FAR numbers are obtained training the neural networks with all users excluding the person under recognition who is considered as an impostor to be rejected by the system.

**Table 1.** FAR and FRR for different feature vector lengths

|                 |                         | <b>FAR</b>      | <b>FRR</b>      |
|-----------------|-------------------------|-----------------|-----------------|
| <b>34 to 19</b> | <b>SOM unsupervised</b> | <b>0.3584 %</b> | <b>0.9615 %</b> |
|                 | <b>SOM supervised</b>   | <b>0.3571 %</b> | <b>1.0355 %</b> |
|                 | <b>LVQ</b>              | <b>0.3584 %</b> | <b>2.1450 %</b> |
| <b>28 to 19</b> | <b>SOM unsupervised</b> | <b>0.3579 %</b> | <b>0.4438 %</b> |
|                 | <b>SOM supervised</b>   | <b>0.3582 %</b> | <b>0.9615 %</b> |
|                 | <b>LVQ</b>              | <b>0.3584 %</b> | <b>0.5917 %</b> |

Table 2 shows the results for combination rules: sum rule, product rule and maximum rule, for each one of the previous pattern. It can be seen that in both cases the best results are obtained for the pattern of the 28 features. In the first table, the best method is the SOM unsupervised for both pattern; and in the second table, the best results are obtained for the product rule.

**Table 2.** FAR and FRR for different combination rules

|                 |                     | <b>FAR</b>      | <b>FRR</b>      |
|-----------------|---------------------|-----------------|-----------------|
| <b>34 to 19</b> | <b>Sum Rule</b>     | <b>0.3584 %</b> | <b>0.5178 %</b> |
|                 | <b>Product Rule</b> | <b>0.3578 %</b> | <b>0.1479 %</b> |
|                 | <b>Max Rule</b>     | <b>0.3576 %</b> | <b>0.4438 %</b> |
| <b>28 to 19</b> | <b>Sum Rule</b>     | <b>0.3579 %</b> | <b>0.3698 %</b> |
|                 | <b>Product Rule</b> | <b>0.3566 %</b> | <b>0.1479 %</b> |
|                 | <b>Max Rule</b>     | <b>0.3576 %</b> | <b>0.4438 %</b> |

If the user makes several repetitions (user puts the hand again if the system does not recognize him), it can be seen that the FRR decreases (Table 3). These results are only for the product rule with the pattern of 28 features (Th1). For FAR near 0, the results for the FRR are shown in the second row of the table 3 (for a new threshold Th2).

**Table 3.** FAR and FRR for several repetitions

| <b>Threshold</b> | <b>FAR</b>      | <b>FRR</b>      | <b>FRR<br/>1 repetition</b> | <b>FRR<br/>2 repetitions</b> |
|------------------|-----------------|-----------------|-----------------------------|------------------------------|
| <b>Th1</b>       | <b>0.3566 %</b> | <b>0.1479 %</b> | <b>0.0 %</b>                | <b>0 %</b>                   |
| <b>Th2</b>       | <b>0.0 %</b>    | <b>22.855 %</b> | <b>10.2071 %</b>            | <b>6.213 %</b>               |

## 7 Conclusions

We have developed a hand geometry recognition system based on the combination of three neural networks used for biometric verification and identification of individuals. The method here proposed used images obtained from a document scanner or a digital



camera, so users were free to put their hands in an arbitrary fashion. The only restriction was to keep their fingers separated. We have tested the system on a public database of 2800 hand images from 280 people.

As mentioned in the introduction, several hand geometry verification systems have been published in literature. The comparative among previous experimental results and the ones given by the proposed method is as follows:

- Jain et. al. [4] do verification using 500 images, 50 users. FAR=2%, FRR=15%
- Jain and Duta [7] do verification using 353 images, 53 persons. FAR=2%, FRR=3.5%
- Sanchez-Reillo et. al. [2] do verification and classification using 200 images from 20 people. Error rates: Classification 3%, Verification 10%.
- Oden et. al. [8] do verification and classification taking 30 images from 28 people. 95% success in Identification and 99% success in verification.
- Chikkerur et. al. [9] do verification using 500 images from 50 users. FAR=0.07%, FRR=0.80%
- Wong et. al. [10] do verification and classification using a dataset of 323 images from 29 users. FAR=2.2%, FRR=11%
- Bulatov et. al. [6] do verification and classification on a database of 714 images from 70 people. For FAR less than 1% they achieve FRR=3% for verification and FRR=6% for identification.
- Kumar et. al. [3] do verification on a database of 1000 hand images from 100 users. For 472 test images the best score is given by the fusion at decision of palmprint and hand geometry: FAR=0%, FRR=1.41%
- Our work does verification using 2800 images (1400 for training and 1400 for test), 280 persons. FAR=0.3566%, FRR=0.1479%

From the results given by other authors, we can state that the best score is given by the system described in this paper. The low scores obtained for FAR and FRR in addition to the simplicity of the acquisition, pegless, and the large database used, 280 individuals, confirm that hand geometry recognition is a suitable modality for medium and low-level security applications.

This hand geometry recognition system can be further combined with other biometric techniques. Perhaps the most immediate, just considering a digital camera or commercial flatbed scanner, is the combination of palmprint features, composed of principal lines, wrinkles, minutiae, etc, and hand geometry features, as done by Kumar et. al. [3]. Our current work is focused on the combination of different biometric techniques in a multimodal biometric system to cope with several of the problems present in unimodal systems, i.e., to improve matching performance, increase population coverage, and facilitate indexing.

## Acknowledgements

This work is partially supported by grants TIC2003-08382-C05-05 from the Spanish Ministry of Science and Technology (MCyT) and FEDER.

## References

1. Anail K. Jain, Arun Ross, Salil Prabhakar, An Introduction to Biometric Recognition. *IEEE Trans. on Circuits and Systems for Video Technology*. VOL. 14, NO 1 (2004) 4–20.
2. Raul Sanchez-Reillo, Carmen Sanchez-Avila, Ana Gonzalez-Marcos, Biometric Identification through Hand Geometry Measurements. *IEEE Trans. on Pattern Analysis and Machine Recognition*. VOL. 22, NO 10 (2000) 1169–1171.
3. A. Kumar, D.C.M. Wong, H.C. Shen, A.K. Jain, Personal Verification using Palmprint and Hand Geometry Biometrics. *Proceedings of the fourth International Conference on audio- and video-based biometric person authentication*, (2003).
4. A.K. Jain, A. Ross and S. Pankanti, A Prototype Hand Geometry-based Verification System, *Proc. of the 2nd International Conference on Audio- and Video-based Biometric Person Authentication*, (1999) 166-171.
5. [http://visgraph.cs.ust.hk/biometrics/Visgraph\\_web/index.html](http://visgraph.cs.ust.hk/biometrics/Visgraph_web/index.html)
6. Y. Bulatov, S. Jambawalikar, P. Kumar, and S. Sethia. Hand Recognition Using Geometric Classifiers. In *proc. of the 1st International Conference on Biometric Authentication (ICBA)*, (2004) 753-759.
7. Anail K. Jain, Nicole Duta. Deformable matching of hand shapes for verification. In *Proceedings of International Conference on Image Processing*, (1999).
8. Cenker Oden, Aytul Ercil, Hilkmert Kirmizita, and Burak Buke. Hand recognition using implicit polynomials and geometric feature. In *proceedings of AVBPA*, (2001) 336-341.
9. [http://www.eng.buffalo.edu/~ssc5/research/papers/biometric\\_hardening.pdf](http://www.eng.buffalo.edu/~ssc5/research/papers/biometric_hardening.pdf).
10. Alexandra L.N. Wong and Pengcheng Shi. Peg-Free Hand geometry Recognition Using Hierarchical Geometry and Shape Matching. *IAPR Workshop on Machine Vision Applications*, (2002) 281-284.
11. N. Otsu. A threshold selection method from grey-scale histogram. *IEEE Trans. Syst., Man, Cybern.*, vol. 8 (1978) 62-66
12. <http://www.cis.hut.fi/projects/somtoolbox/>

# Biometric Identification by Means of Hand Geometry and a Neural Net Classifier

Marcos Faundez-Zanuy and Guillermo Mar Navarro Mérida

Escola Universitària Politècnica de Mataró  
Universitat Politècnica de Catalunya, Barcelona, Spain  
faundez@eupmt.es  
<http://www.eupmt.es/veu>

**Abstract.** This Paper describes a hand geometry biometric identification system. We have acquired a database of 22 people using a conventional document scanner. The experimental section consists of a study about the discrimination capability of different extracted features, and the identification rate using different classifiers based on neural networks.

## 1 Introduction

In recent years, hand geometry has become a very popular access control biometrics which has captured almost a quarter of the physical access control market [1]. Even if the fingerprint is most popular access system [2-4], the study of other biometric systems is interesting, because the vulnerability of a biometric system [5] can be improved using some kind of data fusion [6] between different biometric traits. This is a key point in order to popularize biometric systems [7], in addition to privacy issues [8].

Although some commercial systems, such as the system shown in figure 1 rely on a three-dimensional profile of the hand, in this paper we study a system based on two dimensional profiles. Although three dimensional devices provide more information than two dimensional ones, they require a more expensive and voluminous hardware.



**Fig. 1.** Commercial three-dimensional scanner

A two-dimensional profile of a hand can be get using a simple document scanner, which can be purchased for less than 100 USD. Another possibility is the use of a digital camera, whose cost is being dramatically reduced in the last years.

In our system, we have decided to use a conventional scanner instead of a digital photo camera, because it is easier to operate, and cheaper. On the other hand, although a digital camera is extremely fast in taking a photo, the last generation scanners (such as EPSON 4870 Photo perfection) are able to capture a DIN A4 size colour document (24 bit) at a 150 dpi resolution in less than 15 seconds when using the USB 2 port, which is a quite reasonable time.

This paper can be summarized in three main parts: section two describes a database which has been specially acquired for this work. In section three, we describe the pre-processing and we study the discrimination capability of several measurements on the sensed data. Section four provides experimental results on identification rates using neural net classifiers.

## 2 Database

We have acquired a database of 22 people, and 10 different acquisitions per person. If some acquisition has not let to extract some of the parameters described in the next section, this capture has been rejected and replaced by a new one. Figure 2 shows an example of defective acquisitions and the reason.

The database has been stored in bmp format using 8 bits per pixel (256 gray levels), a resolution of 100 dpi, and an image size of 216x240 mm. Higher resolutions would imply more details but also more computational time in order to process a hand image. In our preliminary experiments we have found that 100 dpi offers a good compromise. Obviously this resolution is insufficient for other related applications such as palm print, which is analogous to fingerprint recognition, but using the ridge and valley pattern of the hand skin. Thus, the system will rely on the silhouette of the hand and will ignore other details such as fingerprints, lines, scars and color. For this reason, the first step of the pre-processing described in the next section will be a binarization and a contour extraction. Although this procedure discards useful information for discrimination, it also alleviates other problems, such as the perspiration of the skin which blots the thin details of the image. Figure 3 shows an example of this phenomenon.



**Fig. 2.** Example of defective acquisitions. The first one is defective because it is cut on the base. In the second one, some fingers are joined. In the third one, one finger is cut

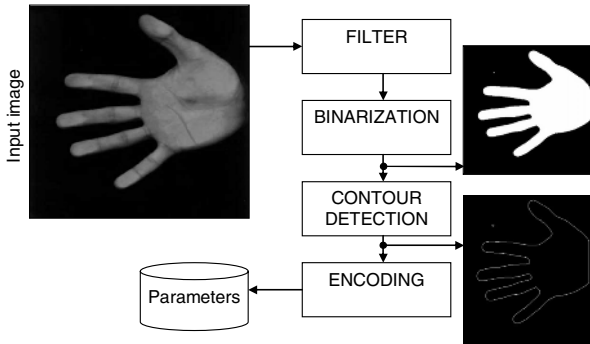


**Fig. 3.** Example of hand acquisition at 150 dpi and 24 bit per pixel (color image), with perspiration problem. This problem can be neglected after the binarization step

### 3 Feature Extraction

#### 3.1 Pre-processing Algorithm

Figure 4 shows a block diagram of the pre-processing algorithm.



**Fig. 4.** Block diagram for the proposed pre-processing scheme

The description of each block is the following:

**Filter**

We apply a low-pass filtering in order to remove spurious noise.

**Binarization**

The goal is the conversion from an image  $I(x, y)$  at 8 bit per pixel to a monochrome image  $I'(x, y)$  (1 bit per pixel. “0”=black, “1”=white), applying a threshold:

$$I'(x, y) = \begin{cases} 1 & \text{if } I(x, y) \geq \text{threshold} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

We use threshold=0.07.

**Contour Detection**

The goal is to find the limits between the hand and the background. For this purpose the algorithm detects the intensity changes, and marks a closed set of one pixel wide and length the perimeter of the hand. Edge points can be thought of as pixel locations of abrupt grey-level change. For example it can be defined an edge point in binary images as black pixels with at least one white nearest neighbour. We use the Laplacian of Gaussian method, which finds edges by looking for zero crossings after filtering the image with a Laplacian of Gaussian filter.

**Coding**

This step reduces the amount of information. We translate a bmp file to a text file that contains the contour description. The encoding algorithm consists of a chain code. In chain coding the direction vectors between successive boundary pixels are encoded. Figure 5 shows our code, which employs 8 possible directions and can be coded by 3-bit code words.

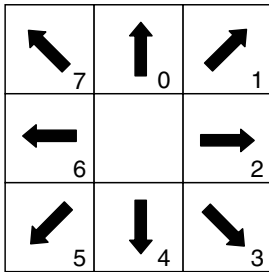


Fig. 5. Contour coding algorithm

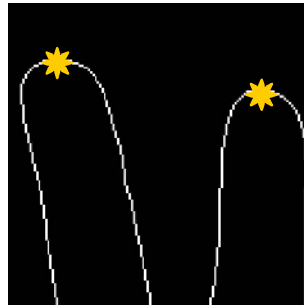


Fig. 6. Maximum of first and middle fingers

Once upon the chain code is obtained, the perimeter can be easily computed: for each segment, an even code implies +1 and an odd code  $+\sqrt{2}$  units. the beginnings and ends of the fingers and wrist are found looking for minimum and maximum values in the chain code.

The finger limits (base and maximum height) are detected in the middle of a region with a “5” and “3”-code. Figure 6 shows, for example, the maximum of the first and middle fingers.

**3.2 Proposed Features**

Using the result of the previous section as input, we propose the following measurements (see figure 7):

1. Thumb finger length.
2. First finger length.
3. Middle finger length.
4. Ring finger length.
5. Little finger length.

6. Wrist length.
7. Thumb base width.
8. First finger width.
9. Middle finger width.
10. Ring finger width.
11. Little finger width.
12. Hand perimeter.
13. Hand surface.

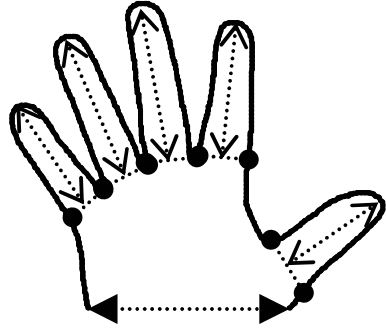


Fig. 7. Measured features

Some of these features have been removed in the experiments due to their low discrimination capability. Our experiments have revealed that results are improved deleting features 1, 6, 7 and 13. Thus, we will select the remaining nine features per image.

## 4 Experimental Results and Conclusions

### 4.1 Conditions of the Experiments

Our results have been obtained with the database described in section 2, the preprocessing of section 3, and the selected parameters of section 4, in the following situation: 22 persons, images 1 to 5 for training, and images 6 to 10 for testing.

### 4.2 Nearest Neighbour Classifier

We obtain one model from each training image. During testing each input image is compared against all the models inside the database ( $22 \times 5 = 110$  in our case) and the model close to the input image (using Mean Square Error criterion) indicates the recognized person.

In our experiments, we are making for each user, all other users' samples as impostor test samples, so we finally have,  $N = 22 \times 5$  (client) +  $22 \times 21 \times 5$  (impostors) = 2420 different tests. We have used two different distance measures:

$$MSE(\bar{x}, \bar{y}) = \sum_{i=1}^P (x_i - y_i)^2 \quad (2)$$

$$MAD(\bar{x}, \bar{y}) = \sum_{i=1}^P |x_i - y_i| \quad (3)$$

Where  $P$  is the vector dimension.

### 4.3 Multi-layer Perceptron Classifier Trained in a Discriminative Mode

We have trained a Multi-Layer Perceptron (MLP) [9] as discriminative classifier in the following fashion: when the input data belongs to a genuine person, the output

(target of the NNET) is fixed to 1. When the input is an impostor person, the output is fixed to -1. Figure 8 shows the neural network architecture. We have used a MLP with 30 neurons in the hidden layer, trained with the Levenberg-Marquardt algorithm, which computes the approximate Hessian matrix, because it is faster and achieves better results than the classical back-propagation algorithm. We have trained the neural network for 10 epochs (50 epochs when using regularization). We also apply a multi-start algorithm and select the best result.

The input signal has been fitted to a [-1, 1] range in each component.

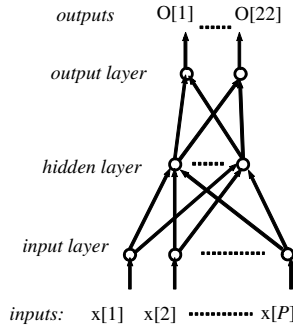


Fig. 8. Multi-layer Perceptron architecture

One of the problems that occur during neural network training is called *overfitting*. The error on the training set is driven to a very small value, but when new data is presented to the network the error is large. The network has memorized the training examples, but it has not learned to generalize to new situations. The adopted solution to the overfitting problem has been the use of regularization. The regularization involves modifying the performance function, which is normally chosen to be the sum of squares of the network errors on the training set. So, this technique helps take the mystery out of how to pick the number of neurons in a network and consistently leads to good networks that are not overtrained. The classical Mean Square Error (MSE) implies the computation of (4):

$$MSE = \frac{1}{N} \sum_{i=1}^P (t_i - a_i)^2 \tag{4}$$

Where  $t, a$  are the  $P$  dimensional vectors of the test input and the model, respectively. The regularization uses the following measure (5):

$$MSEREG = \gamma MSE + (1 - \gamma) \frac{1}{n} \sum_{j=1}^n w_j^2 \tag{5}$$

Thus, it includes one term proportional to the modulus of the weights of the neural net.



In addition, there is another important topic: the random initialization. We have studied two strategies:

- a) To pick up the best random initialization (the initialization which gives the higher identification rate)
- b) A committee of neural networks, which combines the outputs of several MLP, each one trained with a different initialization.

#### 4.4 Radial Basis Function Classifier Trained in a Discriminative Mode

We have trained a Radial Basis Function (RBF) in a similar fashion than MLP of previous section. Figure 9 shows the architecture. Taking into account that a RBF is faster to train, we have worked out a exhaustive study varying the number of centres. Figure 10 shows the identification rate as function of the number of centres. It can be seen that the maximum value is 89.09%, which is achieved using 50 centres.

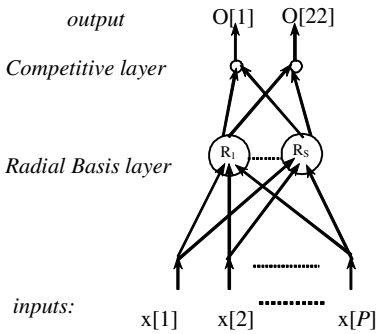


Fig. 9. Radial Basis Function architecture

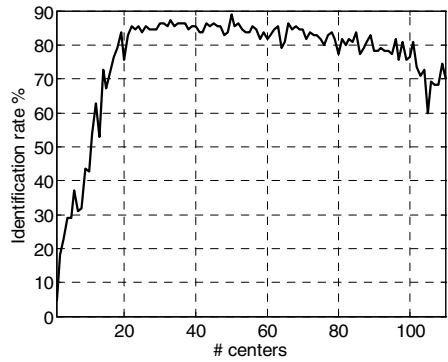


Fig. 10. Identification rate vs. number of centers for the RBF network

#### 4.5 Experimental Results

Table 1 compares the neural net results with the classical nearest neighbour classifier with two different distance measures. It can be appreciated that the neural networks outperform the Nearest Neighbour classifier.

Table 1. Comparison between different Classifiers

| Classifier                                | Identification rate (%) |
|-------------------------------------------|-------------------------|
| Nearest Neighbor (MAD)                    | 64,55%                  |
| Nearest Neighbor (MSE)                    | 73,64%                  |
| Multi-Layer Perceptron (MSE, 10 epoch)    | 91,82%                  |
| Multi-Layer Perceptron (MSEREG, 50 epoch) | 92,73%                  |
| MLP committee 3 nets (MSE, 10 epoch)      | 93,64%                  |
| MLP committee 3 nets (MSEREG, 50 epoch)   | 93,64%                  |
| Radial Basis Function                     | 90%                     |

Although hand-geometry does not offer the good results of fingerprint biometric recognition [10-11], it can be more accepted by the uses, because fingerprint are more related to police, and criminal records.

## Acknowledgement

This work has been supported by FEDER and the Spanish grant MCYT TIC2003-08382-C05-02.

## References

1. Jain A. K., Bolle R., Pankanti S., "Introduction to biometrics" in Biometrics Personal identification in networked society. Kluwer Academic Publishers 1999
2. Faundez-Zanuy, M., "Door-opening system using a low-cost fingerprint scanner and a PC" IEEE Aerospace and Electronic Systems Magazine. Vol. 19 n° 8, pp.23-26, August 2004.
3. Faundez-Zanuy M., Fabregas J. "Testing report of a fingerprint-based door-opening system". In press, IEEE Aerospace and Electronic Systems Magazine. 2005
4. Espinosa-Duró V. , "Fingerprints thinning algorithm" IEEE Aerospace and Electronic Systems Magazine, pp., pp. 28-30. Vol. 18 , Issue: 9. September 2003
5. Faundez-Zanuy, M., "On the vulnerability of biometric security systems". IEEE Aerospace and Electronic Systems Magazine. Vol.19 n° 6, pp.3-8, June 2004.
6. Faundez-Zanuy M., "Data fusion in biometrics" IEEE Aerospace and Electronic Systems Magazine. Vol.20 n° 1, pp.34-38, January 2005.
7. Faundez-Zanuy, M., "Biometric recognition: why not massively adopted yet?". In press, IEEE Aerospace and Electronic Systems Magazine. 2005
8. Faundez-Zanuy, M., "Privacy issues on biometric systems". IEEE Aerospace and Electronic Systems Magazine. Vol.20 n° 2, pp13-15. February 2005
9. Haykin S., "Neural nets. A comprehensive foundation", 2on edition. Ed. Prentice Hall 1999
10. Faundez-Zanuy, M., "Are inkless fingerprint sensors suitable for mobile use?" IEEE Aerospace and Electronic Systems Magazine. Vol.19 n° 4, pp.17-21, April 2004.
11. Faúndez-Zanuy, M. "Technological evaluation of two AFIS systems". In press, IEEE Aerospace and Electronic Systems Magazine. 2005

# Study of a Committee of Neural Networks for Biometric Hand-Geometry Recognition

Marcos Faundez-Zanuy

Escola Universitària Politècnica de Mataró,  
Universitat Politècnica de Catalunya, Barcelona, Spain  
faundez@eupmt.es  
<http://www.eupmt.es/veu>

**Abstract.** This Paper studies different committees of neural networks for biometric pattern recognition. We use the neural nets as classifiers for identification and verification purposes. We show that a committee of nets can improve the recognition rates when compared with a multi-start initialization algorithm that just picks up the neural net which offers the best performance. On the other hand, we found that there is no strong correlation between identification and verification applications using the same classifier.

## 1 Introduction

Neural networks are one of the most powerful tools for pattern recognition [1]. On the other hand, one of the applications of pattern recognition, person identification by means of biometrics [2], has gain market share. Nevertheless, still remain some unsolved problems [3-5].

In this paper, we focus on a biometric identification system improvement by means of a committee of neural networks. Figure 1 shows the general scheme of a biometric system.

These systems can be operated in two ways:

- a) Identification: In this approach no identity is claimed from the person. The automatic system must determine who is trying to access.
- b) Verification: In this approach the goal of the system is to determine whether the person is who he/she claims to be. This implies that the user must provide an identity and the system just accepts or rejects the users according to a successful or unsuccessful verification. Sometimes this operation mode is named authentication or detection.

We will focus on a study of the third block of figure 1. Without loss of generality we will use a set of features and a database extracted from hand-geometry images obtained from a 22 people and 10 different realizations per person set (5 for training and 5 for testing). The feature extraction section and the digital signal input (blocks 1 and 2 of figure 1) can be found in [6]. Our matching algorithm will be a neural network or a committee of neural networks.

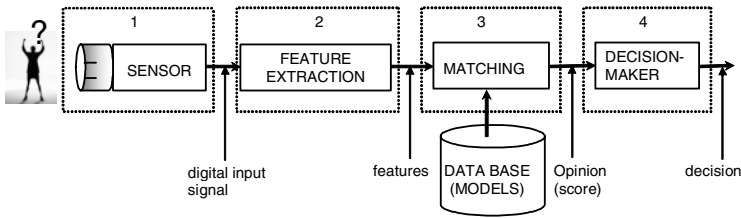


Fig. 1. General scheme of a biometric recognition system

1.1 Biometric Identification

The system performance can be evaluated using an identification rate. This is obtained by the ratio between the number of properly assigned identities and the number of trials of the experiment. For this purpose, having a labelled training set that contains examples of each person, we train a neural network in a discriminative mode: for each input pattern we force the neural net to learn +1 at the output of the corresponding user and -1 at the other outputs.

$$O[i] = \begin{cases} 1 & \text{if } \vec{x} = (x[1], x[2], \dots, x[P]) \in \text{person } \#i \\ -1 & \text{otherwise} \end{cases} \tag{1}$$

Figure 2 shows the neural network scheme.

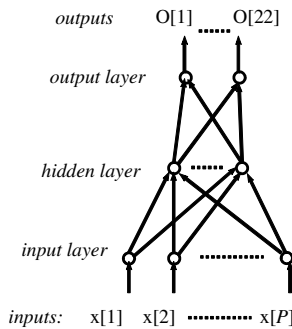


Fig. 2. Neural network architecture

For identification, if we have a population of  $N$  different people, and a labelled test set, we fill a matrix  $S$ , where the elements are interpreted in the following way:

$$s_{ijk} = O[j] \Big|_{\vec{x} \in \text{person}\#i}, k = 1, \dots, \# \text{trials} \tag{2}$$

Where trials is the number of different testing images per person ( $k=5$  in our experiments), and  $s_{ijk}$  is the similarity from the  $k$  realization of an input signal belonging to person  $i$ , to the model of person  $j$ .

This matrix can be drawn as a three dimensional data structure (see figure 3).

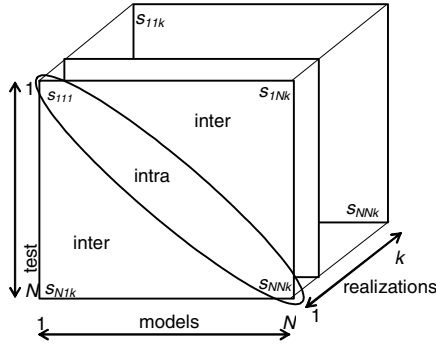


Fig. 3. Proposed data structure

Thus, the identification rate looks for each realization, in each row, if the maximum similarity is inside the principal diagonal (success) or not (error), and works out the identification rate as the ration between successes and number of trials (successes + errors):

```

for i=1:N,
 for k=1:#trials,
 if (s_{iik} > s_{ijk}) $\forall j \neq i$, then success=success+1
 else error=error+1
 end
 end
end
end

```

### 1.2 Biometric Verification

Verification systems can be evaluated using the False Acceptance Rate (FAR, those situations where an impostor is accepted) and the False Rejection Rate (FRR, those situations where a user is incorrectly rejected), also known in detection theory as False Alarm and Miss, respectively. This framework gives us the possibility of distinguishing between the discriminability of the system and the decision bias. The discriminability is inherent to the classification system used and the discrimination bias is related to the preferences/necessities of the user in relation to the relative importance of each of the two possible mistakes (misses vs. false alarms) that can be done in verification. This trade-off between both errors has to be usually established by adjusting a decision threshold. The performance can be plotted in a ROC (Receiver Operator Characteristic) or in a DET (Detection error trade-off) plot [7]. DET curve gives uniform treatment to both types of error, and uses a scale for both axes, which spreads out the plot and better distinguishes different well performing systems and usually produces plots that are close to linear. DET plot uses a logarithmic scale that expands the extreme parts of the curve, which are the parts that give the most information about the system performance. For this reason the speech community prefers DET instead of ROC plots. Figure 4 shows an example of DET plot.

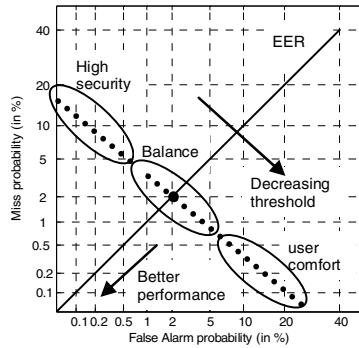


Fig. 4. Example of a DET plot for a user verification system (dotted line)

We have used the minimum value of the Detection Cost Function (DCF) for comparison purposes. This parameter is defined as [7]:

$$DCF = C_{miss} \times P_{miss} \times P_{true} + C_{fa} \times P_{fa} \times P_{false} \tag{3}$$

Where  $C_{miss}$  is the cost of a miss (rejection),  $C_{fa}$  is the cost of a false alarm (acceptance),  $P_{true}$  is the a priori probability of the target, and  $P_{false} = 1 - P_{true}$ .  $C_{miss} = C_{fa} = 1$ .

Using the data structure defined in figure 3, we can easily apply the DET curve analysis. We just need to split the distances into two sets: intra-distances (those inside the principal diagonal), and inter-distances (those outside the principal diagonal).

## 2 Experimental Results and Conclusions

We have trained a neural network on the conditions of previous section, and evaluated the identification rates and minimum value of the DCF. The experiments have been done with the following parameters:

- a) Multi-Layer Perceptron 9×30×22 trained with the Levenberg-Marquardt algorithm during 10 epochs.
- b) Multi-Layer Perceptron 9×30×22 trained with the Levenberg-Marquardt algorithm during 50 epochs using regularization [8].
- c) A committee of three MLP, each one trained on the conditions of section a)
- d) A committee of three MLP, each one trained on the condition of section b).

### 2.1 Committee of Neural Networks

In pattern recognition applications it is well known that a number of differently trained neural networks (that can be considered as “experts”), which share a common input, can produce a better result if their outputs are combined to produce an overall output. This technique is known as ensemble averaging, committee machine [9], data fusion [10], etc. The motivation for its use is twofold [9]:

- If the combination of experts were replaced by a single neural network, the number of equivalent adjustable parameters would be large, and this implies more training time and local minima problems [11].
- The risks of over-fitting the data increases when the number of adjustable parameters is large compared to the size of the training data set.

Figure 5 shows the equivalent scheme for an  $M$  experts' combination. We have considered each classifier as an expert with scalar output  $y_i[n]$  and vectorial input  $\vec{x}$ :

$$y_i[n] = F_i(\vec{x}) = F_i(x[n-1], x[n-2], \dots, x[n-P]) \tag{4}$$

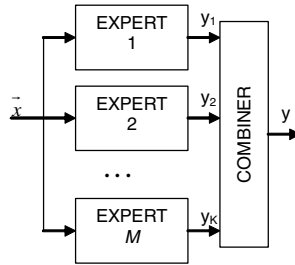


Fig. 5. Committee machine based on ensemble-averaging

The expectation is that differently trained experts converge to different local minima on the error surface, and overall performance is improved by combining the outputs in some way. Let  $\vec{x}$  denote an input vector and  $y_d$  the desired response (predicted output value) for this input value. If the output of the neural net number  $i$  is  $F_i(\vec{x})$ , we can compute the mean squared error (MSE) of the obtained output as:

$$MSE[F_i(\vec{x})] = E\left[\left(y_d - F_i(\vec{x})\right)^2\right] \tag{5}$$

We can average the performances for a set of different neural networks (or the same architecture trained with the same algorithm but using a different random initialization) to obtain the average mean square error:

$$E\left[MSE\left[F_i(\vec{x})\right]\right] = \frac{1}{N} \sum_{i=1}^M E\left[\left(y_d - F_i(\vec{x})\right)^2\right] \tag{6}$$

We can define the Basic Ensemble Method (BEM) regression function [12] as:

$$F_{BEM}(\vec{x}) \equiv \frac{1}{M} \sum_{i=1}^M F_i(\vec{x}) = y_d - \frac{1}{M} \sum_{i=1}^M \left(y_d - F_i(\vec{x})\right) \tag{7}$$

If we assume that the errors  $m_i = y_d - F_i(\vec{x})$ ,  $i=1, \dots, M$  of each individual predictor are mutually independent with zero mean, we can calculate the MSE of  $F_{BEM}(\vec{x})$ :

$$MSE \left[ F_{BEM}(\bar{x}) \right] = E \left[ \left( \frac{1}{M} \sum_{i=1}^M m_i \right)^2 \right] = \frac{1}{M^2} E \left[ \sum_{i=1}^M m_i^2 \right] + \frac{1}{M^2} E \left[ \sum_{i \neq j}^M m_i m_j \right] = \tag{8}$$

$$\frac{1}{M^2} E \left[ \sum_{i=1}^M m_i^2 \right] + \frac{1}{M^2} \sum_{i \neq j}^M E[m_i] E[m_j] = \frac{1}{M^2} E \left[ \sum_{i=1}^M m_i^2 \right] \Rightarrow MSE[F_{BEM}] = \frac{1}{M} E[MSE] \tag{9}$$

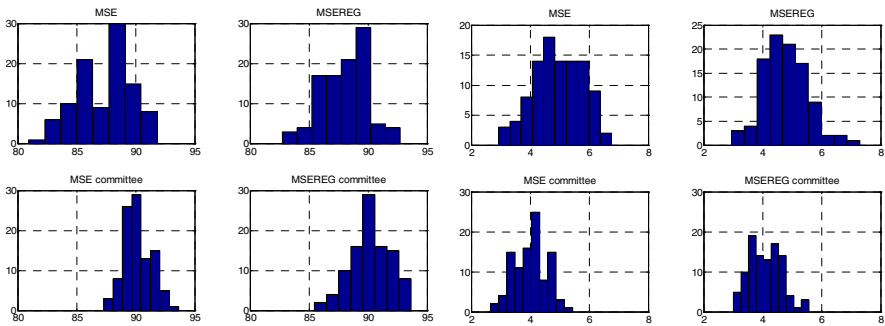
This result shows that, by averaging regression estimates, we can reduce the MSE by a factor of  $M$  when compared to the individual population performance. By increasing the population size, we can, in principle, make the estimation error arbitrarily small, but if  $M \rightarrow \infty$  the assumption that the errors are mutually independent is not valid. In practice there is saturation on performance, and after a given value of  $M$  there is no improvement with an increase on the number of combined neural net classifiers. In our experiments, we will use  $M=3$ .

### 2.2 Results and Conclusions

We have done 100 random initializations for each one of the four schemes defined in the introductory part of section 2. Figure 6 shows the histograms for Identification rate and minimum value of DCF (left and right respectively). Figure 7 shows a fitted Gaussian to the previous histograms (fig. 6). It is clear that the committee of experts improves the mean value and decreases the variance. Thus, with a committee of experts a better recognition result is achieved.

Another interesting study is the correlation between identification and verification rates. That is: if we get a good system for identification, does it mean that we have a good verifier? Table 1 shows the identification and verification results for each studied scenario along with the correlation factor between both parameters.

Although these results are not as good as other biometric traits such as fingerprints [13-16], for our purpose we prefer this set of experimental data, where the classifier produces more errors. Figure 8 shows a two-dimensional plot, where for each classifier, the identification versus the verification results are plotted. Surprisingly, we



**Fig. 6.** Histograms of identification rates (on the left) and DCF (on the right) for different classification strategies and 100 random initializations



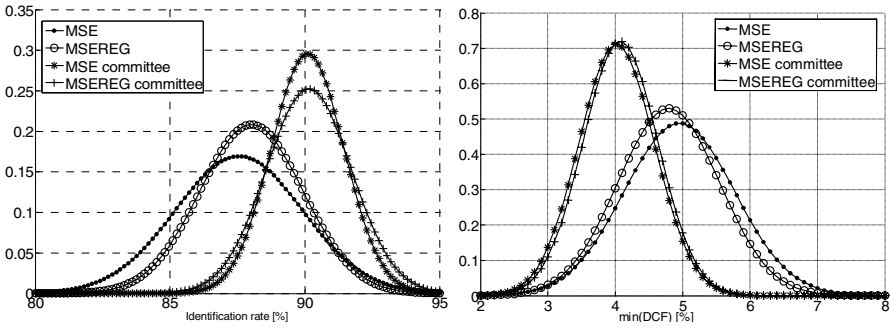


Fig. 7. A fitted Gaussian to each histogram of figure 6

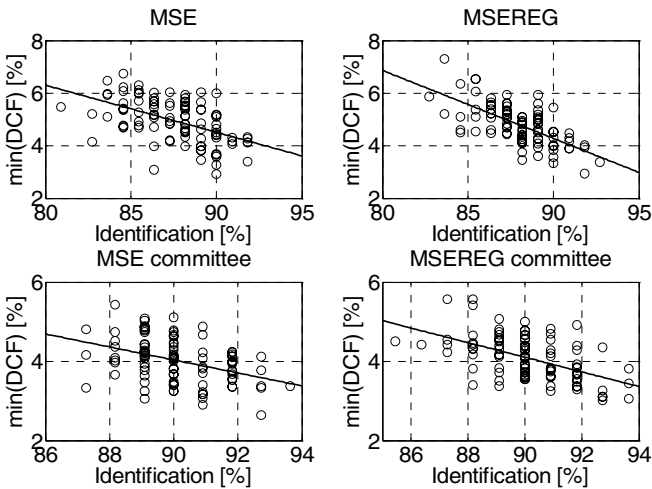


Fig. 8. 2-D plot of identification versus DCF pair values for each trained classifier

Table 1. Comparison between different schemes performance

| Classifier             | Identification rate (%) | Min(DCF) | Corr(Ident,DCF) |
|------------------------|-------------------------|----------|-----------------|
| MLP (MSE)              | 91,82%                  | 2.92%    | -0.52           |
| MLP (MSEREG)           | 92,73%                  | 2.94%    | -0.66           |
| MLP (MSE) committee    | 93,64%                  | 2.64%    | -0.39           |
| MLP (MSEREG) committee | 93,64%                  | 3.01%    | -0.53           |

observe that the best classifier for identification does not necessarily imply the best performance for verification. We have also interpolated a first-order polynomial in order to see the tendency. Fortunately, the best classifier for identification is, the

better the tendency for verification. However, there are several classifiers that achieve the best identification rate, and the equivalent DCF can differ significantly. We think that an interesting research line should solve the question: what needs a good identifier in order to also be a good verifier? This question is not easy to solve. However, we have observed that regularization provides higher correlation (absolute value of the correlation coefficient) between a good classifier and a good verifier than its counterpart without regularization.

## Acknowledgement

This work has been supported by FEDER and the Spanish grant MCYT TIC2003-08382-C05-02.

## References

1. Bishop, C.M. "Neural networks for pattern recognition". Oxford University Press, 1995.
2. Jain A. K., Bolle R., Pankanti S., "Introduction to biometrics" in Biometrics Personal identification in networked society. Kluwer Academic Publishers 1999.
3. Faundez-Zanuy, M., "On the vulnerability of biometric security systems". IEEE Aerospace and Electronic Systems Magazine. Vol.19 n° 6, pp.3-8, June 2004.
4. Faundez-Zanuy, M., "Biometric recognition: why not massively adopted yet?". In press, IEEE Aerospace and Electronic Systems Magazine. 2005.
5. Faundez-Zanuy, M., "Privacy issues on biometric systems". IEEE Aerospace and Electronic Systems Magazine. Vol.20 n° 2, pp13-15. February 2005
6. Faundez-Zanuy, M., Navarro-Mérida, G. M., "Biometric identification by means of hand geometry and a neural net classifier". Lecture Notes in Computer Science IWANN'2005.
7. Martin A. et al. "The DET curve in assessment of detection performance", V. 4, pp.1895-1898, European speech Processing Conference Eurospeech 1997
8. Foresee, F. D., and M. T. Hagan, "Gauss-Newton approximation to Bayesian regularization," Proceedings of the 1997 International Joint Conference on Neural Networks, pp. 1930-1935, 1997.
9. Haykin S., "Neural nets. A comprehensive foundation", 2nd ed. Ed. Prentice Hall 1999
10. Faundez-Zanuy M., "Data fusion in biometrics" IEEE Aerospace and Electronic Systems Magazine. Vol.20 n° 1, pp.34-38. January 2005.
11. Jain A. K., Mao J. "Artificial neural networks: a tutorial". IEEE Computer pp.31-44, March 1996
12. Perrone, M. P., Cooper L. N. "When networks disagree: ensemble methods for hybrid neural networks" in "neural networks for speech and image processing, R. J. Mammone ed., Chapman-Hall 1993
13. Faundez-Zanuy, M., "Door-opening system using a low-cost fingerprint scanner and a PC" IEEE Aerosp. and Electr. Systems Magazine. Vol. 19 n° 8, pp.23-26, August 2004.
14. Faundez-Zanuy M., Fabregas J. "Testing report of a fingerprint-based door-opening system". In press IEEE Aerospace and Electronic Systems Magazine. 2005.
15. Faundez-Zanuy, M., "Are inkless fingerprint sensors suitable for mobile use?" IEEE Aerospace and Electronic Systems Magazine. Vol.19 n° 4, pp.17-21, April 2004.
16. Faundez-Zanuy, M. "Technological evaluation of two AFIS systems". In press, IEEE Aerospace and Electronic Systems Magazine. 2005.

# An Evolutionary Environment for Wind Turbine Blade Design

V. Díaz Casás<sup>1</sup>, F. Lopez Peña<sup>1</sup>, A. Lamas<sup>2</sup>, and R. J. Duro<sup>2</sup>

<sup>1</sup> Grupo de Ingeniería de Fluidos, Universidade da Coruña

<sup>2</sup> Grupo de Sistemas Autonomos, Universidade da Coruña

**Abstract.** The aerodynamic design of wind turbine blades is carried out by means of evolutionary techniques within an automatic design environment based on evolution. A simple, fast, and robust aerodynamic simulator is embedded in the design environment to predict the performance of any turbine produced as intermediate individual of the evolutionary process. The aerodynamic simulator is based on blade element theory in which a panel method is combined with an integral boundary layer code to calculate blade airfoils' characteristics. In order to reduce computations some simplifications were contemplated and the results corrected by means of the application of neural network based approximations. Results of the simulations obtained using this technique, of the application of the automatic design procedure and of the operation of the wind turbines thus obtained are presented.

## 1 Introduction

In the last decade or so, wind turbines have become a common sight in our landscape and an increasingly important source of clean energy. Economical performance of a wind turbine is determined by the efficiency of the blade. Thus, any improvement in the aerodynamic design of wind turbines implies a significant profit increase during the 20 years of its lifetime.

Analytical optimal solutions for each specific operational condition in terms of blade chord and angle distribution can be easily calculated as shown by Hansen [1]. However, wind turbines find themselves immersed in many different operational conditions throughout their lifetimes and designing for a particular one will not optimize the performance during its lifetime and thus its efficiency in energetic and economic terms. Thus, for a good design of a wind turbine, the target is to consider its operation in a range of real conditions and for this case there is no analytical solution. Consequently, until now, in most cases, what engineers have done to obtain these designs is to start with an analytical solution for a point and apply some empirically determined corrections. This approach is not fail proof and requires a lot of experience on the part of the designer. In addition it does not provide a simple way to develop particular blades for specific locations taking into account their wind patterns.

To provide one way of doing this, this paper proposes a different approach. We propose developing and using an evolutionary design environment where the design

can be optimized for the range of conditions the blade will find itself in when in real operation. This approximation to the design problem is based on work by other authors in the application of evolution for the design and optimization of different types of systems in other realms. Examples of applications in this line are described in the work by Kirsch and Rozvany for the structural optimization of bridges and transmission towers [2] or in the work by Keane et al. [3], Song et al. [4] or Ong et al. [5] in airplane and airfoil design optimization.

However, these optimization systems present two important disadvantages: first, they are exclusively created for the optimization of existing designs; and secondly, they can only optimize some type of designs. In this work we are creating the designs in a general purpose evolutionary design environment created in our group which through standardized interfaces provides a way to include the simulators or human evaluation systems necessary for its adaptation to different design problems. In addition, the system is not aimed at the optimization of a given design, but at the creation of possible design alternatives. This design system integrates all of the stages of the design process through the integration of advanced search and analysis strategies. Thus, the design process is approached as an optimization problem in a space of very high, and usually variable, dimensionality. This is the reason why the selection of the search paths must not be left only to the designer, as his/her limitations would severely constrain the search process giving rise to fictitious constraints, specially in the case of complex, non-linear or multidimensional structures that require the interaction of multiple knowledge sources such as the ones we want to consider here.

Thus, the design environment reduces the participation of the human to the formulation of the problem (specification stage) and the subjective evaluation stage, liberating him/her from the classical search for solutions and decision making.

## 2 Design Environment

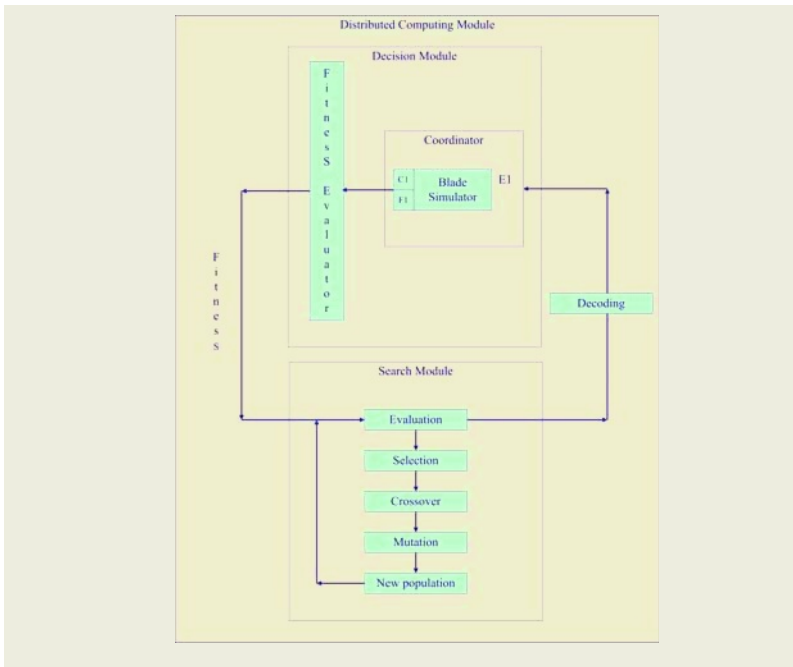
The Basic structure of the design system comprises three blocks:

- Solution search block.
- Decision making block (evaluation with or without human participation).
- Computational distribution block.

The search stage is based on different types of evolutionary algorithms which employ a decision block for the evaluation of candidate solutions. In the particular case of the design of blades for wind turbines, the algorithm that was chosen, as indicated in the experimental section, was a standard genetic algorithm. The reason for this choice was that the emphasis of the work was on evaluating the operation of the system and this could be done more easily through the standard GA. The GA uses a decision module based on a blade simulator that was developed ad hoc for the evaluation of each possible candidate and providing the fitness value needed for evolution to be carried out.

### 3 Implementation of the Decision Module

The decision module consists of a blade simulator which takes into account the aerodynamic processes governing the wind-blade interaction under different angle and wind speed conditions. The simulator should calculate the lift, drag and the rotation of the rotor generated under the conditions under study. Obviously, to obtain the best possible simulation, a realistic CFD code (computational fluid dynamics) should be used as simulator. The environment presented here provides the framework to introduce any simulator we wish, commercial or not. However, in order to reduce computation time so that the system could run within time spans compatible with industrial computing resources, a specific simplified blade simulator was developed.



**Fig. 1.** Structure of the Design Environment

The simulator is based on the well known blade element theory. This classic theory evaluates each blade as divided into small portions, called blade elements, along its span. The blade’s efficiency is obtained through the study of each element after considering how it is influenced by the other elements and by the wake produced by the rotating set of blades. Under these assumptions, each element can be analyzed as a two-dimensional airfoil in which the attack angle (the angle of the incoming wind) is modified by taking into account the flow velocity perturbations induced by the above mentioned elements. In this way, for each element, lift and drag coefficients can be easily computed, and from these the lift, and drag forces and the torque of the whole windmill rotor can be obtained.

The simulator is divided into two main parts: the first one computes the performance of two-dimensional airfoils and the other one applies the blade element theory in order to obtain the resulting forces and moments. The analysis of two-dimensional airfoils can be greatly simplified in cases where the main flow could be considered as ideal or potential, which is mostly true when analyzing windmills. The ideal flow condition occurs in all cases where the Reynolds number ( $Re$ ) achieves values much larger than one. The Reynolds number is a non-dimensional parameter obtained as the ratio between the inertial and the viscous forces acting within the fluid. However, in all cases certain regions exist within the flow-field, the so called boundary layers, where the viscous effects are dominant, thus the flow cannot be considered as ideal inside these regions. Therefore, in aerodynamic calculations it is customary to analyze the main flow by using the Euler equations governing the ideal flow motion and, after that, applying the boundary layer equations to obtain the viscous drag on the solid surfaces. When applying such an approximation, it should be taken into account that in cases where the pressure along the wall increases when moving downstream, the boundary layer can separate from the wall. This separation will induce a further increase of drag forces [7], [8].

With these considerations the process of calculation of the characteristics of the airfoil has been divided in three phases:

- Analysis of the ideal flow around the airfoil.
- Study of the bounded layer.
- Estimation, if needed, of forces values correction due to the detachment of the boundary layer.

Lift and drag forces acting on any airfoil are calculated by integrating the pressure acting on its surface. As the flow is considered ideal or potential, pressure forces are obtained by calculating the flow kinetic energy variation along the airfoil surface. The velocity field of an ideal flow is given as the gradient of a potential function. This potential function can be evaluated by calculating the circulation of the flow velocity around the airfoil.

In potential flow theory viscous forces are neglected, therefore an important component of drag forces –friction forces- is not considered. This situation makes it necessary to add the effects of viscous processes appearing in the boundary layer. These effects are modelled by Thwaites [9], Michel [10] and Head [11] methods, used in laminar, transitional and turbulent boundary layers, respectively. From them, drag coefficient is calculated by using the Squire-Young formula [12], [13]. Nevertheless, as all these methods use the velocity distribution calculated after applying potential flow consideration, when turbulent processes gain importance, the methods' accuracy diminishes.

The effective power produced by each blade is consequently calculated by integrating the effects of each element's airfoil along the blade. To extrapolate 2D airfoil results to 3D blade element, it is necessary to account for the effect of fluid circulation around the blade and its wake. This effect is introduced by applying the Prandtl [14], [15]. In addition, the attack angle of each airfoil is calculated such to be the angle maximizing the drag-lift ratio.

### 3.1 Correction by Neural Network

The range of angles in which it is necessary to carry out the study of the airfoil demands analyzing positions where flow detachment takes place along the surface of the airfoil. In these types of flow configurations the error of the calculations is quite high, since they are based on potential flow studies while, in these cases, the vorticity of the system is high. A neural network was used to correct the deviations that appear when the detachment of the boundary layer takes place below 90% of the chord.

We made use of a simple multilayer perceptron neural network with 6 input neurons (Reynolds number, angle of attack, lift coefficient, drag coefficient, % upper detachment, % lower detachment), two hidden layers of 8 neurons each and 2 output neurons (lift coefficient, drag coefficient). We applied a backpropagation algorithm to obtain the weights of the synapses (training process) using a set of 450 real data. These data were obtained from real analyses from NACA 2410, NACA 0009, NACA 2415, NACA 2541, CLARKY, ClarkySM and DAE11 airfoils [13].

After 1000 steps of training, an average error of around 5% was obtained, showing that this kind of mathematical approach is a very appropriate technique for our problem. We selected a relatively low precision in terms of error in the solution in order to obtain a more general result. Better results for the training error could be achieved, but these led to networks that produced artefacts when new airfoils were analysed. By using this correction, errors in large angle analyses are reduced from over 30% to 8% in the worst cases. The correction achieved by the neural network can be appreciated in Fig. 2.

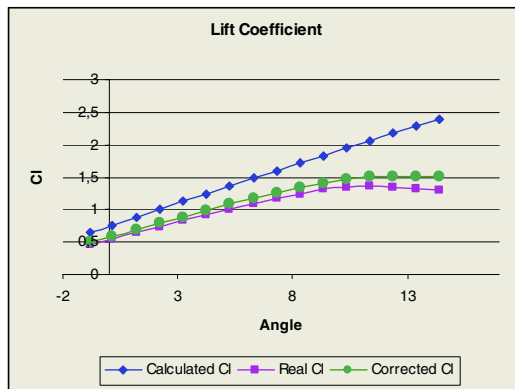


Fig. 2. Real, calculated and corrected lift coefficient for NACA 6409 airfoil. The correction is done by an ANN

The use of neural network based correction implied a significant increase in the accuracy of the method: over 20% in some airfoils. The calculation process, however, only increased by 12 ms over approximately 10s it took for the potential flow calculation.

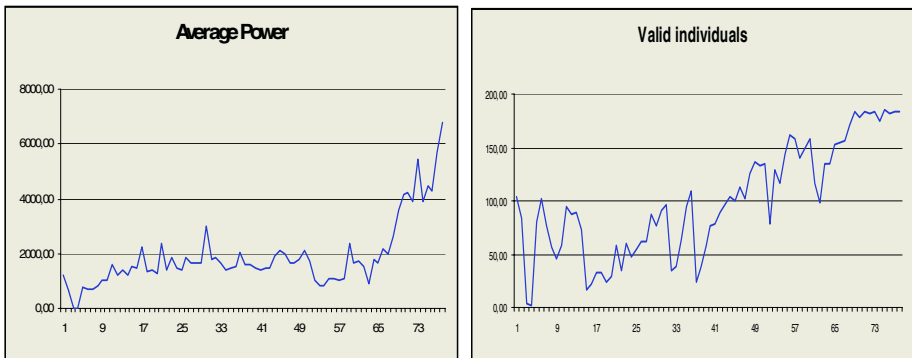
## 4 Experiment

This study is part of a project which purpose is to develop a systematic process for automatic design. We aim to use it as a platform for complex future projects. This requires to use simulation applications and/or commercial calculation and to study the relationships and transmission of information among them.

The use of evolutionary techniques in an automatic design process requires to determine the parameters that define each individual and the requirements that condition the project. In this case these conditions are three: theoretical maximum power to extract, speed of the wind and longitudinal resistance. In the first phase of the project this last constraint has not been introduced.

The objective of this study is to generate a windmill blade whose restrictions are a maximum diameter of 5 m. and maximum wind operation conditions of 10 m/s. A theoretical maximum power, called the Betz limit [16], can be easily obtained for a windmill. This limit is never achieved in reality, but it is useful as a measure of the goodness of a blade by comparison. A simplification is introduced as it is not necessary to contemplate the effect of the terrestrial boundary layer. Rejecting its effect permits considering a potential flow due to the vorticity introduced by the presence of the terrestrial boundary layer.

Since the diameter is taken as an initial restriction, in order to define the blade it will be necessary to define the distributions of chord, twist and airfoil shape along the blade span. In this case we have settled for five halfway control sections, considering the rest of the sections as interpolations from these. In order to define the blade sections or airfoils, we decided to resort to the NACA four digits family of airfoils, which permits their definition through four parameters. Although whole numbers are usually employed to define them, this formulation allows for the use of floating numbers, defining intermediate situations. Thus, the individual genotype is defined by means of twenty-five genes, five per section, where the first four parameters generate the airfoil and the other defines the chord.



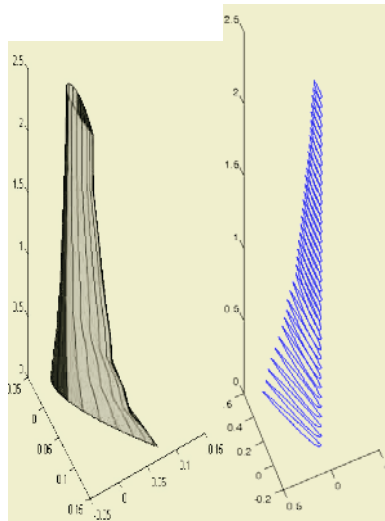
**Fig. 3.** The right figure displays the number of valid individuals during. The left figure displays the average power produced by valid individual is represented



In this project we used a simple genetic algorithm. The population was 200 individuals. The selection was carried out through a tournament scheme among three individuals. The crossover was a simple two point crossover with a rate of 50% and the mutation was random with a rate of 10%.

One of the problems that the algorithm had to face was that there existed chromosomes that did not encode valid individuals. In the first few generations the evolutionary system tried to eliminate the non-valid individuals. When a valid population was stabilized, the population quality started to increase.

In Fig. 3. we provide some data on the evolution process. Before the 65<sup>th</sup> generation, the evolution process was totally functional. In the following generations, the population tended to become genetically stable. The power levels shown in the Fig. 4. are overstated by the simulator due to the simplification of some mechanical effects. Anyway, the power levels achieved by the best individuals come very close to the theoretical optimal.



**Fig. 4.** 3D blade representation for best individual. Notice the twist

To implement the evolutionary process, a cluster of 18 WindowsXP computers was created. We opted for a WINDOWS cluster because great part of the commercial engineering design and numeric simulation applications run under this operating system. This cluster's objective was to make use of idle cpu time of the computers when nobody was using them. To develop the distributed process we used the MPICH2 by Argonne National Laboratory. This conditioned the system vastly since the processes generated by physical users were given priority. It was also decided that the programs were started and closed by the distributed computation program. Since a standardization of the process was sought, this option was chosen despite the fact that it was not the most computationally efficient.

In Fig. 4. 4 we display the best blade obtained for the conditions stated above. This blade produced a power that is very close for the achievable optimum for the case under considerations.

## 5 Conclusions

In this work we have presented an automatic design environment and an aerodynamic simulation strategy for windmill blades that has allowed us to design the blade's profile using a neural network to correct effects having to do with detachment of the flow that cannot be taken into account by potential flow methods. The results obtained clearly meet all of the design parameters that were established leading to very precise behaviours without having to resort to very complicated non-linear models of the plant within the simulator. In the first tests of the design environment, taking some particular conditions into account, near optimal results for the blade were achieved.

## Acknowledgements

This work was funded by the Xunta de Galicia under project PGIDIT03REM08E.

## References

1. Martin O. L. Hansen, "Aerodynamics of Wind Turbines", pp. 80-84.
2. Kirsch, U. & Rozvany, G.I.N.: "Alternative formulations of structural optimization". *Structural Optimization*. 7, pp. 32-41, 1994
3. A. J. Keane, "Wing Optimization Using Design of Experiment, Response Surface, and Data Fusion Methods", *J. Aircraft* 40(4) pp. 741-750 (2003).
4. 122. W. Song, A. J. Keane, M. H. Eres, G. E. Pound, and S. J. Cox, "Two Dimensional Airfoil Optimisation using CFD in a Grid Computing Environment", pp. 50-57 in *Proc. Euro-Par 2003, Klagenfurt* (2003).
5. Y.S. Ong, P.B. Nair, A.J. Keane, and K.C. Wong, "Surrogate-Assisted Evolutionary Optimization Frameworks for High-Fidelity Engineering Design Problems", pp. 307-331 in *Knowledge Incorporation in Evolutionary Computation: Studies in Fuzziness and Soft Computing*, Vol. 167, ed. Yaochu Jin, Springer-Verlag, (2004).
6. Husbands, P., Jermy, G., McIlhagga, M., Ives, R., "Two Applications of Genetic Algorithms to Component Design", *Selected Papers from AISB Workshop on Evolutionary Computing*, Springer-Verlag, Lecture Notes in Computer Science, pp. 50-61, 1996.
7. Lung-an Ying, Pingwen Zhang, "Vortex method", Science Press, pp. 73-78, Beijing, 1994.
8. C. Hirsch, "Numerical computation of internal and external flows", Wiley Series in Numerical Methods in Engineering, Wiley-Interscience Publication, Chichester (UK), 1988, pp. 201-236
9. Thwaites, B. "On the momentum equation in laminar boundary layer flow". ARC RM 2587. 1952.
10. Cebeci, T., Smith, A. M. O.. "Analysis of turbulent Boundary layers". Pp 332-333, 1974
11. Head, M. R. "Entrainment in the turbulent boundary layers". ARC R&M 3152, 1958

12. Martin O. L. Hansen, "Aerodynamics of Wind Turbines", Science Publishers. Ltd, 2000, pp. 11-66.
13. John Moores, Potential Flow 2-Dimensional Vortex Panel Model: Applications to Wing-mills, Master's thesis. University of Toronto. Faculty of Applied Science and Engineering. April 2003.
14. Jones K.D., Davids, S. and Platzaer. "Oscilation Wing Power Generation" ASME paper 99-7050 in Proceedings of the thirds ASME/JSME Joint Fluids Engineering Conference.
15. Abbot, H. and Von Doenhoff, Albert E, "Theory of Wing Sections". Dover (1949; reprinted 1958).
16. Wilson,R.E. "Aerodynamic Performance of Wind Turbines" Energy Research and Development Administration, ERDA/NSF/04014

# Linguistic Properties Based on American Sign Language Recognition with Artificial Neural Networks Using a Sensory Glove and Motion Tracker

Cemil Oz<sup>1,2</sup> and Ming C. Leu<sup>2</sup>

<sup>1</sup> Department of Computer Engineering, University of Sakarya, Turkey

<sup>2</sup> Department of Mechanical and Aerospace Engineering, University of Missouri Rolla, USA  
{ozc, mleu}@umr.edu

**Abstract.** Sign language, which is a highly visual-spatial, linguistically complete and natural language, is the main mode of communication among deaf people. In this paper, an American Sign Language (ASL) word recognition system is being developed using artificial neural networks (ANN) to translate the ASL words into English. The system uses a sensory glove Cyberglove™ and a Flock of Birds® 3-D motion tracker to extract the gesture features. The finger joint angle data obtained from strain gauges in the sensory glove define the hand-shape while the data from the tracker describe the trajectory of hand movement. The trajectory of hand is normalized for increase of the signer position flexibility. The data from these devices are processed by two neural networks, a velocity network and a word recognition network. The velocity network uses hand speed to determine the duration of words. To convey the meaning of a sign, signs are defined by feature vectors such as hand shape, hand location, orientation, movement, bounding box, and distance. The second network is used as a classifier to convert ASL signs into words based on features. We trained and tested our ANN model for 60 ASL words for different number of samples. Our test results show that the accuracy of recognition is 92 %.

## 1 Introduction

American Sign Language (ASL) is the main mode of communication for most deaf people in the United States. However, most people outside the deaf community do not understand ASL. This communication problem affects deaf people's lives and relationships negatively. Deaf people usually communicate with hearing people either through interpreters or text writing. Although interpreters can help the communication between deaf persons and hearing persons, they are often expensive and their use leads to loss of independence and privacy. Communication by writing is used by many deaf people with hearing people, but it is very inconvenient while walking, standing at a distance, or when more than two people are in the conversation. If ASL can be translated automatically into English text and speech, it will be much easier for deaf people to communicate with others. Words in ASL are represented by about 6000 signs, which consist of complex body movements. The signs are created by using the right hand, the left hand or both hands. Some signs also involve facial expressions[1].

Many researchers have studied sign languages (SL) recognition and gesture recognition, but there are major difficulties due to the complexity of hand and body movements in SL expressions. SL recognition research can be categorized into three major classes: (i) computer vision based, (ii) data glove and motion sensor based, and (iii) a combination of these two methods. Computer vision based SL recognition relies on image processing and feature extraction techniques for capturing and classifying body movement and handshape when a deaf person makes a SL sign. On the other hand, data glove and motion tracker based SL recognition methods use a special data glove and a motion tracker for detecting handshape and movement. The third method includes a combination of techniques from each of these two methods.

In parallel to the advancement in sensor and computer technology, some successful computer vision based ASL and other SL recognition systems have been developed [2,3,4,5,6,7,8,9]. There are also many studies in the area of glove and motion sensor based ASL and other SL recognition [10,11,12]. These studies are generally based on Artificial Neural Networks (ANN) [2,10,12,13], Hidden Markov Models (HMM) [3,4,6,7,9], and statistical methods [5,8]. Some recently published papers include the use of both methods for SL recognition [13].

In this paper, we present the design of an ASL word recognition system using ANN. The neural network for word recognition is triggered using a velocity neural network. If the response of the velocity network is positive, then the feature vectors, which are based on linguistic properties of ASL, are extracted by using the data from the Cyberglove™ and Flock of Birds® for every time cycle until the response of the velocity network becomes negative. We do not use all of these extracted features for actual word recognition, but we choose the feature vectors at six time instants. The system is able to recognize a vocabulary of 60 ASL words successfully.

## 2 Hardware of System

We use a right-hand Cyberglove™ to retrieve the finger joint angle values for gesture features. The glove has 18 sensors which measure the bending angles at various positions when the frequency of data collection is up to 150 Hz. We have used 15 sensors of the glove. To track the position and orientation of the hand in 3-D space, the Flock of Birds® motion tracker mounted on the wrist is used.

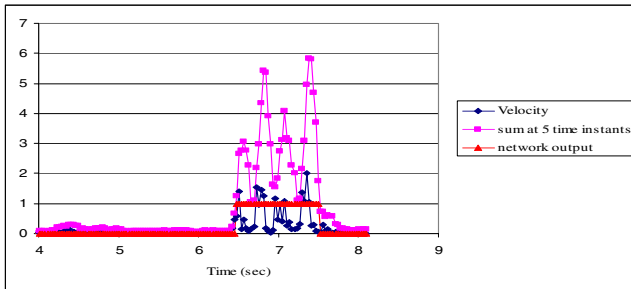
Open Inventor SDK (Software Development Kit) is used in the software development for the 3-D scene rendering and interactive programming. We use the Microsoft® Speech SDK for the programming of speech synthesis. Figure 1 shows the system input devices.



**Fig. 1.** The image on the right is Cyberglove™. The image on the left is Flock of birds® 3-D motion tracker

### 3 Data Collection for Word Recognition

A velocity network is designed which classifies the hand movement at any time instant as “signing” (1) or “not signing” (0). It is detailed in our earlier paper [14]. Figure 2 shows an example of the hand velocity profile during a typical signing movement. During the signing movement, the hand velocity can rise or drop momentarily due to momentary shaking and stopping of the hand. Hence, the use of a threshold value of velocity does not give a good solution for classification of hand movement. The ideal output for the shown velocity graph should be 1 from the time the first sudden change in velocity is seen till the time the velocity graph shows a series of low velocities. Hence, after numerous trials, the input to the velocity neural network was selected as the 'summation of velocities during the previous five time instants'.



**Fig. 2.** Velocity network performance during determining the duration of ASL words

If the output of velocity network is 1, feature vectors, hand-shape, hand-location, hand-orientation and movement are extracted using the 15 data elements from the glove and 6 data elements from motion tracker, then feature vectors and sensors data are recorded in an array. This data is collected at a frequency of 33.33 Hz. No features are extracted, and sensor data are recorded if the output of the velocity network is zero. When the speed network output changes from 1 to 0, the features and data recording for the given sign are completed, and the sign recognition phase starts. The data at five equally divided time instants are chosen from the recorded data, giving a total of 300 data elements. In our study, a virtual hand model in VRML displayed on the computer screen is updated at 33.33 Hz using the data gathered from the glove and motion tracker.

### 4 Feature Extractions

In this study, the feature vectors of a sign are extracted based on an earlier linguistic description of ASL, which were defined by Stokoe and Battison [15,16], Their purpose was developing a national system for writing sign, containing symbols defined by hand-shape, location, movement and orientation. Distance, time and bounding box are also used additionally.

**Distance and time.** Each time a speaker signs an ASL word, information is instantly gathered as to x, y, z position, wrist rotation and finger bending. Let  $x_i, y_i, z_i$  be the x-position, y-position and z-position stored in the  $i$ th cycle of the sample. A change in direction from the last cycle is indicated by the vector  $(\Delta x_i, \Delta y_i, \Delta z_i)$  pointing in the direction the signer is going when

$$\begin{aligned} \Delta x_i &= x_i - x_{i-1} \\ \Delta y_i &= y_i - y_{i-1} \\ \Delta z_i &= z_i - z_{i-1} \end{aligned} \tag{1}$$

Thus, if we take the length of this vector and call it

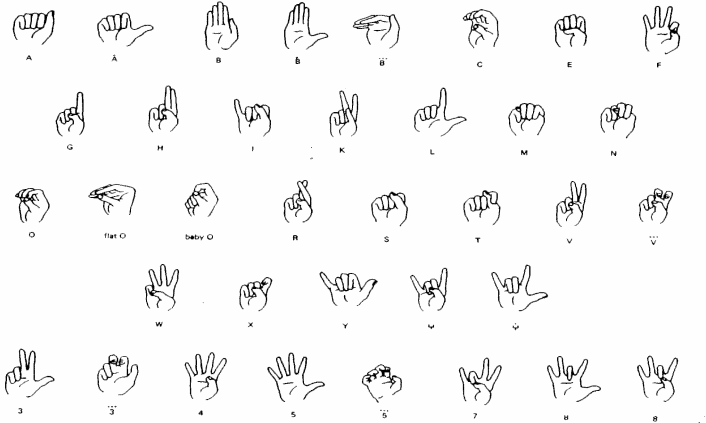
$$\Delta_i = \sqrt{\Delta x_i^2 + \Delta y_i^2 + \Delta z_i^2} \tag{2}$$

then the sum of  $\Delta_i$  would give us a measure of total distance covered by the sign. Every sign covers a different distance from others. We define the feature distance as:

$$\text{Distance} = \sum_{i=1}^n \Delta_i \tag{3}$$

where  $n$  is the number of cycles in the sample.

**Bounding boxes.** Some signs are bigger than others and differ in their location around the body. The whole sign fits in the box represented by  $(x_{\min}, y_{\min}, z_{\min})$  and  $(x_{\max}, y_{\max}, z_{\max})$ . So we can use this bounding box as a feature vector. Our working space boundaries were between  $(-30, -30, -30)$  and  $(30, 30, 30)$ , and we normalized bounding boxes data from 0 to 1.



**Fig. 3.** American Sign Language hand-shapes [15]

**Hand shape.** Most of the hand shapes are the ASL alphabet, basic numbers and some possible hand shapes for signing words. ASL has 36 hand shapes. We designed an

ANN model for hand shape recognition. Hand shape recognition depends on finger bending data which are collected from the CyberGlove. Our ANN hand shape recognition model consists of 15 input neurons, 28 hidden neurons and 36 output neurons. Figure 3 shows ASL hand shapes.

We trained and tested our model for both single users and multiple users. It has an accuracy of 98%. This model is not given in this paper because it was detailed in an earlier paper [1]. We used hand shapes as a feature for sign recognition. The drawback of this system is that it has a glove constant-size. This glove produces different data for different hand sizes. Therefore, the system can only be reliable for a hand which has the same size as the training hand.

**Hand Location.** One of our feature vectors is hand position. It is based on four measurement points (a), (b), (c), and (d) as shown in fig. 4. Sign speaking space is designated as a rectangular prism shape and divided into 14 regions by using four measurement points. The (d) point is measured when the hand is parallel to the floor and the palm faces the floor. According to our system, (a), (b) and (c) points are used for dividing a two dimensional coordinate system; (d) is used for the third dimension. These points are used together for dividing the action space into regions. Every signer has to establish his own reference points, which can then be called up from a stored file at any time. Figure 4 shows how we form the hand position locations.

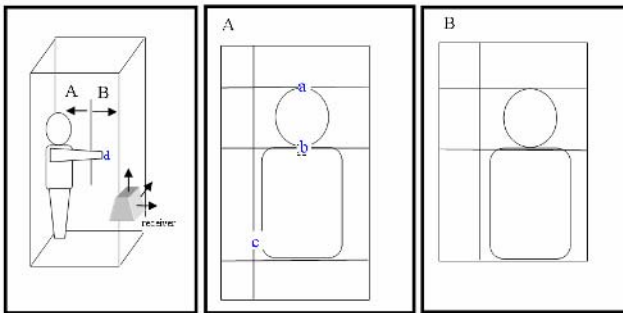


Fig. 4. Location of hand position

**Orientation.** Orientation features are produced with a neural network classifier using 3 of Flock of Birds' alpha, beta, and gamma sensors. We used a total of six orientations; palm facing up, palm facing down, palm facing left, palm facing right, palm facing front, and palm facing body. The accuracy of orientation module is 95.7%.



**Movement.** Movement features are produced by calculating differences between present and previous coordinate values for every time instant. There are three axes and every axis has three states; zero, minus, and positive. There are 27 features in total.

## 5 Design of ASL Word Recognition System

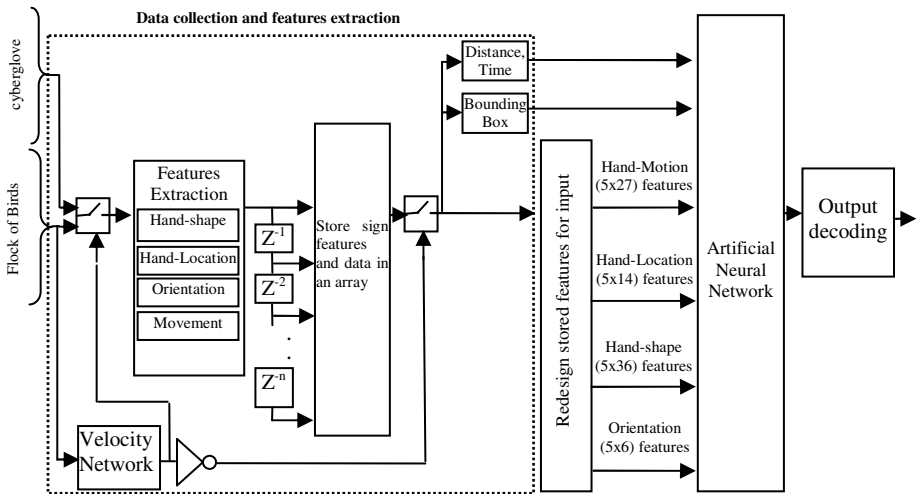
Two of the ASL words used in the training set for the ANN are given in Table 1 as example. When we look at the definition, we see that each word has a start position with a hand shape and continues by changing hand shape and position.



**Table 1.** Two words in American Sign Language [1]

| ASL words                                                                                | Definition                                                                                                                                              |
|------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------|
| <br>and | The right "5" hand, palm facing the body, finger facing left, moves from left to right, meanwhile closing until all its finger touch around it's thumb. |
| <br>Are | The right hand "R" position, palm facing left, is held at the lips, and the hand moves straight out and away from the lips.                             |

A multi-layer ANN is designed to recognize all possible ASL words. Although the developed system is fully flexible for any number of words, in demonstration the system is trained for recognition of 60 words in ASL. The input to the network is the extracted feature vectors of ASL words, which are produced from collected data during the signing period. The signing period is the time period over which the output of velocity network is 1. At each time interval, the data from Cyberglove and Flock of Birds and extracted features are recorded. We used a total of 300 data for every sign, two for distance and time, six for bounding box, 180 (36X5) for hand-shape, 70 (14X5) for hand-location, 30 (6x5) for hand-orientation, 135 (27x5) for movement. The input to the network is the 300 data elements at five time instants recorded by the data collection algorithm. The proposed ANN, a multi-layer feedforward network, consists of 300 input neurons, 100 hidden neurons, and 60 output neurons and it is



**Fig. 5.** Block diagram of ASL word recognition system

designed to recognize the isolated words in ASL. A Levenberg-Marquardt backpropagation algorithm is used for training. The ANN is trained and tested for two

different data sets, a single-user data and a multi-user data. The output vector consists of 60 elements, the maximum of which corresponds to a word or sign.

The design of the recognition system is shown in fig. 5. It consists of four parts: data collection and features extraction, selection of input data, trained network, and output decoding.

In the data collection and features extraction, the velocity network triggers the data recording and feature extraction. The sensor data and extracted features are recorded until the velocity network output becomes zero. As explained earlier, the data at six time instants are selected by the selection of input data part in Fig. 5. Every input group represents a feature vector, which is selected from the recorded data, corresponding to six selected time instants. This data of 300 elements is an input to the ANN. The three-layer word recognition network has 300 inputs, 100 hidden neurons and 60 output neurons. In the final part, the output selection block determines the maximum output of ANN, and then the threshold block checks whether the maximum output value is acceptable or not. If it is acceptable, the corresponding index of the output vector is the recognized word. If it is not acceptable, the recognized word is considered to be out of the known ASL vocabulary. The recognized word is displayed on the screen using the Open Inventor interface. As illustration, two of the ASL words recognized by our system are shown in Fig. 6.



Fig. 6. Examples of recognized signs by our system

## 6 Test Results

We tested the model with single-user and multi-user data. In both tests, the word recognition system trained with three, six and twelve samples of data with 60 words. At the testing stage, the real time data were used. In total, 360 ASL words were in the training set used for the test. Both the single user model and the multiple-user model were tested sequentially. The model tested started with *abortion* and ended with *beautiful* in alphabetical order, and then it continued with ten additional words which were taken randomly like *you*, *I*, *yes* etc. Some hand-shape feature extraction data were signed by deaf teachers from the Missouri School for Deaf. The word recognition system testing results are given in the Table 2.

**Table 2.** ASL word recognition system test results

|                              | Single person trained ANN |     |     | Multiple person trained ANN |     |     |
|------------------------------|---------------------------|-----|-----|-----------------------------|-----|-----|
| Training data for every word | 3                         | 6   | 12  | 3                           | 6   | 12  |
| Total test data              | 360 (6*60)                |     |     | 360 (6*60)                  |     |     |
| Missed                       | 42                        | 20  | 14  | 43                          | 23  | 16  |
| Misrecognized                | 29                        | 20  | 16  | 35                          | 19  | 20  |
| Recognized                   | 289                       | 320 | 330 | 282                         | 318 | 324 |

## 7 Conclusion

A real-time American Sign Language word recognition system is being developed using ANN to translate the ASL words into English. The data from sensors of CyberGlove and Flock of Birds are processed by a velocity network, a feature extraction module, and a word recognition network. Features are extracted using sensor data based on linguistic properties of ASL words. The system was trained and tested for single and multiple users for a vocabulary of 60 ASL words. Test results have shown that the proposed technique is capable of recognition in real time and the recognition accuracy of the system is 92 %.

## References

1. Sternberg, M. L. A.: American sign language dictionary (Revised Edition). In: Harper Perennial, ISBN: 0-06-273275-7, (1994)
2. Wysoski, S. G., Lamar, M. V., Kuroyanagi, S., Iwata, A.: A rotation invariant approach on static gesture recognition using boundary histograms and neural networks. In: Proceeding of the 9<sup>th</sup> International conference on neural information processing , vol. 4, (2002) 2137-2141
3. Vogler, C., Metaxas, D.:ASL Recognition based on a coupling between HMMs and 3D motion analysis. In: proceedings of sixth IEEE International Computer Vision Conference, January (1998) .363-369.
4. Lalit,G., Sui, M.: Gesture-based interaction and communication: automated classification of hand gesture contours. In: IEEE Transactions on Systems, Man, and Cybernetics part c: Application and Reviews, vol. 31, no. 1, February (2001) 114-120.
5. Yang,M. H., Ahuja, N.: Recognizing hand gestures using motion trajectories. In: IEEE, (1999) 466-472.
6. Stanner, T., Pentland, A:Real-time American sign language recognition from video using hidden markov models. In: Computer Vision International Symposium Proceeding, (1995) 265-270.
7. Vogler, C., Metaxas, D.: Adapting hidden markov models for ASL recognition by using three-dimensional computer vision methods. In: Proceeding of the IEEE International Conference on Systems, Man and Cybernetics, Orlando, (1997) 12-15.
8. Cui,Y., Weng, J.: A learning-based prediction and verification segmentation scheme for hand sign image sequence”, IEEE transactions on Pattern Analysis and Machine Intelligence, vol. 21, no.8, August (1999) 798-804.

9. Vogler, C., Metaxas, D.: Parallel hidden markov models for American sign language recognition. In: IEEE proceeding of International Computer Vision Conference, September (1999) 116-122.
10. Allen, M. J., Asselin, P. K., Foulds, R.: American Sign Language Finger Spelling Recognition System. In: 29<sup>th</sup> Bioengineering Conference proceeding, IEEE, (2003) 22-23.
11. De Marco, R. M., Foulds, R. A.: Data Recording and Analysis of American Sign Language. In: IEEE 29<sup>th</sup> proceeding of Bioengineering Conference, (2003), pp. 49-50.
12. Wang, H. G., Sarawate, N. N., Leu, M. C.: Recognition of American sign language gestures with a sensory glove. In: Japan USA Symposium on Flexible Automation, Denver, CO, July (2004)
13. Lee, L. K., Kim, S., Choi, Y. K., Lee, M. H.: "Recognition of hand gesture to human-computer interaction", IEEE, (2000) 2177-2122.
14. Oz, C., Sarawate, N. N., Leu, M. C.: American sign language word recognition with a sensory glove using artificial neural networks. In: Asme Press, Intelligent Engineering Systems through Artificial Neural Networks, Vol. 14, ISBN 0-7918-0228-0, (2004) 633-638.
15. Wilbur, R. B.: American sign language linguistic and applied dimensions. In: College-Hill Press, second edition, ISBN 0-316-94013-5, (1987)
16. Battison, R.: Lexical borrowing in American sign language. In: Silver Spring, MD : Linstok Pr., ISBN: 0-932130-02.,(1978)

# Crack Detection in Wooden Pallets Using the Wavelet Transform of the Histogram of Connected Elements

M.A. Patricio<sup>1</sup>, D. Maravall<sup>2</sup>, L. Usero<sup>1</sup>, and J. Rejón<sup>1</sup>

<sup>1</sup> Dpto. Ciencias de la Computación, Universidad de Alcalá, Spain

<sup>2</sup> Dpto. de Inteligencia Artificial, Universidad Politécnica de Madrid, Spain

{miguel.patricio, luis.usero, julio.rejon}@uah.es

dmaravall@fi.upm.es

**Abstract.** The paper presents the application of the wavelet transform of the frequency histogram of connected elements to the detection of very thin cracks in used pallets. First, the paper presents this novel concept and introduces the parameters that define a connected element, showing that the conventional grayscale intensity histogram of a digital image is a particular case of the histogram of connected elements. Then, the discriminant capability of the wavelet transform of this generalized histogram is analyzed. In particular, the information conveyed by the histogram of connected elements is exploited to detect very thin cracks in used pallets. An artificial neural network classifier to discriminate sound wood from defective wood with very thin cracks has been designed. The exhaustive experimental test carried out with numerous boards of used pallets has validated the proposed method, in particular its remarkably low ratio of false alarms.

## 1 Introduction

The authors have been working for several years on the development of automated wooden pallet inspection systems using computer vision techniques, and as a result of this work there are several industrial inspection plants in full operation in a number of European countries. The objective of the automatic inspection of pallets is to gradually substitute manual inspection, which, apart from being a hard and painstaking job, is prone to errors, both false alarms -i.e., ready-for-use pallets classified as defective- and false negatives. After several years of R+D efforts, the ratios currently achieved by the inspection systems in which the authors have been working are comparable to those of qualified human operators. Concerning operating times, automation inspection has surpassed manual operation by almost one order of magnitude: 3 seconds versus 20 seconds on average, respectively.

The pallets arrive at the visual inspection plants after a period of several months of use. This means that inspection can be really complex as many potential defects may be present: splinters, cracks, fissures, broken elements, lack of

volume in several elements of the object, etc. To make things worse, the surface of wooden pallets is often stained with different kinds of remains, such as grease, mud, paint and so on. This paper focuses on the detection of very thin cracks -in the 1 mm range-. The main difficulty in recognizing thin cracks does not lie in actual detection -which can be successfully performed when applying appropriate segmentation techniques and provided that there is enough resolution in the digital images- but in the numerous false alarms generated by wood veins, paint remains, shadows induced by the lighting system, etc. Roughly speaking, the better the detection ratio of thin cracks is, the higher the false alarm ratio is, which means that a very delicate balance between both ratios has to be struck.

As a consequence of our work on this hard recognition problem, we have introduced a novel concept for digital image segmentation: the so-called frequency histogram of connected elements (FHCE), [1]. Being a conventional unidimensional histogram, the FHCE incorporates all the computational advantages, in terms of both simplicity and speed, inherent to histogram-based segmentation methods. At the same time, it includes information about the spatial distribution of the specific discriminant feature in the digital image, as bidimensional histograms also do.

Taking into account how much information that the FHCE conveys as compared with the conventional gray-level histogram, we explore the application of the FHCE wavelet transform to texture discrimination in this paper. Our main objective in this paper is to compare the discriminant capability of different mother functions, also known as analyzing wavelets. More specifically, we are looking for the best mother functions for dividing regions of a digital image into two classes: sound wood and defective wood.

The paper has been divided in to two parts. The first one is focused on the FHCE wavelet transform. We then address the selection of discriminant features based on wavelet functions. The paper ends with the discussion of the experimental results concerning the detection of very thin cracks.

## 2 FHCE Wavelet Transform

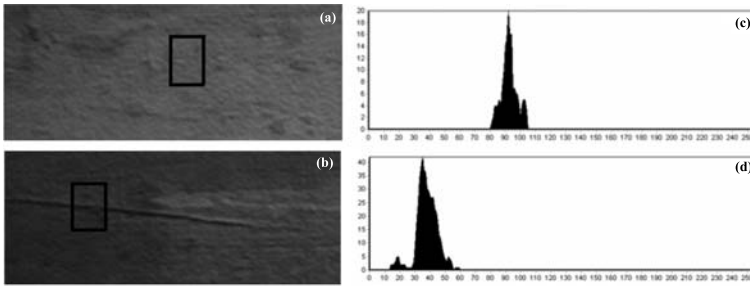
In textured images, as is well known, the basic idea is to apply a window whose size is big enough to capture the essential structure of any texture present in the image. In our particular application of detecting thin cracks, we have found that a window of  $40 \times 30$  pixels seems to be optimum in most cases.

In our process design, we have to decide about the two FHCE parameters: *spatial predicate* and *connectivity level* (see [1]). After exhaustive experimentation with a plethora of digital images of sound and defective wooden boards, we have selected a  $5 \times 3$  window as the spatial predicate or morphological structure. Note that the number of horizontal pixels is higher than the vertical pixels, owing to the a priori knowledge available about the problem at hand. In fact, there is empirical evidence that cracks in a piece of wood tend to appear in the same direction as the wood grain. As the computer vision inspection is

performed horizontally from the wooden boards standpoint, the shape of the selected neighborhood function is easily deduced.

To conclude the selection of the FHCE parameters, the connectivity level that a particular morphological structure should possess to be considered as such must be selected. The FHCE is computed for each image portion by moving a window of the same 5x3 shape as the neighborhood across all the pixels. This scanning process is performed by means of a top-bottom and left-right movement and by computing, at each pixel, the maximum and the minimum gray level within its neighborhood. Each pixel's neighborhood is classified as a connected element if and only if the difference between the maximum and the minimum values is small as compared with the dynamic range of the histogram in the whole window. After experimental work, we have chosen a 10% ratio, which is a good compromise between wooden portions in good and bad conditions. Therefore, if a particular neighborhood possesses a gray-level variability of less than ten percent of the dynamic range of the global window, the respective pixel is a connected element and the FHCE will compute a new event.

Figure 1 illustrates an example of FHCE computation for sound wood and defective wood.

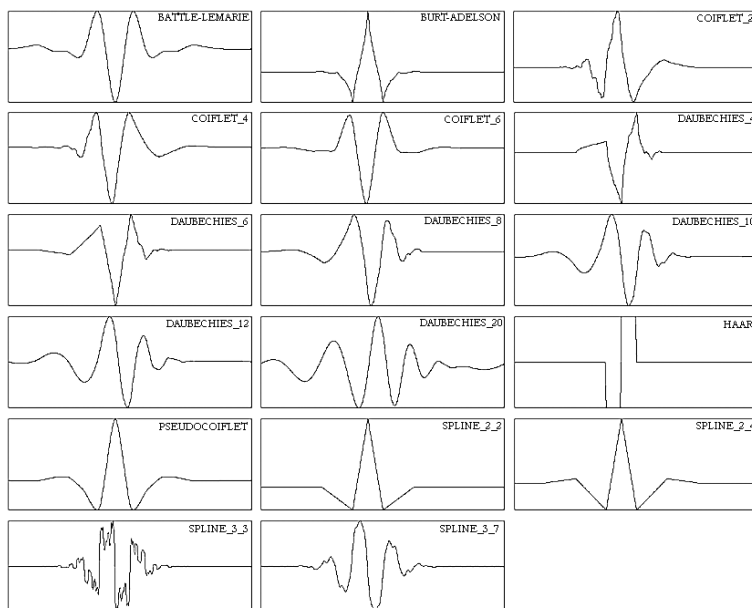


**Fig. 1.** Instances of FHCE for digital images of (a) sound and (b) defective wood. The FHCE shown in the right-hand column match the windows appearing in the digital images in the left-hand column. Note that the main differences between the two instances are on the left-hand side of the FHCE

It is clear from Figure 1, that there is a distinction between the FHCE for portions of sound wood and for defective wood with cracks, respectively. Note, in particular, the left side of the FHCE: whereas side lobes appear in the image for defective wood, this characteristic is not present in the FHCE of sound wood. Because each FHCE is shaped quite differently, we are in an excellent starting position to embark upon the task of selecting optimal features for the automatic wood classification. We can view the FHCE of a window for a portion of wood with a defect –a crack in our case– as a signal with two distributions. The first distribution corresponds to the connected elements that belong to the defect –i.e. the crack– and the second distribution to the connected elements belonging to the sound wood portion. Moreover, it is quite evident that the sound

wood portion has a FHCE with a single distribution. In conclusion, there is clear evidence that sound wood can be discriminated from defective wood using the low frequency components of the FHCE spectrum [2] –i.e. by computing the Fourier transform of the FHCE-. However, as the FHCE of a textured region is a non-stationary function, it is advisable to work with spatio-temporal representations. The wavelet transform is able to represent both the spatial and temporal domains of a particular function.

Broadly speaking, the wavelet transform is a mathematical tool that divides a function into a set of functions, which are in turn scaled and shifted versions of a primitive function, called the mother wavelet function. In principle, there is an infinite number of functions that hold the condition of being mother wavelet functions, but only a few of them have been used in practice. Figure 2 shows the mother functions used in our work.



**Fig. 2.** Mother functions used in our experimentation: BATTLE-LEMARIE [3]; BURT-ADELSON [4]; COIFLET\_2, COIFLET\_4, COIFLET\_6 [5]; DAUBECHIES\_4, DAUBECHIES\_6, DAUBECHIES\_8, DAUBECHIES\_10, DAUBECHIES\_12, DAUBECHIES\_20 [4]; HAAR [4]; PSEUDOCOIFLET [6]; SPLINE\_2\_2, SPLINE\_2\_4, SPLINE\_3\_3, SPLINE\_3\_7 [4]

Although the wavelet transform has been intensively applied as a powerful and efficient tool in the pattern recognition discipline, it really has a clear limitation as regards time shift. Let  $g(k)$  and  $f(k)$  be two generic FHCEs and  $WTg(u)$  and  $WTf(u)$  their respective wavelets transforms, then



$$\begin{aligned}
 g(k) &= f(k + \tau) \text{ \textbf{does not mean that}} \\
 WTg(u) &= W Tf(u + \tau)
 \end{aligned}
 \tag{1}$$

This is a significant drawback [3] affecting the usual pyramidal multiscale representation. Although several attempts to solve this problem have appeared in the technical literature, we have chosen an alternative way based on applying a pre-processing technique to recognize the normalization of the FHCE. To determine whether a particular FHCE belongs to a sound wood or to a defective window, we confine our analysis to the left half of the respective FHCE and then apply the following algorithm.

**Step 1.** Let  $h(k)$  be a generic FHCE, then compute the following parameters:

- o  $g_{min}$ : the lowest gray level in  $h(k)$  with frequency higher than zero.
- o  $g_{max}$ : gray level in  $h(k)$  with the highest frequency.
- o  $max$ : maximum frequency in  $h(k)$  for  $h(k_{max})$ .

**Step 2.** Compute the normalized FHCE as follows:

$$h^*(k) = \begin{cases} \frac{h(g_{min}+k)}{max} * 100 & \text{for } 0 \leq k \leq g_{max} - g_{min} \\ 0 & \text{for } g_{max} - g_{min} < k \leq N - 1 \end{cases}
 \tag{2}$$

Finally, the feature vector is put together from the application of the wavelet transform to the normalized FHCE. We also use an alternative normalization based on the definition of the energy distribution of the wavelet transform [7]. Therefore, the feature vector contains 4744 discriminant variables (18 transforms –17 wavelet transforms plus 1 fourier transform– by 256 coefficients for each transform, plus 17 wavelet transforms by 8 energy levels).

### 3 Feature Selection

Obviously, the number of discriminant features is too high (as we mentioned above 4744), and many of them are irrelevant for discrimination purposes.

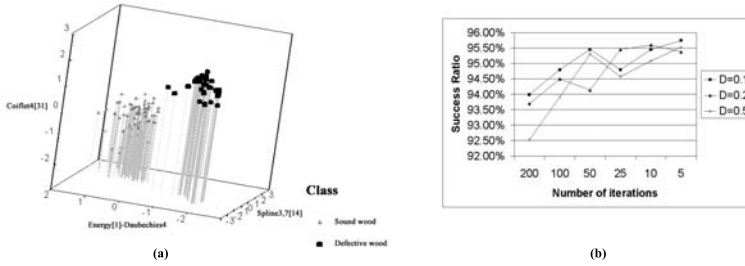
We have selected 60 boards of defective wood –i.e. with cracks- and 150 boards of sound wood, and we have run a program to automatically generate a set of training and testing samples from these 210 boards. After exhaustive experimentation, the optimum window size chosen for the generation of digital image samples was  $40 \times 30$ . Finally, for benchmark purposes, 1365 digital image samples (973 of sound wood and 392 of defective wood) have been employed. Note that sound wood samples account for approximately 70% of the complete set. This proportion is quite similar to the rate of occurrence in real-life inspections.

To reduce the number of discriminant variables, we have applied the so-called incorporation method [8] and we have output the following discriminant features as the optimum subset:

- o  $\frac{\text{Coiflet4}(31)}{\text{Coiflet4}}$ . Coefficient 31 of the wavelet transform whose mother function is

- o Energy(1).Daubechies4. Energy percentage of level 1 of the Daubechies4 wavelet transform.
- o Spline3,7(14). Coefficient 14 of the wavelet transform whose mother function is Spline 3,7.

Figure 3(a) shows the scatter diagram of 100 samples taken at random from the 1365 discriminant vectors used in our experimental tests. Obviously, each coordinate matches one of the three selected discriminant variables.



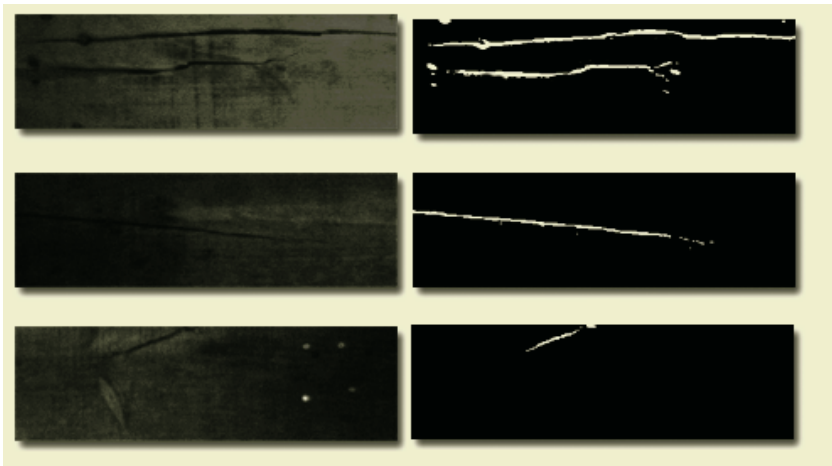
**Fig. 3.** (a) Scatter diagram of 100 samples taken at random from the experimental data. (b) Results of the ANN in crack detection and recognition

## 4 Classifier Design and Experimental Results

Having selected a particular feature vector, the next step is classifier design. In this respect, we have chosen a classifier based on artificial neural networks (ANN), as our own experience with many pattern recognition techniques indicates that ANNs are particularly well suited for dealing with the problem of discriminating textures of sound wood from textures of defective wood -in our case, with a crack-. More specifically, of the numerous existing ANN models, we have focused on the feedforward multilayer perceptron (MLP), trained with the backpropagation (BP) algorithm, as the detection ratio and the false alarms ratio can be balanced across a judicious selection of the MLP parameters: namely, the learning coefficient, the number of iterations and the maximum permitted deviation, [9].

The most demanding and reliable method for evaluating any automatic classifier is the *leave-one-out* policy. Using this policy, the classifier -in our case, the feedforward multilayer perceptron- is trained with all the available training samples except one, which is used to evaluate the classifier itself. By repeating the process with all the training samples, the average success ratio of the classifier is an excellent estimation of its future performance for new samples or cases, i.e. working in real-life situations. The average success ratio is computed as follows:

$$P = f(\text{defect}/\text{defect}) * f(\text{defect}) + f(\text{good}/\text{good}) * f(\text{good}) \quad (3)$$



**Fig. 4.** Several wood boards with very thin cracks (left hand) and the binarization of the wood boards using the method described in the paper (right hand). Note that all the existing thin cracks have been detected and that there are no false alarms which is a real plague for any method aimed at detecting very thin cracks in wood

where  $f(\text{defect})$  and  $f(\text{good})$  are the relative frequency of a defective and a good sample, respectively;  $f(\text{defect/defect})$  and  $f(\text{good/good})$  are the a posteriori relative frequency of success for defective and good samples, respectively.

The evaluation was carried out using different values for the maximum number of iterations and maximum deviation design parameters. Just to give an idea of the evaluation results, Figure 3(b) shows the results obtained for maximum deviation = (0.1; 0.2; 0.3) and for several values of the maximum number of iterations. The best result is obtained with a maximum deviation of 0.1 and for 5 iterations (96.52%). Another remarkable result that it is worth mentioning in our experimentation is the Fourier transform of the FHCE performed poorly, as compared with the wavelet transforms.

The final detection of cracks is based on scanning the whole image with a small window with a horizontal and vertical overlapping. For each  $40 \times 30$  individual window, the ANN-based recognizer makes the decision to classify the window either as sound wood or as defective wood. For every window labelled as defective wood, the next step is to segment the pixels of the crack, which can be done straightforwardly by an adaptive thresholding algorithm [1], because of the clear bimodal nature of the FHCE, in which one distribution is formed by the connected elements of the crack and the other by pixels of the sound wood.

Figure 4 shows several wood boards with very thin cracks and the excellent crack detection results achieved by the proposed method. These results have been achieved thanks to the information conveyed by the histogram of connected elements.

## 5 Conclusion

In this paper, we have exploited the information conveyed by the histogram of connected elements to address the detection of very thin cracks. We have also investigated the discriminant capability of the wavelet transform of the histogram of connected elements and designed an ANN-based classifier to discriminate sound wood from defective wood. The signal inputs of the ANN classifier are three components of a wavelet transform of the histogram of connected elements. An exhaustive experimental test has validated the proposed method for detecting very thin cracks. As it is depicted in Figure 4, an additional advantage is the very low ratio of false alarms achieved by the proposed method.

## References

1. D. Maravall and M. A. Patricio, "Image Segmentation and Pattern Recognition: A Novel Concept, the Histogram of Connected Elements" in *Pattern Recognition and String Matching*, Kluwer Academic Publishers, 2002, pp. 399–452.
2. R. Bajcsy, "Computer description of textured surfaces". *Proceedings of the 3<sup>rd</sup> International Conference in Artificial Intelligence*, 1973, pp. 572–579.
3. S. G. Mallat, "Multifrequency channel decomposition of images and wavelet models", *IEEE Trans. on Acoustics, Speech and Singal Processing* Vol 37, 1989, pp. 2091–2110.
4. I. Daubechies, *Ten lectures on wavelets*. CBMS-NSF regional conference series in applied mathematics, 61. 2<sup>nd</sup> ed. Phipadelphia: SIAM, 1992.
5. T. Beylkin, R. Coifman, and V. Rokhlin, "Fast Wavelet Transforms and Numerical Algorithms I". *Comm. Pure Appl. Math.*, Vol. 44, 1991, pp. 141–183.
6. L. M. Reissell, "Multiresolution Geometric Algorithms Using Wavelets I: Representation for Parametric Curves and Surfaces". *Technical Report 93-17*, Departament of Computer Science, University of British Columbia, Vancouver, May 1993.
7. A. Laine and J. Fan, "Texture classification by wavelet packet signatures". *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 15, No. 11, 1993, pp. 1186–1191.
8. M. A. Patricio and D. Maravall, "Segmentation of text and graphics/image using graylevel histogram Fourier transform", in *Proc. of the SSPR&SPR*, LNCS 1876, Springer-Verlag, 2000, pp. 757–766.
9. A. K. Jain, J. Mao and K. Moidin, "Artificial neural networks: a tutorial". *Computer*, Vol. 29, No. 3, 1996, pp. 31–44.

# A Competitive-Based Method for Determining the Number of Groups: A Clinical Application

Antonio Sánchez, Francisco Vico<sup>1</sup>, Santiago Cabello<sup>1</sup>, Francisco Veredas<sup>1</sup>, Isaac López<sup>1</sup>, Yamina Seamari<sup>1</sup>, Javier Farfán<sup>2</sup>, and Guillermo García-Herrera<sup>2</sup>

<sup>1</sup> Dept. Lenguajes y Ciencias de la Computación, ETSII Informática,  
Universidad de Málaga, Málaga, Spain

{fjv, ilm, scc, fvn, ysm}@geb.uma.es  
<http://www.geb.uma.es>

<sup>2</sup> Hospital General de Antequera, Sección de Traumatología,  
Antequera, Málaga, Spain  
doctorfarfan@yahoo.com, ficot@mixmail.com

**Abstract.** A proper gait assessment in patients with knee or hip injuries strongly determines the diagnosis and consequently the evolution of the pathology, the quality of life of implanted patients, and the overall costs involved. Among the different strategies to clinically assess gait, 3D optical tracking provides a reliable and objective evaluation. This method involves state-of-the-art image analysis that performs anatomical measurements upon bony landmarks identified by markers attached to the patient. We show how this technology can be used to perform patients diagnosis and follow-up by grouping the results of gait measurement with a competitive neural network where the number of clusters is automatically determined.

## 1 Introduction

3D optical tracking has been extensively applied in recent years [1], [2], [3], [4], [5], [6], [7], [8], [9], either with expensive commercial set-ups [1], [2], [3], [4], [7], [8], or with home-made cheap machinery [6], [9]. Both approaches provide a reliable method to assess human gait in clinic and high performance applications [1], [2], [4], [5], [6], [8], [9].

This new field has been widely analyzed from the computer vision perspective, while data processing still remains open to researchers. Most efforts has been done with conventional statistical techniques, but the application of artificial neural networks in gait pattern identification has inspired very few works [10], [11], [12].

The following results show that competitive learning is an alternative to SOM for gait patterns generation.

## 2 Data Acquisition for Gait Analysis

Human gait was recorded in a room wide enough for the subjects to perform a gait start, a normal gait (including at least one complete cycle for each leg) and a gait end.

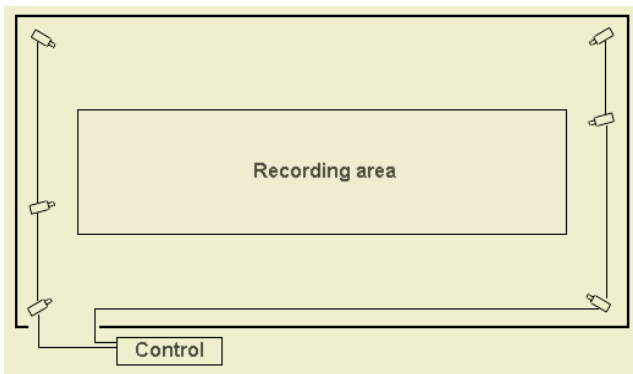
MJPEG video was recorded from six synchronized PCs, each connected to a camera (1/500 sec shutter speed) as shown in Fig. 1. Each video stream contains a particular view of the gait, where the markers reflect the highest level of intensity of light in the image, since the scene is illuminated with six halogen focus (300 W), each on top of a camera. This arrangement maximizes the light returned by the reflecting material that covers the markers.

Dedicated software was developed to automatically obtain the 2D coordinates of the markers in every scene, that merged into 3D coordinates. This operation is performed using DLT (Direct Linear Transformation) coefficients [13] generated upon a calibrating structure. This information allows a reliable three-dimensional reconstruction of the walking subject, and the precise computation of a number of measurement (distances and angles) that characterize human gait.

In order to automatically detect the patient's anatomical coordinates, 19 markers were placed on characteristic bony landmarks on the legs and hip of the patient (sacrum, up-front iliac bone, lateral-external area at the bottom of the thigh, frontal area at the bottom of the thigh, tibia -upper frontal area-, fibula -upper external area-, ankle, second metatarsal head, and heel). These markers are spherical, reflecting, hollow wood balls, 2.5 mm diameter, fixed to the skin with clinical sticking tape. The reflecting cover makes possible the automatic detection in the image (Fig. 2).

The experiment was performed on 166 patients, classified in three groups: healthy, pathologic, and implanted patients (with a *PFC*<sup>®</sup> *Sigma*<sup>™</sup> prosthesis<sup>1</sup> of either type simple, non-rotating anatomic, or non-rotating stable).

Dedicated software was coded to automatically obtain the 3D coordinates of each marker, and to compute anatomic measurements (distances and angles) upon these coordinates. The selected measurements were: pelvic obliquity, hip obliquity, knee obliquity, pelvic tilt, hip flex-extension, knee flex-extension, plantar flexion, pelvic rotation, and ankle rotation<sup>2</sup>.



**Fig. 1.** Arrangement of cameras around the recording area where gait is performed, and calibrating structure

<sup>1</sup> *PFC Sigma* is a trademark of *DePuy Orthopaedics, Inc.*

<sup>2</sup> Legs trajectories, pelvic tilt and pelvic rotation can also be obtained, but were not considered in this study.



**Fig. 2.** Placement of markers on bony landmarks, and scenario for gait analysis, as recorded from the six cameras, where the markers are clearly identified in the image

The evolution of the complete gait cycle comprises five different phases: early swing (which results in the maximum flexion of the knee joint), late swing (which is characterized by the heel gently touching the floor), weight acceptance (which occurs when the other leg is about to be lifted up and the body weight is shifted to the leg of interest), mid-stance (characterized by a straight leg with the foot flat on the floor), and terminal stance (in which the body weight shifts to the other leg and the heel is taken off the floor). In our experiments the complete cycle was analyzed for each measurement (Fig. 3), and principal components analysis was performed to reduce the dimensionality, resulting that knee flex-extension is the measurement that contributes the most to describe gait variability.

### 3 Clustering with Competitive Learning

The problem of clustering a set of  $n$ -dimensional points (patterns) consists in finding a set of points in this space (prototypes) that minimize the overall distance of the patterns to their corresponding prototypes, i.e., the lowest distance from each pattern to the prototypes. Determining the number of prototypes and their location in this space is a NP-complete problem for a dimensionality equal or greater than two. Among other algorithms, competitive neural networks [14] have been proposed to approximate solutions to the clustering problem for a fixed number of prototypes.

Competitive learning operates in a neural network by (1) selecting the closest prototype (winner weight vector) to the input pattern according to a given distance

measure, and (2) moving the winner neuron's weights to the input. The system is then defined in a formal way as a fully connected two-layers network, where the  $n$  inputs neurons ( $\mathbf{x}$ ) store the input patterns, and the  $k$  output neurons ( $\mathbf{y}$ ) raise a value according to the equation:

$$y_i = \begin{cases} 1 & \text{if } h_i = \max_j \{h_j\} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

where  $i$  ranges from 1 to  $k$ , and  $h_i$  is the synaptic potential, given by the expression:

$$h_i = \sum_{j=1}^n w_{ij}x_j - \frac{1}{2} \sum_{j=1}^n w_{ij}^2 \quad (2)$$

and  $w_{ij}$  is the synaptic weight from input neuron  $i$  to output neuron  $j$ . Considering  $\mathbf{w}_i$  as the weights vector reaching output neuron  $i$ , notice that we the term  $\mathbf{w}_i \cdot \mathbf{w}_i/2$  has been introduced in the synaptic potential to warranty that  $h_i \geq h_j \Leftrightarrow \|\mathbf{w}_i - \mathbf{x}\|^2 \leq \|\mathbf{w}_j - \mathbf{x}\|^2$ , for all  $j \neq i$ , where  $\|\cdot\|$  represents the Euclidean distance.

Then, the winner neuron will be the one closest to the input pattern,  $\mathbf{x}$ .

Competitive learning is implemented by modifying the winner neuron's weight vector according to this equation:

$$\begin{aligned} \Delta \mathbf{w}_r(k+1) &= \eta(\mathbf{x}(k) - \mathbf{w}_r(k)) \\ \Delta \mathbf{w}_i(k+1) &= 0, \quad \forall i \neq r \end{aligned} \quad (3)$$

where  $r$  is the winner neuron, and  $\eta$  is the learning rate that controls the speed of convergence to the solution.

The cumulative effect of this rule has the result of pushing the weight vectors to the centroids of the clusters, i.e. areas with high density of patterns. In the learned configuration  $\mathbf{w}_i$  will store the coordinates of the prototype for cluster  $i$ , as a new point in the  $n$ -dimensional space. The sample is then classified by assigning each pattern to the closest prototype.

As in many neural models, the main drawback of this learning rule is its sensitive dependence on initial conditions. The set of prototypes obtained strongly depends on the random initial value of the weights. Another important problem when applied to a sample of highly dimensional patterns is that the number of desired prototypes (output neurons) must be specified, even that no prior knowledge is available.

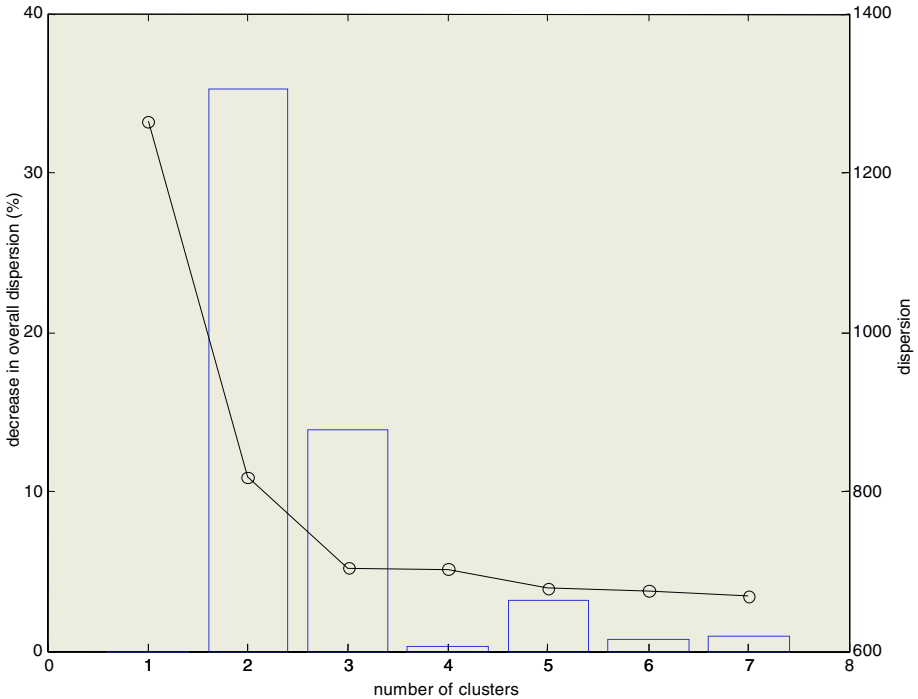
The first drawback can be avoided by performing multiple runs of the algorithm, in a way we obtain statistically good solutions. The second problem is very much application-dependent, but can also be attacked from a statistical approach. Here we propose to determine the number of groups by looking at the reduction in overall dispersion of the sample with respect to the obtained prototypes. As it can be seen in Fig. 3 this measurement defines a curve that decreases strongly at the beginning, to enter later on a steady level. We propose to determine the number of groups by thresholding this measurement. A change rate below, say, 5% would indicate that further refinement does not achieve a significant improvement, so the previous option defines a proper level that balances the ratio between overall dispersion and number of groups. The proposed method derives from an evolutionary version of the competitive algorithm that has been successfully applied to cluster images of olive pomace in an industrial set-up [15].



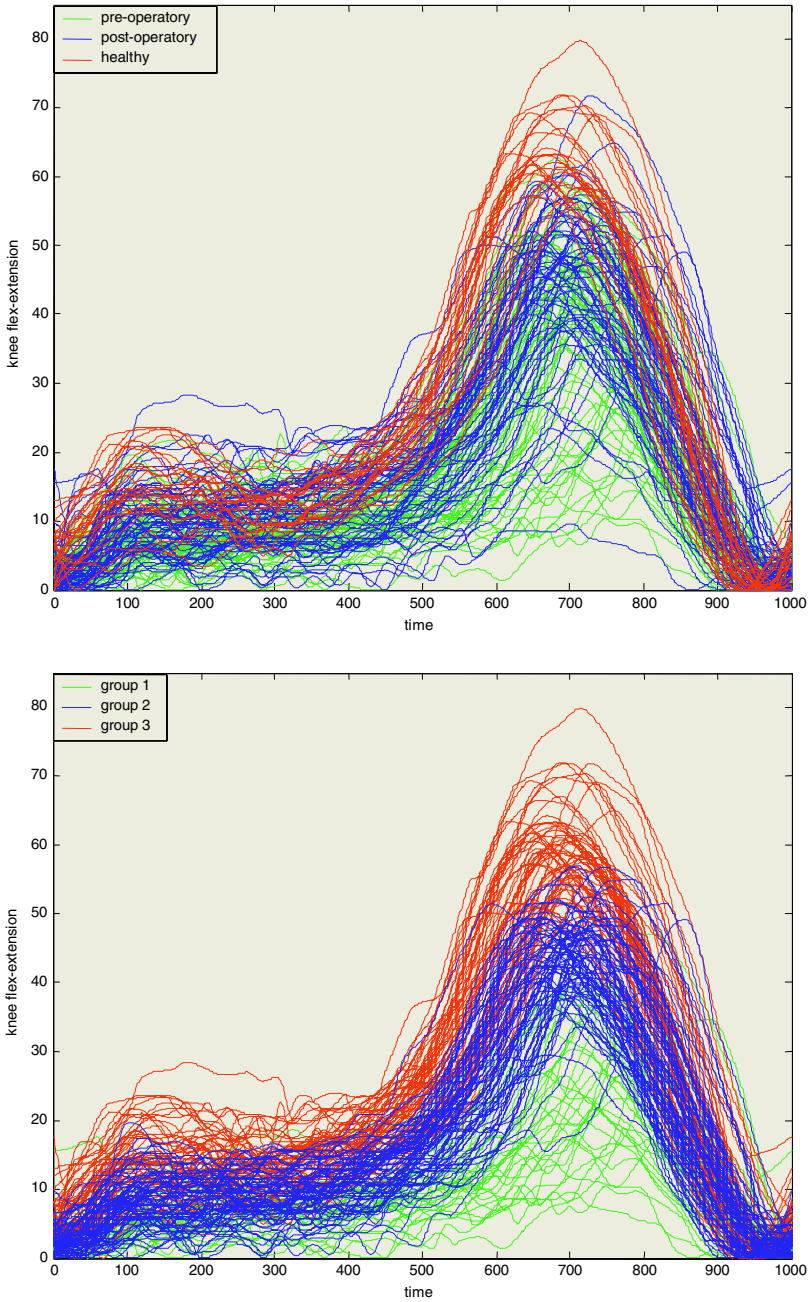
### 4 Gait Patterns Generation

The application of competitive learning to clustering the gait cycle generates the prototypes that better fit the different patterns of gait. In this experiment, knee flex-extension was computed for a complete cycle of each of the 166 patients. The curves obtained for the three classes of patients are presented in Fig. 4(a).

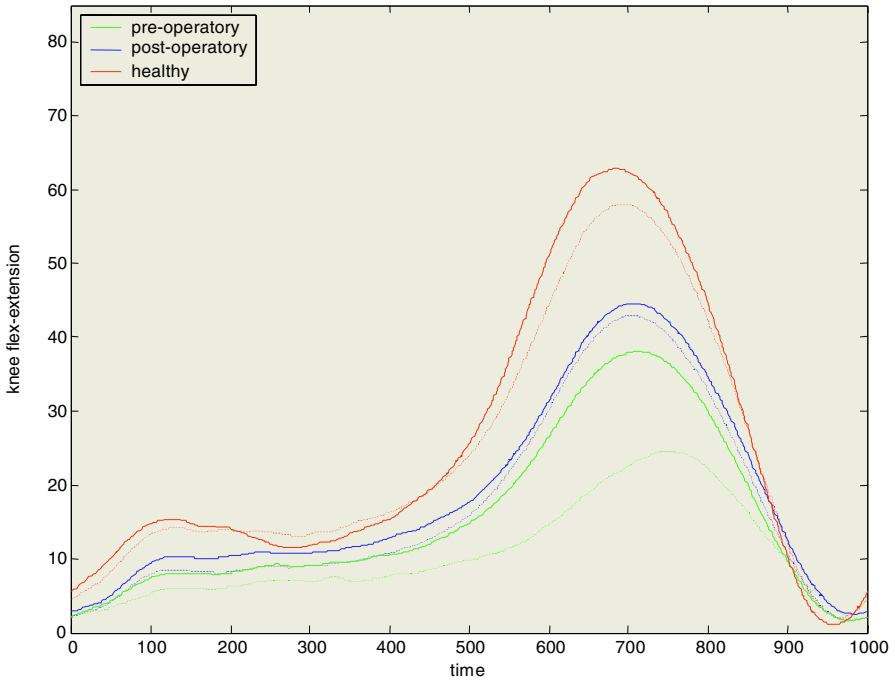
Clustering this sample of patterns with equation (3) and a variable number of clusters results in a high overall dispersion of the patterns (with respect to the closest prototype). Fig. 3 shows this dispersion graphically. We can appreciate that the reduction in overall dispersion from one to two prototypes is high, as it is when we increase to three prototypes. On the contrary, using more than three prototypes does not reduce dispersion significantly. Three is, indeed, the number of classes that we can clinically distinguish. The resulting groups of patterns are represented in Fig. 4(b), and they closely reshape the distribution in Fig. 4(a). This comparison can also be checked by matching original and obtained prototypes (Fig. 5).



**Fig. 3.** Dispersion of the patterns with respect to the closest prototype (line, right axis), and reduction of overall dispersion in percentage with respect to the previous number of classes (bars, left axis)



**Fig. 4.** Temporal evolution (1000 samples) of a complete gait cycle. (a) Original classes (pre- and post-operative, and healthy patients). (b) Result of clustering the sample into three groups



**Fig. 5.** Prototypes of original classes (dashed line), and clusters obtained (solid line)

Training the network to cluster the 166 patterns was done in real time on a conventional PIV-based computer.

Clustering with competitive neural networks is a powerful technique that generate prototypes of behaviour at a low computational cost. The application of this learning model to cluster gait patterns from 3D optical tracking measurements yields a number of gait patterns that precisely fit clinical assessment.

This clustering process demonstrates the existence of robust gait patterns, and confirms the clinically tested beneficial effects of total knee joint replacement, since the pattern of pre-operative patients moves in the direction of a healthy gait after surgery (post-operative pattern).

This method allows the dynamical computation of prototypes that can be used to assess gait, in diagnosis, as well as in patient follow-up.

## Acknowledgments

This work has been partially funded by *Johnson & Johnson Productos Profesionales, S.L.*, and projects PTR95.0734.OP (*PETRI* program) and OTRI-PAI-03-18 (*Junta de Andalucía* and OTRI of the *Universidad de Málaga*).

## References

1. Kawakami, H., Sugano, N., Yonenobu, K., Yoshikawa, H., Ochi, T., Hattori, A., Suzuki, N. Gait Analysis System for Assessment of Dynamic Loading Axis of the Knee. *Gait & Posture* **21** (2005) 125-130.
2. Ghousayni, S., Stevens, C., Durham, S., Ewins, D.: Assessment and Validation of a Simple Automated Method for the Detection of Gait Events and Intervals. *Gait & Posture* **20** (2004) 266-272
3. Hallemans, A., De Clercq, D., Otten, B., Aerts, P.: 3D Joint Dynamics of Walking in Toddlers: A Cross-Sectional Study Spanning the First Rapid Development Phase of Walking. *Gait & Posture*. In press.
4. Kim, C.M., Eng, J.J.: Magnitude and Pattern of 3D Kinematic and Kinetic Gait Profiles in Persons with Stroke: Relationship to Walking Speed. *Gait & Posture* **20** (2004) 140-146
5. Manal, K., Stanhope, S.J.: A Novel Method for Displaying Gait and Clinical Movement Analysis Data. *Gait & Posture* **20** (2004) 222-226
6. Alkjaer, T., Simonsen, E.B., Dyhre-Poulsen, K.: Comparison of Inverse Dynamics Calculated by Two- and Three-Dimensional Models During Walking. *Gait & Posture* **13** (2001) 73-77
7. Miller, C., Mulavara, A., Bloomberg, J.: A Quasi-Static Method for Determining the Characteristics of a Motion Capture Camera System in a "Split-Volume" Configuration. *Gait & Posture* **16** (2002) 283-287
8. Sadeghi, H., Allard, P., Shafie, K., Mathieu, P.A., Sadeghi, S., Prince, F., Ramsay, J.: Reduction of Gait Data Variability Using Curve Registration. *Gait & Posture* **12** (2000) 257-264
9. Churchill, A.J.G., Halligan P.W., Wade, T.W.: RIVCAM: A Simple Video-Based Kinematic Analysis for Clinical Disorders of Gait. *Computer Methods and Programs in Biomedicine* **69**(3) (2001) 197-209
10. Holzreiter, S.H., Köhle, M.E.: Assessment of Gait Patterns Using Neural Networks. *J Biomech.* **26**(6) (1993) 645-651
11. Köhle, M.E., Merkl, D.: Identification of gait patterns with self-organizing maps based on ground reaction force. In: Proc. of the European Symposium on Artificial Neural Networks (ESANN 96), Bruges, Belgium, April (1996) 24-26
12. Köhle, M.E., & Merkl, D. Analyzing human gait patterns for malfunction detection. In: Proceedings of the 2000 ACM symposium on Applied computing, Como, Italy (2000) 41-45
13. Bouguet, J. Y.: Camera Calibration Toolbox for Matlab. (2004)  
[http://www.vision.caltech.edu/bouguetj/calib\\_doc/](http://www.vision.caltech.edu/bouguetj/calib_doc/)
14. Ahalt, S. C., Krishnamurthy, A. K., Chen, P., Melton, D. E.: Competitive learning algorithms for vector quantization. *Neural Networks* **3**(3) (1990) 277-290
15. Sánchez, A. M., Vico, F. J., Veredas, F. J., Cabello, S., López, I.: Evolutionary competitive learning for automatic detection of oil leaks in olive pomace. Technical Report # 05/04. Departamento de Lenguajes y Ciencias de la Computación, Universidad de Málaga, Spain (2005).

# Knowledge Extraction from Neural Networks Using the All-Permutations Fuzzy Rule Base: The LED Display Recognition Problem

Eyal Kolman and Michael Margaliot\*

School of Electrical Engineering, Tel-Aviv University, Tel-Aviv, Israel 69978

**Abstract.** A major drawback of artificial neural networks is their black-box character. In this paper, we use the equivalence between artificial neural networks and a specific fuzzy rule base to extract the knowledge embedded in the network. We demonstrate this using a benchmark problem: the recognition of digits produced by a LED device. The method provides a symbolic and comprehensible description of the knowledge learned by the network during its training.

## 1 Introduction

The ability of artificial neural networks (ANNs) to learn and generalize from examples makes them very suitable for use in numerous real-world applications where exact algorithmic approaches are unknown or too difficult to implement. The knowledge learned during the training process is distributed in the weights of the different neurons and it is very difficult to comprehend exactly what the ANN is computing. The problem of representing the knowledge learned by the network in a comprehensible form received a great deal of attention in the literature (see, e.g., [1, 2, 3]).

Rule based systems process information in a manner that is much easier to comprehend because the system's knowledge is stated using symbolic If-Then rules. In particular, fuzzy rule bases (FRBs) enable the use and manipulation of expert knowledge stated using *natural language* [4, 5, 6]. Thus, the knowledge is easy to understand, verify, and, if necessary, refine.

Recently, a great deal of research has been devoted to designing hybrid intelligent systems that fuse subsymbolic and symbolic techniques for information processing [7] and, in particular, to creating a synergy between ANNs and FRBs [8]. Such a synergy may lead to systems with the robustness and learning capabilities of ANNs and the "white-box" character of FRBs.

A well-known neuro-fuzzy model is the Adaptive Network-Based Fuzzy Inference System (ANFIS) developed by Jang *et al.* [9], which is a feedforward network representation of the fuzzy reasoning process. A similar mapping of a

---

\* Corresponding author: Dr. Michael Margaliot. Tel: +972-3-6407768; Fax: +972-3-640 5027; Homepage: [www.eng.tau.ac.il/~michaelm](http://www.eng.tau.ac.il/~michaelm); Email: [michaelm@eng.tau.ac.il](mailto:michaelm@eng.tau.ac.il)

rule base into an ANN was suggested by Fu and Fu [10]. However, these approaches cannot be used to extract knowledge from a standard ANN.

Jang and Sun [11] showed that radial basis function networks (RBFNs) are mathematically equivalent to FRBs with Gaussian membership functions. However, this equivalence holds only for the special case of RBFNs.

Benitez *et al.* [12] showed that ANNs with Logistic activation functions are equivalent to the result of inferencing a set of Mamdani-type fuzzy rules (see also [13, 14]). However, this is not a standard FRB, as the operators used in the inferencing method are not those commonly used in FRBs.

Recently, the authors introduced a new Mamdani-type FRB referred to as the All-Permutations Fuzzy Rule Base (APFRB) [15, 16]. Inferencing the APFRB, using *standard* tools from fuzzy logic theory, yields an input-output relationship that is mathematically equivalent to that of a feed-forward ANN. More precisely, there exists an invertible transformation  $T$  such that

$$T(\text{ANN}) = \text{APFRB} \text{ and } T^{-1}(\text{APFRB}) = \text{ANN}. \quad (1)$$

This equivalence enables bidirectional information flow between the ANN and the corresponding APFRB. It also enables application of tools from the theory of ANNs to APFRBs and vice versa.

In this paper, we use the equivalence  $T(\text{ANN}) = \text{APFRB}$  to extract knowledge from an ANN in the form of symbolic rules. We demonstrate this approach on a benchmark problem involving the recognition of the digits displayed by a LED device. An ANN is trained to recognize the ten possible digits. Calculating  $\text{APFRB} = T(\text{ANN})$ , and simplifying the rules, yields a simple symbolic description of the ANN's knowledge. The final result is an FRB with ten rules that correctly classifies all the training examples. Furthermore, this FRB is tractable, and provides a comprehensible representation of the ANN's functioning. For example, it is possible to deduce that the ANN learned to focus its efforts on the digits that are harder to recognize.

The rest of the paper is organized as follows. In section 2, we review the APFRB and its equivalence to an ANN. In Section 3, we present the benchmark problem and the ANN trained to solve it. In Section 4, we apply the equivalence (1) to extract information from the trained ANN. In Section 5, we show that the FRB is tractable and allows us to represent the ANN's functioning in a comprehensible form. The final section concludes. Due to space limitations, some of the technical details have been omitted. They can be found in an extended version of this paper [17].

## 2 All-Permutations Fuzzy Rule-Base

We briefly review the APFRB and its equivalence to an ANN (more details and the proofs can be found in [15]). For the sake of simplicity, we consider the case of an FRB with output  $\mathbf{f} \in \mathbb{R}^2$ ; the generalization to the case  $\mathbf{f} \in \mathbb{R}^n$  is straightforward.

**Definition 1 (APFRB).** A fuzzy rule base with input  $\mathbf{x} \in \mathbb{R}^m$  and output  $\mathbf{f} \in \mathbb{R}^2$  is called an APFRB if the following conditions hold.

1. Every input variable  $x_i$  is characterized by two linguistic terms:  $\text{term}_-^i$  and  $\text{term}_+^i$ . The membership functions  $\mu_-^i(\cdot)$  and  $\mu_+^i(\cdot)$  that model these terms satisfy the following constraint: there exists a  $v_i \in \mathbb{R}$  such that

$$(\mu_+^i(y) - \mu_-^i(y))/(\mu_+^i(y) + \mu_-^i(y)) = \tanh(y - v_i), \quad \forall y \in \mathbb{R}. \quad (2)$$

2. The form of every rule is

$$\text{If } x_1 \text{ is } \text{term}_{+/-}^1 \text{ and } \dots \text{ and } x_m \text{ is } \text{term}_{+/-}^m \text{ Then } \mathbf{f} = \begin{pmatrix} a_0 \pm a_1 \cdots \pm a_m \\ b_0 \pm b_1 \cdots \pm b_m \end{pmatrix} \quad (3)$$

where  $\text{term}_{+/-}^i$  stands for either  $\text{term}_+^i$  or  $\text{term}_-^i$ ,  $\pm$  stands for either the plus or the minus sign, and  $a_i, b_i \in \mathbb{R}$ . The actual signs in the Then-part are determined in the following manner: if the term characterizing  $x_i$  in the If-part is  $\text{term}_+^i$ , then in the Then-part,  $a_i$  and  $b_i$  appear with a plus sign; otherwise  $a_i$  and  $b_i$  appear with a minus sign.

3. The rule base contains exactly  $2^m$  rules spanning, in their If-part, all the possible assignment combinations of  $x_1, \dots, x_m$ .

Several commonly used fuzzy membership functions satisfy the constraint (2). For example, the pair of Gaussian membership functions

$$\mu_{=k}(y) := \exp(-(y - k)^2/(2k)) \quad \text{and} \quad \mu_{=-k}(y) := \exp(-(y + k)^2/(2k)), \quad (4)$$

satisfy (2) with  $v = 0$ . The sigmoid functions

$$\mu_{>k}(y) := (1 + \exp(-2(y - k)))^{-1} \quad \text{and} \quad \mu_{<k}(y) := (1 + \exp(2(y - k)))^{-1}, \quad (5)$$

satisfy (2) with  $v = k$ . Thus, the linguistic terms *equals k* and *equals -k* (modeled by Gaussians) and *larger than k* and *smaller than k* (modeled by sigmoids) can be used in an APFRB.

Summarizing, an APFRB with input  $\mathbf{x} \in \mathbb{R}^m$  and output  $\mathbf{f} \in \mathbb{R}^2$  is defined using  $2^m$  rules and  $3m + 2$  parameters: the  $2(m + 1)$   $a_i$ s and  $b_i$ s in (3), and the  $m$   $v_i$ s in (2) (in the general case where  $\mathbf{f} \in \mathbb{R}^n$ , the number of parameters is  $n(m + 1) + m$ ). The next result shows that the APFRB is an ANN in disguise.

**Theorem 1.** Applying the product-inference rule, singleton fuzzifier, and the center of gravity defuzzifier to an APFRB yields

$$\mathbf{f} = \begin{pmatrix} a_0 + \sum_{i=1}^m a_i \tanh(x_i - v_i) \\ b_0 + \sum_{i=1}^m b_i \tanh(x_i - v_i) \end{pmatrix}. \quad (6)$$

PROOF. See [15].

In other words, each output  $f_i$  of the APFRB can be obtained by first feeding the inputs  $x_i$  to a hidden layer computing the activation functions  $\tanh(x_i - v_i)$ , and then computing a weighted sum of the hidden layer's outputs.

Note that if we choose  $a_0 = \sum_{i=1}^m a_i$  and  $b_0 = \sum_{i=1}^m b_i$ , then (6) becomes

$$\mathbf{f} = \left( \frac{\sum_{i=1}^m 2a_i / (1 + \exp(-2(x_i - v_i)))}{\sum_{i=1}^m 2b_i / (1 + \exp(-2(x_i - v_i)))} \right). \tag{7}$$

The hyperbolic tangent functions in (6) and the Logistic functions in (7) are standard activation functions in numerous ANNs. Thus, the APFRB’s outputs  $\mathbf{f}$  can be represented as the output of a standard ANN.

Conversely, consider an ANN with input  $\mathbf{z} \in \mathbb{R}^n$ , a single hidden layer with  $m$  units, and two output units. Its output  $\mathbf{f} \in \mathbb{R}^2$  is given by

$$\mathbf{f} = \left( \frac{\sum_{j=1}^m c_j h(y_j + \theta_j)}{\sum_{j=1}^m d_j h(y_j + \theta_j)} \right), \tag{8}$$

where  $y_j := \sum_{i=1}^n w_{ji} z_i$  is the input to the  $j$ th neuron in the hidden layer,  $\theta_j$  is the bias of this neuron, and  $c_j$  ( $d_j$ ) is the weight from this neuron to the first (second) output neuron. Comparing (8) with (6) yields the following.

**Corollary 1.** *If the activation function in the ANN is  $h(z) = \tanh(z)$ , then (8) is the output of an APFRB with:  $a_0 = 0$ ,  $b_0 = 0$ ,  $a_i = c_i$ ,  $b_i = d_i$ ,  $v_i = -\theta_i$ ,  $x_i = y_i$ ,  $i = 1, \dots, m$ .*

*If  $h(z) = 1/(1 + \exp(-z))$ , then (8) is the output of an APFRB with:  $a_0 = \sum_{i=1}^m a_i$ ,  $b_0 = \sum_{i=1}^m b_i$ ,  $a_i = c_i/2$ ,  $b_i = d_i/2$ ,  $v_i = -\theta_i/2$ , and  $x_i = y_i/2$ ,  $i = 1, \dots, m$ .*

Summarizing, Theorem 1 establishes an equivalence between a single<sup>1</sup> hidden layer ANN and an APFRB, and explicitly defines the transformation  $T$  in (1).

Corollary 1 implies that we can immediately extract the knowledge embedded in an ANN in the form of fuzzy If-Then rules. We demonstrate the usefulness of this approach using a benchmark problem.

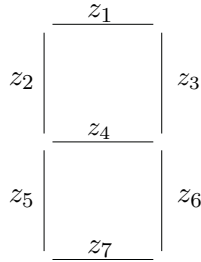
### 3 The LED Display Recognition Problem

The LED display recognition problem [18] concerns learning to recognize digits displayed using a seven-segment light emitting diodes (LED) display. Several pattern recognition algorithms were applied to this problem including classification trees [18], instance-based learning algorithms [19], and ANNs [20].

The database contains 2050 supervised examples in the form  $(z_1, z_2, \dots, z_{24}, v)$ . The first seven inputs,  $z_1 \dots z_7$ , are the diodes’ states (1 for on, 0 for off) (see Fig. 1). For example, the vector  $\{1, 1, 0, 1, 1, 1, 1\}$  represents the digit 6, and  $\{1, 1, 1, 1, 0, 1, 1\}$  the digit 9. The value  $v \in \{0, 1, \dots, 9\}$  is the displayed digit. The inputs  $z_8 \dots z_{24}$  are independent random variables with  $\text{prob}(0) = \text{prob}(1) = 1/2$ . These noise inputs make the recognition task more challenging as it becomes necessary to discriminate between meaningful and useless inputs.

<sup>1</sup> The equivalence is easily generalized to ANNs with multiple hidden layers; see [15].





**Fig. 1.** The LED device

We trained a 24-6-10 ANN using the backpropagation algorithm. The number of hidden units was determined using a trial and error approach; a network with less than 6 hidden units could not be trained to correctly classify all the examples<sup>2</sup>. Each of the ten outputs  $f_0, \dots, f_9$  corresponds to a different digit and the final classification is based on the winner-takes-all approach. That is, the ANN's classification is digit  $i$ , where  $i = \arg \max_{0 \leq k \leq 9} \{f_k\}$ . After training, the ANN correctly classified the database. The ANN's parameters (204 weights and 16 biases), however, provide no insight on its functioning.

## 4 Knowledge Extraction

Let  $x_j := \sum_{i=1}^{24} w_{ji} z_i$ ,  $j = 1, \dots, 6$ , denote the input of the ANN's  $j$ th hidden neuron. We applied Corollary 1 to represent the ANN as an APFRB with  $2^6 = 64$  fuzzy rules and an output  $\mathbf{f} \in R^{10}$ . For example, one of the rules is:<sup>3</sup>

**If**  $x_1$  equals 1 and  $x_2$  equals 1 and  $x_3$  equals  $-1$  and  $x_4$  equals 1 and  $x_5$  equals  $-1$  and  $x_6$  equals 1 **Then**  $\mathbf{f} = (-1.4, -2.5, -0.6, -0.7, -0.2, -0.5, -11, -1.4, 0, 0.4)^T$ . The membership functions defining the terms *equals*  $\pm 1$  are the Gaussian functions in (4).

The inferencing amounts to computing a weighted sum,  $\mathbf{f}$ , of the sixty-four vectors in the Then-part of the rules, and the final digit classification is  $i := \arg \max_{0 \leq k \leq 9} \{f_k\}$ , where  $f_k$  is the  $k$ th entry in  $\mathbf{f}$ .

This rule base provides a symbolic representation of the ANN's functioning. In particular, we have a fuzzy classifier that solves the LED recognition problem. However, the comprehensibility of this classifier is hindered by the large number and the complexity of the rules. To gain more insight, we applied a rule reduction and simplification technique for the APFRB (due to lack of space, the technique is not presented here; see [15] for more details). This reduces the number of rules from 64 to 10. Furthermore, the resulting FRB is simple enough to allow us to interpret its functioning.

<sup>2</sup> A similar network was used in [20].

<sup>3</sup> The numerical values were rounded to one decimal digit, without affecting the classification accuracy.

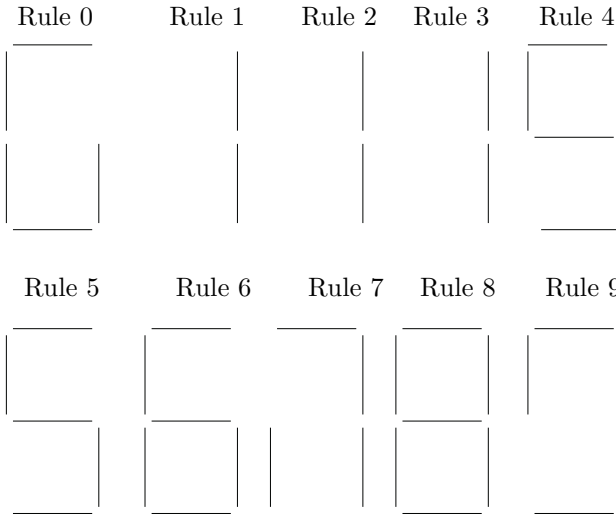


Fig. 2. The pattern yielding maximal DOF for each rule

## 5 Interpreting the FRB

The symbolic structure of FRBs makes them much easier to understand than ANNs. In particular, we can analyze the operation of an FRB by understanding the If-part and the Then-part of each rule.

**The If-part:** Consider the degree of firing (DOF) of the ten rules for each possible input (namely, the  $2^7 = 128$  possible binary vectors  $(z_1, \dots, z_7)$ ). The ratio of the highest DOF and the second highest DOF for the ten rules is: 9.3, 12.7, 3.4, 1.5, 5.4, 5.4, 4.5, 2.3, 19.4 and 2.4. Thus, with the exception of the fourth rule, every rule is tuned to a single specific input pattern and yields a much smaller DOF for any other pattern.

Figure 2 depicts the pattern yielding the highest DOF for each rule. It may be seen that rules 1, 5, 6 and 8 are tuned to recognize the digits 1, 5, 6 and 8, respectively. Rules 0 and 7 are tuned to patterns that are one Hamming distance away from the real digits 0 and 7.

Comparing the DOF for the ten patterns representing the digits  $0, \dots, 9$  only, we find that rules 2 and 3 have the highest DOF when the input is the digit one, and rule 4 has the highest DOF when the input is the digit five. For all other rules, rule  $i$  yields the highest DOF when the input is digit  $i$ .

**The Then-part:** Considering the output vectors  $\mathbf{f}^i$ ,  $i = 0, \dots, 9$ , and letting  $(\mathbf{f}^i)_k$  denote the  $k$ th entry of the vector  $\mathbf{f}^i$ , we find that  $\arg \max_k (\mathbf{f}^i)_k = i$  for all  $i$ . In other words, if only rule  $i$  fired, then the inferencing would yield digit  $i$ . In most rules, there is a considerable difference between entry  $i$  and the second largest entry in  $\mathbf{f}^i$ . In five of the ten rules, the largest entry is positive and the other nine entries are negative. Thus, when such a rule fires it not only

contributes to the classification towards a specific digit, but also contributes negatively to all other possible classifications.

Summarizing, we see that the FRB includes seven rules that are tuned to a specific digit. These are rules 0, 1, 5, 6, 7, 8, and 9. Each of these rules responds with a high DOF when the input is the appropriate digit.

On the other hand, rules 2, 3 and 4 are not tuned to the corresponding digit. For example, rule 2 displays the highest DOF when the input is the digit 1. The fact that the rule base correctly classifies digits 2, 3 and 4 is due to the weighted combination of all the rules' outputs, and not to the specific action of a single rule.

This behavior motivated us to try to reach an understanding as to the distinction between the two sets of digits

$$S_1 := \{0, 1, 5, 6, 7, 8, 9\} \text{ and } S_2 := \{2, 3, 4\}. \quad (9)$$

Let  $H(d_1, d_2)$  denote the Hamming distance between the LED representations of the digits  $d_1$  and  $d_2$  (e.g.,  $H(1, 7) = 1$ ). Let  $M_i$  denote the set of digits  $d$  that satisfy  $\min_{0 \leq j \leq 9} \{H(d, j)\} = i$  (i.e., the digit closest to digit  $d$  is at distance  $i$ ). Then,

$$M_1 = \{0, 1, 3, 5, 6, 7, 8, 9\} \text{ and } M_2 = \{2, 4\}. \quad (10)$$

The definition of  $M_i$  suggests that the digits in the set  $M_1$  are more difficult to recognize than those in the set  $M_2$ .

Comparing (9) with (10), we see that there is a high correspondence between  $M_i$  and  $S_i$ . Thus, the FRB (or the original ANN) dedicates specially tuned rules for the more "tricky" digits.

The notion that digits that are more difficult to recognize deserve more attention is quite intuitive. However, understanding that the ANN implements this notion by observing its weights and biases is all but impossible. It is only through the knowledge extraction process that this notion emerges.

## 6 Conclusions

The output of a feed-forward ANN can be represented as the result of inferring a fuzzy rule base with a special structure—the APFRB. This equivalence allows the bi-directional flow of information between the subsymbolic knowledge representation in the ANN and the symbolic rules of the APFRB.

In this paper, we studied one application of this equivalence. The transformation  $\text{APFRB} = T(\text{ANN})$  extracts the knowledge from a trained ANN in the form of symbolic fuzzy rules. We demonstrated this approach using a medium-size ANN trained to solve a benchmark problem. The 24-6-10 network was transformed into a set of 64 fuzzy rules. Simplification of this rule base yields a comprehensible representation of the ANN's functioning. In particular, it is possible to conclude that the ANN dedicates special rules to digits that are more difficult to recognize.

## References

1. Cloete, I., Zurada, J.M., eds.: Knowledge-Based Neurocomputing. MIT Press (2000)
2. Andrews, R., Diederich, J., Tickle, A.: Survey and critique of techniques for extracting rules from trained artificial neural networks. *Knowledge-Based Systems* **8** (1995) 373–389
3. Tickle, A., Andrews, R., Golea, M., Diederich, J.: The truth will come to light: directions and challenges in extracting the knowledge embedded within trained artificial neural networks. *IEEE Trans. Neural Networks* **9** (1998) 1057–1068
4. Tron, E., Margaliot, M.: Mathematical modeling of observed natural behavior: a fuzzy logic approach. *Fuzzy Sets Systems* **146** (2004) 437–450
5. Dubois, D., Nguyen, H.T., Prade, H., Sugeno, M.: Introduction: The real contribution of fuzzy systems. In Nguyen, H.T., Sugeno, M., eds.: *Fuzzy Systems: Modeling and Control*. Kluwer (1998) 1–17
6. Zadeh, L.A.: Fuzzy logic = computing with words. *IEEE Trans. Fuzzy Systems* **4** (1996) 103–111
7. McGarry, K., Wermter, S., MacIntyre, J.: Hybrid neural systems: from simple coupling to fully integrated neural networks. *Neural Computing Surveys* **2** (1999) 62–93
8. Mitra, S., Hayashi, Y.: Neuro-fuzzy rule generation: survey in soft computing framework. *IEEE Trans. Neural Networks* **11** (2000) 748–768
9. Jang, J.S.R., Sun, C.T., Mizutani, E.: *Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*. Prentice-Hall (1997)
10. Fu, L.M., Fu, L.C.: Mapping rule based systems into neural architectures. *Knowledge Based Systems* **3** (1990) 48–56
11. Jang, J.S.R., Sun, C.T.: Functional equivalence between radial basis function networks and fuzzy inference systems. *IEEE Trans. Neural Networks* **4** (1993) 156–159
12. Benitez, J.M., Castro, J.L., Requena, I.: Are artificial neural networks black boxes? *IEEE Trans. Neural Networks* **8** (1997) 1156–1164
13. Castro, J.L., Mantas, C.J., Benitez, J.M.: Interpretation of artificial neural networks by means of fuzzy rules. *IEEE Trans. Neural Networks* **13** (2002) 101–116
14. Zhang, D., Bai, X.L., Cai, K.Y.: Extended neuro-fuzzy models of multilayer perceptrons. *Fuzzy Sets Systems* **142** (2004) 221–242
15. Kolman, E. and Margaliot, M.: Are artificial neural networks white boxes? *IEEE Trans. Neural Networks*, to appear. [Online]. Available: [www.eng.tau.ac.il/~michaelm](http://www.eng.tau.ac.il/~michaelm)
16. Kolman, E., Margaliot, M.: Neural networks = fuzzy rule bases. In Ruan, D. *et al.* eds.: *Applied Computational Intelligence – Proceedings of the 6th International FLINS Conference*. World Scientific (2004) 111–117
17. Kolman, E., Margaliot, M.: Knowledge extraction from neural networks using the all-permutations fuzzy rule base, submitted. [Online]. Available: [www.eng.tau.ac.il/~michaelm](http://www.eng.tau.ac.il/~michaelm)
18. Breiman, L., Friedman, J., Olshen, R., Stone, C.: *Classification and Regression Trees*. Wadsworth International Group (1984), ch. 2
19. Aha, D.W., Kibler, D., Albert, M.K.: Instance-based learning algorithms. *Machine Learning* **6** (1991) 37–66
20. Boger, Z., Guterman, H.: Knowledge extraction from artificial neural networks models. In: *Proc. IEEE Int. Conf. Systems, Man and Cybernetics (SMC97)*, Orlando, Florida (1997) 3030–3035

# Controlling Force Based on Radial Fuzzy Functions in High-Speed Machining Processes

Rodolfo Haber-Guerra<sup>1,2,\*</sup>, Rodolfo Haber-Haber<sup>3</sup>, and José R. Alique<sup>1</sup>

<sup>1</sup>Instituto de Automática Industrial (CSIC),  
km. 22,800 N-III, La Poveda, 28500 Madrid, Spain  
{rhaber, jralique}@iai.csic.es

<sup>2</sup>Escuela Politécnica Superior, Ciudad Universitaria de Cantoblanco,  
Calle Francisco Tomás y Valiente, 11  
28049 – Madrid, Spain

Rodolfo.Haber@ii.uam.es

<sup>3</sup>Departamento de Control Automático, Universidad de Oriente, 90400, Cuba  
rhaber@rhh.uo.edu.cu

**Abstract.** This paper addresses the development of a new control strategy to regulate cutting force in a high-speed machining process. Fuzzy basis functions (FBF), on the basis of L.X.Wang's approach, serve as basement for designing and implementing adaptive fuzzy control system in an open computerized numerical control (CNC). The controller uses cutting force measured from a dynamometric platform, and mathematically processed by means of an integrated application, to perform real-time modification of feed rate. The integration process, design steps and results of applying the adaptive fuzzy-control system in actual high-speed machining operations corroborate the suitability of the proposed control strategy for real-time applications. Moreover, the results show a good transient response in the cutting force pattern despite the complexity of the mechanized part.

**Keywords:** Fuzzy control, Radial Fuzzy Functions, high-speed machining processes.

## 1 Introduction

Manufactured products are mainly made by machine tools that perform the basic functions such as removing material, joining material, and assembling discrete parts into products. The role of the machine tool in today's integrated manufacturing environment is changing dramatically, driven by the necessity of achieving productivity improvements year-after-year to remain cost competitive against a global market. Based on the past advancements with computerized numerical control (CNC), PC-based controller platforms, and innovative sensing, the machine tool's utilization rate and precision has risen while failures, problems, and downtime have fallen.

---

\* Corresponding author.

The newer machine tools are increasingly able to interact electronically with their surroundings on the basis of open CNCs. However there is a large gap of needed technologies and modelled intelligence to capture the next significant level of productivity and cost benefit. Continuous increases are required in the machine utilization rate, precision, and autonomic operation.

Nowadays most machine tools are equipped with open CNC. In CNC systems, machining parameters are usually selected according to operators experience, handbooks or the result of a post-processing stage from CAD/CAM systems. Therefore the machining parameters selected are usually conservative and off-line optimization to produce optimized cutting conditions can not deal with process variability (e.g., tool wear), complexity and uncertainty.

In order to increase the machining efficiency it is necessary to modify in real time machining parameters. Adaptive control of relevant variables such as cutting force, torque or vibration signals is one of the most effective methods. Nowadays, the research effort in machining force control continues due to the economic profits that this technology can provide. Recent studies reveal that model-based force controllers perform well in a laboratory environment, however up to date, many control systems are not applicable for all production environments [1,2].

The complexity and uncertainty of processes like the high-speed machining processes are what make the realm known as *intelligent systems technology* a feasible option to classical control strategies. Indeed, artificial-intelligence techniques have roused considerable interest in the scientific community and have been applied to machining [3,4,5]. The current study is focused on the design and implementation of an adaptive fuzzy controller in an open CNC system in order to improve machining efficiency. From all available techniques, an approach inspired in the adaptive fuzzy control scheme proposed in [6,7] is selected due to its viability of real time implementation, and the possibility of guaranteeing asymptotic stability. To the best of our knowledge, the main advantage of the herein suggested approach are: (i) adaptive fuzzy controller in an open CNC to deal with a real industrial process, (ii) a simple computational procedure to fulfil the time requirements, (iii) a few expert knowledge required, and (iv) easy implementation and synchronization with CNC. The results with adaptive fuzzy control strategy through industrial tests show a higher machining efficiency.

This paper is organized as follows. In Section 2 we present a brief study of the machining process, explaining why it is considered a complex process and setting up the milling process as a case study; in Section 3 we design the adaptive fuzzy controller to optimize a high speed milling (HSM) process. In Section 4 we describe how the adaptive fuzzy controller can be embedded in open CNCs. Finally, in Section 5 we show the experimental results.

## 2 High-Speed Milling Process

From the whole range of operations involved in high-speed machining [8], a high-speed milling process was chosen for this case study. This choice obeys a pessimistic criterion, for high-speed milling is one of the most complex machining operations there is. From the viewpoint of tool-wear monitoring, some of the most significant

variables and parameters involved in the HSM process are the spatial position of the cutting tool (on Cartesian coordinate axes), spindle speed  $sp$  [rpm], relative feed speed between tool and worktable (feed rate)  $f$  [mm/min], cutting speed  $V_c$  [m/min], cutting power invested in removing metal chips from the workpiece  $P_c$  [kW], cutting force exerted during the removal of metal chips  $F$  [N], radial depth of cut  $DoC$  [mm] and cutting-tool diameter  $d$  [mm]. In order to evaluate system performance, we need to select certain suitable performance indices. This work deals essentially with rough milling, so the main index is the metal-removal rate (MRR).

In the world of machining, cutting force is considered to be the variable that best describes the cutting process. Recent studies reveal the relevance of cutting-force signatures for tool-wear monitoring in high-speed machining processes. The mean and peak values of cutting force exhibit the best performance for tool-condition monitoring [9].

### 3 Fuzzy Logic Controller for HSM Processes

The design and mathematical background of the adaptive fuzzy control system to regulate the cutting force  $F$  are briefly described in this section. Details about FBF in control systems are well-described and analysed in [6], including stability analysis. The design can be summarized as follows.

The adaptive controller has two inputs: the cutting force error ( $\Delta F=x_1$ ) and the change in cutting force error ( $\Delta^2 F=x_2$ ). Gaussian membership functions are considered for input variables, defined as:

$$\mu_{A_i^l}(x_i) = \exp\left[-\frac{1}{2}\left(\frac{x_i - \bar{x}_i^l}{\sigma_i^l}\right)^2\right] \quad (1)$$

where  $x_i$  is the value of the  $i$ -th input variable,  $A_i^l$ ,  $\bar{x}_i^l$ ,  $\sigma_i^l$  are the label of the membership function, centre and the shape, respectively of the  $l$ -th fuzzy membership function,  $l=1..m_i$  is the number of the fuzzy intervals.

The manipulated (action) variable we selected is the feed-rate increment ( $\Delta f$  in percentage of the initial value programmed into the CNC). Singletons are considered for output membership functions. The feed per tooth is considered constant and preset by the operator.

The controller performs on-line actions to control the feed rate. We consider a set of rules consisting of linguistic statements that link each antecedent with its respective consequent:

$$\text{If } \Delta F \text{ is } A_1^l \text{ and } \Delta^2 F \text{ is } A_2^l \text{ THEN } f \text{ is } z^j \quad (2)$$

where initial values for  $z^j$  can be selected on the basis of expert criteria.

The *Product* compositional operator was selected for the compositional rule of inference. The crisp controller output, which is used to change the machine feedrate, was obtained by defuzzification employing the center-average defuzzifier defined as

$$f = \frac{\sum_{l_{\Delta F}=1}^{m_{\Delta F}} \sum_{l_{\Delta^2 F}=1}^{m_{\Delta^2 F}} z^{l_{\Delta F} + l_{\Delta^2 F}} \left( \mu_{A_{\Delta F}^{l_{\Delta F}}} (x_{\Delta F}) \cdot \mu_{A_{\Delta^2 F}^{l_{\Delta^2 F}}} (x_{\Delta^2 F}) \right)}{\sum_{l_{\Delta F}=1}^{m_{\Delta F}} \sum_{l_{\Delta^2 F}=1}^{m_{\Delta^2 F}} \left( \mu_{A_{\Delta F}^{l_{\Delta F}}} (x_{\Delta F}) \cdot \mu_{A_{\Delta^2 F}^{l_{\Delta^2 F}}} (x_{\Delta^2 F}) \right)} \tag{3}$$

The crisp control action (generated at each sampling instant) defines the final actions that will be applied to the CNC set points. From (3), Fuzzy Basis Functions FBF's can be defined as:

$$\xi^{(l_{\Delta F}, l_{\Delta^2 F})} (\Delta F, \Delta^2 F) = \frac{\mu_{A_{\Delta F}^{l_{\Delta F}}} (x_{\Delta F}) \cdot \mu_{A_{\Delta^2 F}^{l_{\Delta^2 F}}} (x_{\Delta^2 F})}{\sum_{l_{\Delta F}=1}^{m_{\Delta F}} \sum_{l_{\Delta^2 F}=1}^{m_{\Delta^2 F}} \left( \mu_{A_{\Delta F}^{l_{\Delta F}}} (x_{\Delta F}) \cdot \mu_{A_{\Delta^2 F}^{l_{\Delta^2 F}}} (x_{\Delta^2 F}) \right)} \tag{4}$$

and the control action

$$f \left[ \left[ \Delta F, \Delta^2 F \right] \right] = \underline{\theta}^T \underline{\xi} (\Delta F, \Delta^2 F) \tag{5}$$

where  $\underline{\theta}^T = \left[ z^1, \dots, z^{l_{\Delta F} + l_{\Delta^2 F}} \right]$  is the vector modified according to the following adaptive control law. Initial fuzzy basis functions are automatically generated for all combinations. The vector  $\underline{\theta}$  is calculated using the adaptive law:

$$\dot{\underline{\theta}} = \gamma \underline{e}^T \underline{p}_n \underline{\xi}(\underline{x}) \text{ if } (|\underline{\theta}| < M_\theta) \text{ or } ((|\underline{\theta}| = M_\theta) \text{ and } \underline{e}^T \underline{p}_n \underline{\theta}^T \underline{\xi}(\underline{x}) \leq 0) \tag{6}$$

$$\dot{\underline{\theta}} = \Gamma \left\{ \gamma \underline{e}^T \underline{p}_n \underline{\xi}(\underline{x}) \right\} \text{ if } ((|\underline{\theta}| = M_\theta) \text{ and } \underline{e}^T \underline{p}_n \underline{\theta}^T \underline{\xi}(\underline{x}) > 0) \tag{7}$$

where  $\underline{p}_n$  is extracted from  $\mathbf{P}$  matrix solving the Lyapunov equation [7],  $M_\theta$  is the actuator constraint,  $\gamma$  is the parameter to modify the speed of the adaptive law.

The projection operator  $\Gamma\{*\}$  can be calculated as:

$$\Gamma \left\{ \gamma \underline{e}^T \underline{p}_n \underline{\xi}(\underline{x}) \right\} = \gamma \underline{e}^T \underline{p}_n \underline{\xi}(\underline{x}) - \gamma \underline{e}^T \underline{p}_n \frac{\underline{\theta} \underline{\theta}^T \underline{\xi}(\underline{x})}{|\underline{\theta}|^2} \tag{8}$$

Feedrate values ( $f$ ) were generated on-line by the controller and fed in with the set point for the cutting force setpoint and the measured value. Adaptation is performed on-line according to (6)-(8) in each cycle of control at 0.02 seconds.

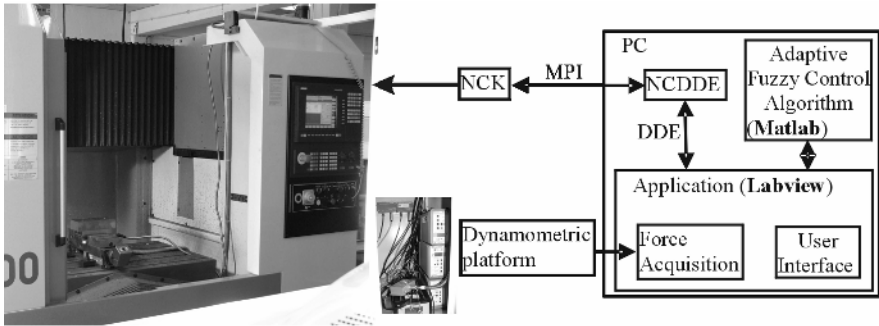
### 4 Implementation of Adaptive Fuzzy Control System

This section explains how the adaptive fuzzy controller designed in Section 3 is implemented in the open CNC. Nowadays the integration of any control system in the



CNC is a complex task that requires the use of various software utilities, technologies and development tools. Three classical technologies are used: a software-developing tool (LABVIEW and MATLAB), an open- and real-time CNC (Sinumerik 840D) and communications technology.

The general outline of the control system is depicted in Figure 1. First, the fuzzy controller was programmed, compiled and verified through simulations. A dynamic-link library (DLL) was generated as a result. The general framework for data acquisition and processing was programmed in LABVIEW including the user-interface PC (MMC). Inter-module communications between the MMC and the numerical control kernel of the CNC were established through DDE. The whole system consists of a Daqboard-2005 data-acquisition module to measure the force signal (at 10 kHz and a 0.1-10kHz bandpass filter) in the PC.

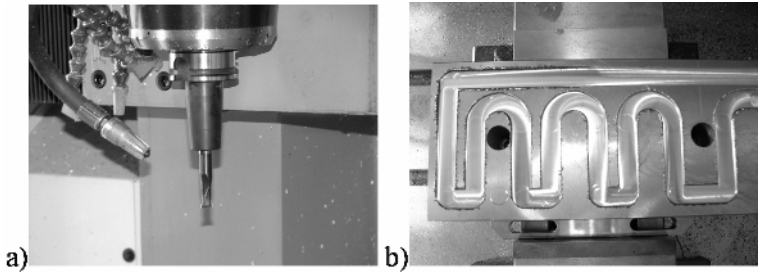


**Fig. 1.** Diagram of the control system

Cutting-force measurement is performed using a three-component platform featuring a Kistler 9257A dynamometer. The dynamometer is installed on the machine-tool table and connected to a series of Kistler 5011A amplifiers. LABVIEW includes drivers for this acquisition card and allows to enable the connection of the numerical control kernel (NCK) through dynamic date exchange (DDE). The physical connection between the application running in the PC and the open CNC is through multipoint interface (MPI). The system developed in LABVIEW sends control action to NCK using DDE. This system can be started and stopped either manually through an MMC application or under the NC program's control.

## 5 Experimental Tests – Evaluation

All cutting tests were performed on an HS1000 Kondia milling machine equipped with a Sinumerik 840D open CNC. The workpiece material used for testing was GGG40-quality steel. The actual dimensions of the profile were 180x105 (mm). The profile is depicted in Figure 2. A two-fluted Karnasch 30.6472 carbide end mill 12 mm in diameter was used as the tool for rough milling operations. The maximum depth of cut was 2 mm, the nominal spindle speed was  $sp=17000$  rpm, and the nominal feedrate,  $f=1740$  mm/min. The dimensions of the profile were 180x105 (mm). The profile and tool are depicted in Figure 2.

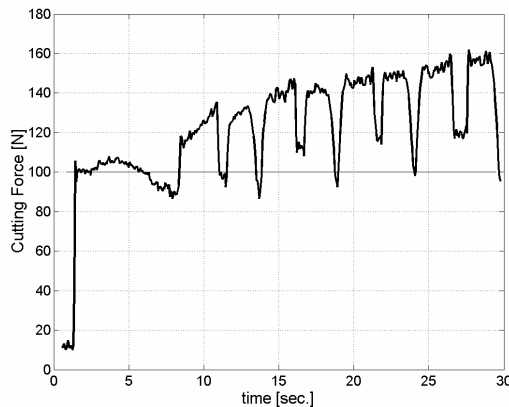


**Fig. 2.** Cutting tool, b) Testing profile

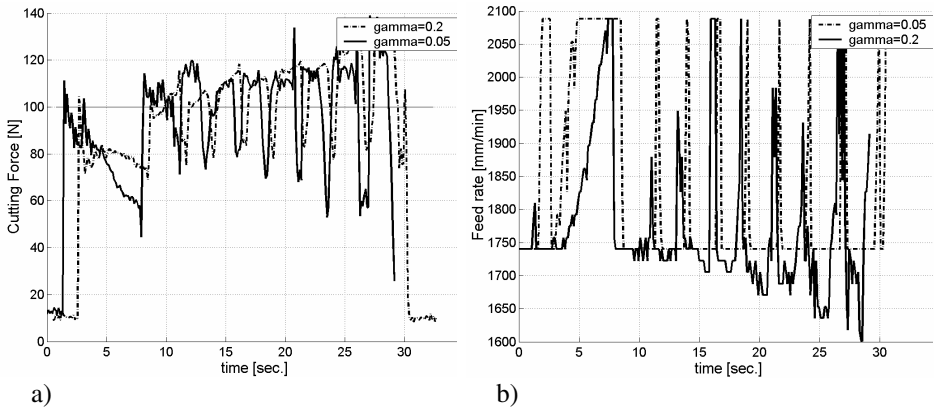
The design parameters for the adaptive fuzzy control system were  $\bar{x}_1 = \bar{x}_2 = [-16, -8, 0, 8, 16]$ ,  $\sigma_1 = \sigma_2 = [4, 4, 4, 4, 4]$ ,  $M_\theta = 100$ , including different values of gamma ( $\gamma = 0.2$  and  $\gamma = 0.05$ ). Neither expert knowledge nor training procedure was used after design. The set point of 100N was estimated according to constraint given by the power available at the spindle motor, material and the tool characteristics.

The behavior of cutting force without control systems is shown in figure 3. The cutting force can reach high values that can increase tool wear and deteriorate the quality and geometric profile of the cutting surface. Likewise, peaks in the cutting-force pattern may be the result of overloading, which would indicate a process irregularity and an increased danger of tool breakage or workpiece damage.

In order to evaluate the performance of the control systems, several runs with various set points were carried out. The results of applying the adaptive fuzzy controller to mechanize an irregular profile (see Figure 2b) are depicted in Figure 4. The response of system (cutting force) and the control action (feed rate) are plotted considering two values of gamma, in solid line for gamma=0.05 and in dashed line for gamma=0.2. The transient response is better for higher values of gamma but the



**Fig. 3.** Behavior of cutting force without control systems (CNC working alone)



**Fig. 4.** a) Time response of the cutting force, b) manipulated variable (feed rate) for two values of  $\gamma$

machining time is slightly higher. From the technological viewpoint, the overshoot is allowable for rough milling operations. It is important to note that in some approaches the goal of the controller design is to limit the percent overshoot to 20% [10].

In general, the machining time is reduced slightly using the adaptive fuzzy control and the cutting force pattern is kept constant according with geometric profile and preserving the tool useful life. Furthermore, the developed system does not require synchronization steps or a calibration procedure.

## 6 Conclusions

This work shows preliminary results of the design and implementation of an adaptive fuzzy control system in a high speed milling process, on the basis of fuzzy basis functions (FBF). The effectiveness of the developed system is shown through actual machining operations in real time. The proposed system does not require any *a priori* knowledge during the design stage and the controller parameters are adjusted on-line according to the adaptation law dealing with process disturbances. Moreover, it can run in parallel with other CNC applications without any synchronization problems. Real-time tests show a higher machining efficiency: in-process time is reduced by 5% and the cutting force is kept constant preserving the useful tool life.

## References

1. Landers R.G, Ulsoy A.G., Ma Y.-H: A comparison of model-based machining force control approaches, Int. Journal of Machinte Tools and Manufacture 44 (2004) 733-748
2. Haber-Guerra R. E.: Preliminary results in high speed machining control and monitoring. A personal perspective, School of Mechanical Engineering, Georgia Institute of Technology, July 7, (2004) Summary: [http://pmrc.marc.gatech.edu/fmpr/PMR\\_CSeminars.html](http://pmrc.marc.gatech.edu/fmpr/PMR_CSeminars.html).

3. Haber-Guerra R.E., Alique J.R., Alique A., Haber-Haber R.: Controlling a complex electromechanical process on the basis of a neurofuzzy approach, *Future Generation of Computer Systems* (doi:10.1016/j.future.2004.03.001), (2004)
4. Haber-Guerra R.E., Alique J.R.: Nonlinear internal model control using neural networks: Applications to machining processes, *Neural Computing and Applications* 13 (2004) 47-55
5. Liu Y., Zhuo L., Wang C.: Intelligent adaptive control in milling process, *International Journal of Computer Integrated Manufacturing* 12(5) (1999) 453-460
6. Wang L.X.: Stable adaptive fuzzy controllers with application to inverted pendulum tracking, *IEEE Transaction on Systems, Man, and Cybernetics*, (1996) 667-691
7. Wang L.X.: Automatic design of fuzzy controllers, *Automatica* 35 (8) (1999) 1471-1475
8. Haber-Guerra R. E., Jiménez J.E., Coronado J.L., Jiménez A.: Cutting force model for high speed machining processes, *Revista Metalurgia Madrid* 40(4) (2004) 247-258
9. Haber-Guerra R. E., Jiménez J.E., Peres C.R., Alique J.R.: An investigation of tool wear monitoring in a high-speed machining process, *Sensors and Actuators A: Physics* 116 (2004) 539-545
10. Rober S.J., Shin Y.C., Nwokah O.D.I.: A digital robust controller for cutting force control in the end milling process, *Journal of Dynamic Systems, Measurement and Control* 119 (1997) 146-152

# Sequential PN Acquisition Scheme Based on a Fuzzy Logic Controller

Rosa Maria Alsina, Jose Antonio Morán, and Joan Claudi Socoró

Communications and Signal Theory Department,  
Enginyeria i Arquitectura La Salle – Universitat Ramon Llull,  
Pg. Bonanova, 8 – 08022 Barcelona, Spain  
{[raalsina](mailto:raalsina), [moran](mailto:moran), [jclaudi](mailto:jclaudi)}@salleURL.edu

**Abstract.** One of the most important problems to be solved in a DS-SS system is the acquisition of the PN sequence. In time-varying environments this fact becomes even more important because the data decoding depends on the performance of the acquisition and tracking. In this work a new sequential acquisition system based on a fuzzy logic controller is proposed. The fuzzy logic controller extracts relevant information about the transmission conditions improving the the stability and robustness of the receiver.

## 1 Introduction

The acquisition system is the most sensitive stage in a spread spectrum receiver [1, 2]. There are several schemes to deal with this problem. The most used is the serial search strategy due to its low complexity, where the local code phase is changed step by step in equal increments until the correct acquisition point is found.

The serial search strategy has some drawbacks. The decisional system of the acquisition state is a key process in the overall system performance. This becomes more important when we deal with a time-varying channel, i.e. ionospheric channel. Several factors contribute to the randomness of the acquisition problem [3] i.e. uncertainty about the code phase, channel distortion, noise and interference, data randomness and unknown carrier phase and doppler offset.

In our system, a ionospheric channel acquisition strategy is studied [11, 12, 13] dealing especially with the specific problems of ionospheric transmission, as channel distortion and multipath. In this paper, a brief review of serial search acquisition strategies is made in section 2, a new fuzzy logic controller for the acquisition stage is presented in section 3 and some results of the tests made on the proposal are shown in section 4.

## 2 Serial Search Acquisition Improvements

The structure of a serial search system is shown in figure 1. Once obtained the energy estimation, the system must decide whether it is acquired or not. A threshold value is needed for this purpose.

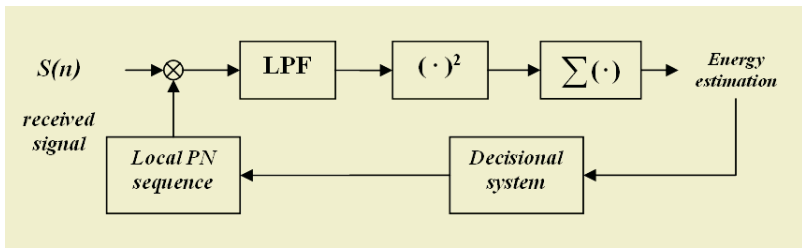


Fig. 1. Serial search strategy for spread spectrum acquisition

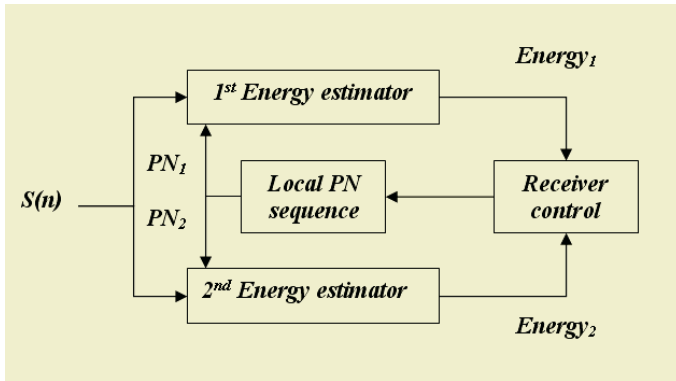


Fig. 2. AD-TLC acquisition scheme [4]

Firstly, the threshold level will be set to a constant value, without taking into account the received signal power or the noise level. Despite it is not an optimal solution when working with a fading channel, it is the most used in conventional systems due to its simplicity. The second approach was presented in [4] with some approximations to Automatic Decision Threshold Level Control (AD-TLC) (see figure 2).

There are some differences between both structures. The AD-TLC consists on two branches of a serial search model, controlled by different versions of the same PN sequence. The aim of the structure is to reach acquisition in the upper branch and the lower branch is used as a noise power reference, preventing both branches from being acquired simultaneously. This brings us to the Constant False Alarm Algorithm (CFAR)[4].

The goal of CFAR algorithm is to adapt the threshold value dynamically to keep a parameter called  $P_{fa}^1$  in a constant value. In that algorithm [4] the desired  $P_{fa}$  is set by the quotient  $\frac{r}{R}$  where  $r$  and  $R$  are integer parameters set by the controller. So the algorithm operates in the following way:

- The parameter  $y = \min(ener_1, ener_2)$  is evaluated at each integration time  $\tau_d$  where  $ener_1$  and  $ener_2$  are the energies of both branches. This value is used as an estimation of the received noise level.

<sup>1</sup> False Alarm Probability.

- After  $R$  integration times,  $\bar{y}$  will resume the number of times that the noise level is being over the threshold. The quotient  $\frac{\bar{y}}{R}$  is an estimation of the  $P_{fa}$ , which optimum value is  $\frac{r}{R}$ .
- The current threshold is increased if the estimated  $P_{fa}$  is greater than the desired or decreased if its lower.

This algorithm has been tested over the ionospheric communication channel presented in [11]. The results show that the acquisition scheme is too slow in most situations. This can be overcome by the use of an adaptive step depending on the difference of the instantaneous  $P_{fa}$  and the desired one. Another problem of this algorithm is that the control system waits  $R$  integration times to operate, so there is a compromise between the tracking speed and the robustness of the estimations.

Our proposal to improve the behaviour of the receiver is a dynamical estimation of  $P_{fa}$  using a low pass filter (LPF). This filter dynamically estimates the mean value of  $\bar{y}$ , so the computation time is then reduced. The knowledge of the dynamical response of the filter will help us in the controller design, choosing a classical PI<sup>2</sup> controller will be used for this purpose.

Some differences are shown between the features of the original CFAR controller and our improved system, because the problem of the original CFAR is its high dependence on the parameters  $R$  and  $r$ . The use of a low-pass filter provides a robust and dynamical estimation of the  $P_{fa}$ , improving the acquisition time.

Other possibilities have been investigated to improve the results achieved with the PI controller. Fuzzy logic controllers offered better results but they needed higher computational load.

### 3 The Fuzzy Logic Controller

Up to this point we have to deal with the problem of setting the correct threshold. This can be achieved with a CFAR algorithm as we have seen in the previous section. Nevertheless, the use of this serial search scheme does not offer the optimum performance. Its response highly depends on the channel variability and this difficults the design of the parameters of the serial search acquisition system.

A symbol time can be considered as a good choice for the integration time ( $\tau_d$ ) as results show. On the other hand, it also simplifies the design of the receiver blocks. Nevertheless, depending on the transmission conditions (SNR) this time may not be enough. At low signal to noise ratios a longer time must be used. The use of a sequential algorithm can overcome this problem using a variable integration time depending on the measures reliability.

It is known in [2] than the sequential method is optimum at known signal to noise ratios (SNR). This is not the case through a ionospheric channel as deep fading cause high SNR fluctuations [11]. Nevertheless, the structure of the

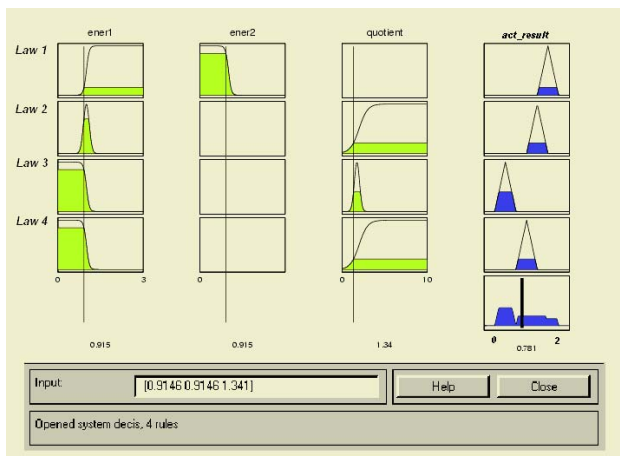
---

<sup>2</sup> Proportional integral.

receiver provides a measure of the received noise level. This value can be used to improve dynamically the system performance.

In a serial search method the decision is just taken depending on the measured value against the threshold. If it is higher an acquisition state is set and the other way round. A few amount of information is being used so the performance is far from the optimal. A logical reasoning system based on fuzzy logic theory [5, 6] has been developed to improve the system results using the knowledge about the channel. The reasoning system consists in a set of logical laws that generate a final decision [7]. Other examples of fuzzy logic applied to spread spectrum communications can be found, especially working in detection [9, 10].

In our case the result of the fuzzy logic controller is a variable  $act_{result} \in [0, 2]$  and its value is proportional to the probability of being acquired: on low probability of acquisition situations a value lower than the one is generated depending on the input values and the other way round. Each integration time  $\tau_d$  a sequence  $dec[n] = dec[n - 1] \cdot act_{result}$  is updated. Considering the case of boundary values for the variable, some limits have been set using the channel behaviour knowledge. If the sequence value is higher than 1.8 an acquisition state is set. On the other hand, a non-acquisition state is reached if the sequence is lower than 0.2. For example, if the fuzzy logic controller gives a value equal to 1.9 an acquisition will be set only in one step. Otherwise the system waits for another value. Depending if it is greater or lower than the one the sequence will increase or decrease its final value. A positive slope will take us to an acquisition state and a negative slope to a non-acquisition.



**Fig. 3.** Result computation for the fuzzy logic controller

Finally, we must design the rules to set an optimal reasoning control logic. In our case four rules have been used. More rules can be added if necessary on future implementations. However, the use of a great set of rules will increase the computational cost, so we must take care about this fact. In figure 3 we can



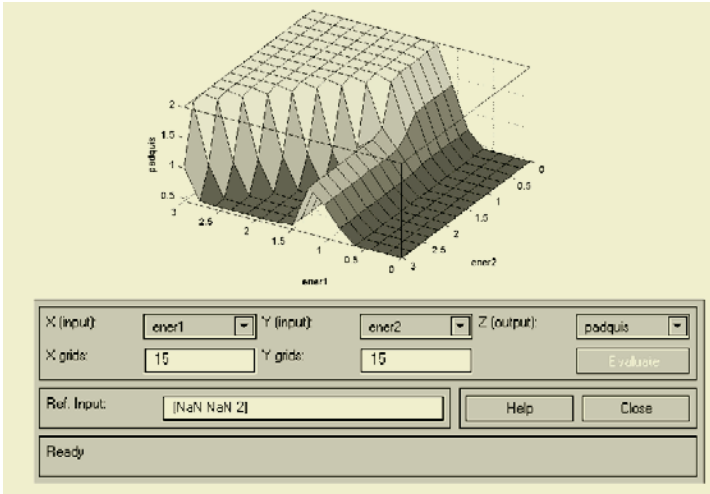


Fig. 4.  $act_{result}$  in function of  $ener_1$  and  $ener_2$

see the fuzzy logic controller implemented. Three different parameters control the acquisition algorithm as can be seen in figure 3. The variables used in the controller are the ratios  $\frac{ener_1}{threshold}$ ,  $\frac{ener_2}{threshold}$  and  $\frac{ener_1}{ener_2}$ . The variables were selected using the knowledge about the system performance.

The input variables are used in the following way:

- The first law uses the ratio  $\frac{ener_1}{threshold}$  and  $\frac{ener_2}{threshold}$ . If the first ratio is greater than one and the second one is lower there is a high probability of being acquired.
- The second law is active when  $ener_1$  is near the threshold. Then the ratio  $\frac{ener_1}{ener_2}$  is compared to one. If it is bigger than one the system is probably acquired, but maybe the threshold is not set on its optimum value, i.e. there is a fading.
- If  $ener_1$  is lower than the threshold and  $\frac{ener_1}{ener_2}$  is nearly one it shows than both measures of energy are nearly equal and lower than the threshold. The system is in a non-acquisition situation.
- Similar to the one above but now  $\frac{ener_1}{ener_2}$  is greater than one. The threshold is not correct but we may be acquired, as there is a great difference between  $ener_1$  and  $ener_2$ .

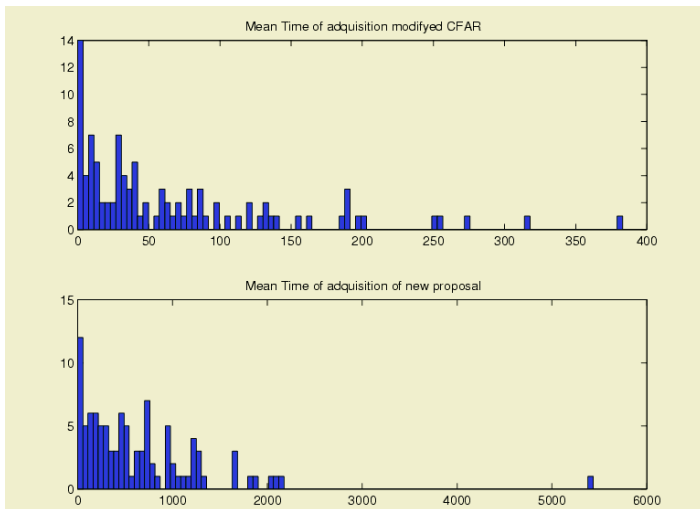
The results will be obtained using a defuzzification method based on the centroid [8]. The parameters have been used in order to generate a set of laws that control the acquisition system. The advantage of this system is that some impossible situations in a *single dwell* can be evaluated. The use of three different parameters gives us more information about the channel state and we can apply a different criterion depending on the values of this parameters as figure 4 shows. As a conclusion the reasoning capability of the receiver has been increased.

## 4 Results

In this section the results obtained with the new acquisition approach will be summarized. One of the most severe problems found was that the variations on the signal to noise ratio had a direct influence on the performance of the system.

The results obtained with this new system offer better stability to the acquisition system. In the ionospheric channel it is very important to keep acquired until there is an evidence of being lost. This needs to have a system evaluating the situation and taking the suitable decision. On the other hand, the reliability of the measures is not always the same, because it depends on the channel state. The sequential algorithm takes a variable time to conclude the acquisition state depending on the channel conditions. If we have a good signal to noise ratio a short time will be used. On the other hand we will wait until we are sure about the decision.

The final implementation of this new approach has increased the mean time of the system remaining acquired. Simulations have been done up to -17dBs of signal to noise ratio and good results have been obtained through a white gaussian channel. Obviously, the results are better than those of the CFAR algorithm. In the figure 5 we can see some results of the proposed algorithm compared to the CFAR.



**Fig. 5.** Mean time of acquisition  $pdf's$  for CFAR and Fuzzy Logic Controller

As we can see, the mean time of acquisition is highly improved with the new approach. It's also important to remark that the  $P_{fa}$  is even decreased with the use of the new approach. Nevertheless, at low SNR the algorithm is a bit slower in the initial synchronization due to the fuzzy logic controller.

## 5 Conclusions

Up to this point we have seen the process to develop the new proposed acquisition system and some results have been compared against other strategies. The main advantage of a sequential fuzzy logic based algorithm is its robustness. When it is acquired it is really difficult to leave this state even with low signal to noise ratios. This is very important when working with the ionospheric channel as the SNR is variable with time.

To fight against this problem the system uses a time varying integration. Depending on the conditions a variable is generated as an indicator of the probability of being acquired. This variable is modified on a sequential way reaching a great value of confidence. Then the system makes the decision. Although its good response when the system is acquired we should take into account that the first acquisition time is just a bit higher. This is not an important fact in our channel and in our application. The reason is that the first acquisition is going to happen fewer times than the reacquisition so we have an improvement on the total mean value. On the other hand, when a non-acquisition happens there are optimal strategies to search the correct position.

There are very important points that we must take care about. The computational load of developing a fuzzy logic controller on a DSP is very important. The mean number of FLOPS needed to do all the process related to fuzzy controller is greater compared to the conventional serial search system (including CFAR algorithm). In our experiments, we have used the received samples of the ionospheric transmission and we have processed them offline on a PC. Nevertheless, it is important to remark that some criterions can be used on the design of the controller to reduce the total number of operations. Some of them have already been used, and others will be implemented in future work, but the number of FLOPS is still too high.

As a final conclusion a new proposal for acquisition system with very good performance on a highly varying channel has been proposed. We must remark that although the cost is high over a serial search method is far away from the cost of a parallel strategy. On the other hand a new line of investigation has been opened to motivate the use of logical reasoning systems on the acquisition schemes. In our opinion the fuzzy logic theory is the best way to transfer the expert knowledge about the channel to a set of logical rules.

## References

- [1] M.K.Simon, J.K.Omura, R.A.Scholtz, B.K.Levitt: Spread Spectrum Communications Handbook, McGraw Hill, 1994.
- [2] R.L.Peterson, R.E.Ziemer, D.E.Borth: Spread Spectrum Communications Handbook, Prentice Hall, 1995.
- [3] S.G.Glisic, B.Vucetic: Spread Spectrum CDMA Systems for Wireless Communications, Artech House Publishers, 1997.

- [4] S.G.Glisic: Automatic Decision Threshold Level Control (ADTLC) in Direct Sequence Spread Spectrum System based on Matched Filtering, IEEE Transactions on Communications, Vol. 36, No. 4, April 1991.
- [5] L.A.Zadeh: Fuzzy Sets, Information and Control, Vol. 8, pages 338-353, 1965.
- [6] L.A.Zadeh: Fuzzy Logic, Computer, p. 83-92, April 1988.
- [7] L.A.Zadeh: Outline of a new approach to the analysis of complex systems and decision processes. IEEE Transactions Systems Man Cybernetics, 3, 28-44, 1973.
- [8] Werner Van Leekwijck, Etienne E. Kerre: Defuzzification: Criteria and classification. Fuzzy Sets and Systems 108, 159-178, 1999.
- [9] J.Jang, K.Ha, B.Seo, S.Lee, C.W.Lee: A Fuzzy Adaptive Multiuser Detector in CDMA Communication Systems, International Conference on Communications ICC1998.
- [10] J.Bas, A.Pérez, M.A.Lagunas: Fuzzy recursive Symbol-by-Symbol detector for single user CDMA receivers, ICASSP 2001, Salt Lake City (Utah-USA).
- [11] J.Ll.Pijoan, C.Vilella, J.R.Regué, J.C.Socoró, J.A.Morán: DSP-based ionospheric radiolink using DS-SS, COST-262 fourth management comitee meeting, 22-23 november 1999, Barcelona (Spain).
- [12] J.A.Morán, J.C.Socoró, X.Jové, J.L. Pijoan, F. Tarrés: Multiresolution Adaptive Structure for acquisition in DS-SS receiver, ICASSP 2001, Salt Lake City (Utah-USA).
- [13] C.Mateo, R.M.Alsina: Diseno de un sistema de control adaptativo a las condiciones del canal para un sistema de adquisición de un receptor DS-SS, XIX Congreso de la Unión Científica Nacional de Radio, URSI2004, Barcelona (Spain).

# Fuzzy Logic System for Students' Evaluation

José Antonio Montero, Rosa Maria Alsina, Jose Antonio Morán,  
and Mariona Cid

Communications and Signal Theory Department,  
Enginyeria i Arquitectura La Salle – Universitat Ramon Llull,  
Pg. Bonanova, 8 – 08022 Barcelona, Spain  
{montero, ralsina, moran, si08186}@salleURL.edu

**Abstract.** A fuzzy evaluation system to decide critical students' final marks is presented. The marks of the assessment tests made by students along the academic year are transformed into linguistic terms and used for assigning values to linguistic variables. Subjective criteria as i.e. *student's interest* or *student's progression* are used in order to decide whether every analyzed student passes or fails the considered subject. This paper presents the features of the subject where the fuzzy system has been applied in, the proposed fuzzy decision system, and results when it is applied using real students' marks.

## 1 Introduction

The evaluation of learning achieved by a student is a very difficult task for teachers. Results from tests, exams and exercises are normally used to decide whether the learning of a student is proper or not. It is important to remember that assessments should focus on students acquiring knowledge, as well as the disposition to use skills and strategies and apply them appropriately [1, 2]. The design of all these tests must be matched with subject objectives for this purpose. Assuming that proposed tests are adequate, the question is: how can the results of these tests be used in order to decide the students' final marks? Normally, teachers use to weigh up every partial result in the most adequate way, in order to obtain the final mark. However, this final mark is sometimes very close to threshold mark, which indicates the limit between passing and failing, and it is not clear if that student (critical student along this paper) should pass or fail.

In this paper, a fuzzy logic system to help the teacher deciding the final mark of critical students is presented. Firstly, we briefly review the subject which this model is applied in, and the reasons why we have used fuzzy logic to solve this situation. Secondly we propose our fuzzy logic system, based on the features presented in the previous section. Finally, results obtained when the proposed model is used with real student marks are discussed.

## 2 Scenario Description

The proposed fuzzy system has been applied using marks obtained by students at Linear Algebra subject. This subject belongs to first academic year curricu-

lum of telecommunications engineering in Ingeniería i Arquitectura La Salle from Barcelona (Spain). Details of methodological reform process applied along previous two years in this subject can be consulted in [3, 4, 5].

There are four different teachers for this subject and there are six groups of students, each one of about sixty pupils. Students make periodic tests every week. These tests are i.e., an exercise during the class, a little exam, or some homework exercises can be given. These different tests are of medium difficulty level. At the end of every trimester<sup>1</sup>, about ten marks have been obtained from these monitoring tests made by every student. Moreover, students must do the trimester exam, which is corrected by all teachers<sup>2</sup>. The difficulty level of this exam is high and his mark supposes the 70% of the trimester final mark<sup>3</sup>. At the end of the third trimester, teachers review all these marks and decide whether students pass or fail the subject. To calculate this final mark, trimester marks are averaged and the threshold mark is fixed to five (being ten the best mark). It is important to be taken into account that the minimum trimester mark to compute a valid final mark is four. If a trimester mark is lower than four, the student is required to do another exam at the end of the academic year.

A problem is presented when a student's final mark is near the passing limit. Does a critical student with a 4.9 i.e. as final mark have less knowledge than a student with a 5 as final mark? It is really necessary to put a threshold mark to decide if a student passes or fails the subject, but all four teachers think that other criteria should be used to decide the critical students' final mark. Criteria as progression, interest shown by student, or students' general results, i.e., could help to decide the final marks of critical students. These criteria are not objective as calculating average marks and, as a consequence, it is more difficult to apply them equally for all students. Despite teachers decide all together how to take into account each criterion, it is very difficult to guarantee their application in the same sense by all teachers due to their subjectivity.

Fuzzy logic theory can solve this problem because allows an approximation to teacher reasoning. Fuzzy set theory was formalized by Professor Lotfi A. Zadeh in 1965 [6, 7] and, since then, it is applied to solve situations in a lot of disciplines. Application of fuzzy logic in control systems can be prominent [8, 9, 10] but fuzzy theory can be also found in others fields as, i.e. education [11, 12, 13].

In this paper, an approximation to teacher reasoning using fuzzy theory is presented. Concepts as student's interest are treated defining different fuzzy sets (i.e. the following fuzzy sets *null*, *low*, *normal*, *high* and *very high* can be defined). All additional criteria used to evaluate critical students can be handled as this, and several rules (implications) can be fixed by teachers in order to decide the critical students' final marks. The whole model proposed is presented in following section.

---

<sup>1</sup> The academic year is divided into three trimesters.

<sup>2</sup> Every teacher corrects a problem of every student's exam, this way objectivity in the criteria is maintained.

<sup>3</sup> Every trimester, teachers put a partial mark to each student.

### 3 Fuzzy Decision System

The fuzzy system will be employed in the evaluation process of critical students to decide their final marks. A student is assigned into the critical students set when his final mark, calculated averaging his trimester marks, is below but close to five or when his final mark is over five but not all his trimester marks are over four. The first thing we must define is the criteria we should use in order to decide these students' final marks.

Teachers of Linear Algebra take into account the following criteria in order to decide the final mark: interest and effort shown by student, his progression along the academic year, the others students' results, and how about those trimester marks are lower than four if this situation occurs. Therefore, the fuzzy evaluation system should use this information to offer to the teacher a final recommendation about every critical student's mark.

#### 3.1 Linguistic Variables Defined in the Fuzzy Model

**Recommendation:** The output fuzzy system offers to the teacher a recommendation regarding critical students' final mark. Four possible recommendations are defined: *strongly fail*, *fail*, *pass* and *strongly pass*. These different kinds of recommendation correspond to the fuzzy sets defined of the output linguistic variable *recommendation*. The offered recommendation depends on four input linguistic variables: *interest and effort*, *progression*, *student's mark in relation to mean group mark*, and *trimester mark in relation to four*. These variables are presented in following points.

**Interest and Effort:** During the trimester, students are requested to deliver to the teacher different kind of exercises, controls and exams. The results of these tests, and the number of tests presented to the teacher by every student, allow calculate an indicator of the interest shown by student. I.e., if 30 tests have been proposed by the teacher along the academic year and a student has presented all tests with proper results, interest and effort shown by him can be accepted as very high. On the other hand, if a student has only presented 10 tests and their marks are low, his interest and effort towards Linear Algebra subject can be considered as low. Fuzzy sets *null*, *low*, *normal*, *high* and *very high* have been defined for the linguistic variable *Interest and effort*.

**Progression:** The linguistic variable *progression* is function of student's trimester partial marks. The second trimester mark is compared to the first trimester mark, and the third trimester mark is compared with the second trimester mark. These comparisons allow to generate an indicator of student progression. Fuzzy sets *decreasing*, *regular*, and *increasing* have been defined for this linguistic variable.

**Student's Mark in Relation to Mean Group Mark:** The position of student into a ranking about all students' marks is another data to be taken into account in our context. I.e., if the best all students' mark is 6, then a 4.7 mark

can be considered a good mark. On the other side, if the best all students' mark is 10, a 4.7 will be considered a bad mark. Fuzzy sets which have been defined for this variable are *below mean*, *mean* and *above mean*.

**Trimester Mark in Relation to Four:** All three student's trimester marks must be larger than four in order to average them to calculate his final mark. But it is not the same situation whether a student has a 3.75 mark or a 1.25 mark as trimester mark. Furthermore, it is not the same case if the student has only one trimester mark below four or two trimester marks.

Fuzzy sets defined for this variable, if this situation is presented, are *very below four* and *little below four*. If this situation is not presented the linguistic variable *trimester mark in relation to four* is fixed to *not below four* value.

Figure 1 shows a general diagram of the proposed system.

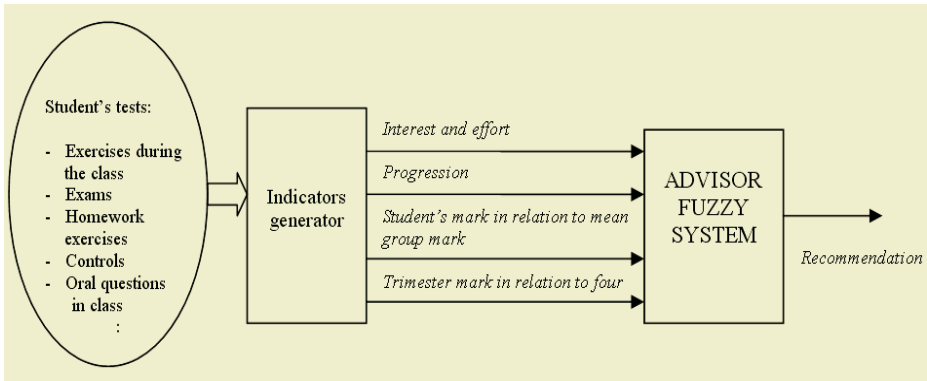


Fig. 1. General diagram of the proposed system

### 3.2 Linguistic Rules

We use rules in natural language to describe teacher's reasoning. Experienced teachers can give us the necessary information to construct the fuzzy system rules. A set of rules like the following is created:

IF (*Trimester mark in relation to four is very below four*) THEN (*recommendation is strongly fail*).

IF (*Trimester mark in relation to four is little below four*) AND (*Student's mark in relation to mean group mark is above mean*) AND (*progression is increasing*) AND (*interest and effort is very high*) THEN (*recommendation is strongly pass*).

Tables 1 to 6 show rules used to model teacher's reasoning. Each table is presented assuming two fixed variables. SF, F, P and SP represent *strongly fail*, *fail*, *pass* and *strongly pass* output fuzzy sets respectively.

Finally, the linguistic values of the linguistic variable *recommendation* are defuzzified using the *center of gravity* method.



**Table 1.** Recommendation whether Trimester mark in relation to four is little below four and progression is decreasing for all cases of Interest and effort

| Student's mark<br>in relation to<br>mean group mark | null | low | normal | high | very high |
|-----------------------------------------------------|------|-----|--------|------|-----------|
| above mean                                          | F    | F   | F      | F    | P         |
| mean                                                | SF   | SF  | SF     | F    | F         |
| below mean                                          | SF   | SF  | SF     | SF   | F         |

**Table 2.** Recommendation whether Trimester mark in relation to four is little below four and progression is regular for all cases of Interest and effort

| Student's mark<br>in relation to<br>mean group mark | null | low | normal | high | very high |
|-----------------------------------------------------|------|-----|--------|------|-----------|
| above mean                                          | F    | F   | P      | P    | P         |
| mean                                                | SF   | SF  | F      | F    | P         |
| below mean                                          | SF   | SF  | F      | F    | F         |

**Table 3.** Recommendation whether Trimester mark in relation to four is little below four and progression is increasing for all cases of Interest and effort

| Student's mark<br>in relation to<br>mean group mark | null | low | normal | high | very high |
|-----------------------------------------------------|------|-----|--------|------|-----------|
| above mean                                          | F    | P   | P      | P    | SP        |
| mean                                                | F    | F   | F      | P    | P         |
| below mean                                          | SF   | SF  | F      | F    | P         |

**Table 4.** Recommendation whether Trimester mark in relation to four is not below four and progression is decreasing for all cases of Interest and effort

| Student's mark<br>in relation to<br>mean group mark | null | low | normal | high | very high |
|-----------------------------------------------------|------|-----|--------|------|-----------|
| above mean                                          | F    | F   | P      | P    | SP        |
| mean                                                | F    | F   | F      | P    | P         |
| below mean                                          | SF   | SF  | F      | F    | P         |

**Table 5.** Recommendation whether Trimester mark in relation to four is not below four and progression is regular for all cases of Interest and effort

| Student's mark<br>in relation to<br>mean group mark | null | low | normal | high | very high |
|-----------------------------------------------------|------|-----|--------|------|-----------|
| above mean                                          | P    | P   | SP     | SP   | SP        |
| mean                                                | F    | F   | P      | P    | SP        |
| below mean                                          | F    | F   | F      | P    | SP        |

**Table 6.** Recommendation whether Trimester mark in relation to four is not below four and progression is increasing for all cases of Interest and effort

| Student's mark<br>in relation to<br>mean group mark | null | low | normal | high | very high |
|-----------------------------------------------------|------|-----|--------|------|-----------|
| above mean                                          | P    | SP  | SP     | SP   | SP        |
| mean                                                | P    | P   | SP     | SP   | SP        |
| below mean                                          | F    | F   | P      | P    | SP        |

## 4 Results

In this section, the results applying the fuzzy logic system with real students' marks are presented. Matlab fuzzy logic toolbox is used to implement this fuzzy decision system. Students' marks are contained in an Excel file, and Matlab handles these marks using Matlab Excel Link toolbox.

Results of 2003-04 academic year have been introduced to the fuzzy advisor system. Fifty critical students have been selected. These students have been evaluated by teachers and also by the fuzzy system. The teachers met to analyze critical students and lasted about four hours to adopt the final decision for all cases. All final recommendations were decided by the fuzzy system immediately. A comparison between two strategies to decide critical students' final marks is now presented: forty six critical students were equally evaluated by the fuzzy system and teachers. The system did not evaluate four critical students and listed them with the *very difficult to decide* label.

The system should be tested on more case studies, but no performance comparison can be made with previous courses marks, because the evaluation criteria did not fit the fuzzy system rules. This will be experimented in the future, using 2004-05 academic year results.

## 5 Conclusions

A fuzzy system to help teacher to decide the final mark of students who are very close to the pass limit mark has been presented. The validity of the proposed

fuzzy system has been proved with real students' marks. The proposed system recommendation agrees with the teacher's final decision in 92% of cases. In 8% of cases, the system does not offer any advice. It is important to emphasize that the system did not offer any incorrect recommendation. After applying the fuzzy system, the number of critical students who must be evaluated by teacher is reduced from fifty to four.

Guaranteeing the same way to apply subjective criteria by different teachers is a very difficult task and, also, very time consuming. The use of the proposed fuzzy system guarantees the lack of bias when subjective criteria are applied and time dedicated by teacher is strongly reduced.

Nowadays Europe is immersed in a very deep reform process regarding university education [14]. This reform will demand methodological and pedagogical changes and, because of that, application of new students' assessment and evaluation methods will be needed. Different subjective criteria will be taken into account along the students' evaluation process and teacher's final decision will be adopted using both subjective and objective criteria [2].

The design of new fuzzy systems in order to be applied when new pedagogical and methodological models are used by teachers is proposed as future work. How can these fuzzy systems improve their results applying different methods of implication and defuzzification [15], or defining different kinds of fuzzy set forms is proposed as another future work line.

## References

- [1] Kay Burke: How to assess authentic learning. Merrill Prentice Hall. (1999) 57-95.
- [2] Preston D. Feden, Robert M. Vogel: Methods of teaching. Applying cognitive science to promote student learning. Mc Graw-Hill. (2003) 257-302.
- [3] José A. Montero, Nuria Escudero, Fco. Javier Pajares, Oscar Garcia, José A. Moran: Implantación de una metodología constructivista en la docencia del álgebra en ingeniería. Proceedings of XII Congreso Universitario de Innovación Educativa en las Enseñanzas Técnicas, Barcelona, (2004).
- [4] José A. Montero, Javi Pajares, Miquel Ferrer, Oscar Garcia, Nuria Escudero, José A. Moran, Elisa Martínez: Aprender a utilizar conceptos algebraicos ante problemas 'reales' utilizando el aprendizaje cooperativo, Proceedings of III Jornada sobre aprendizaje cooperativo, Barcelona, (2003).
- [5] José A. Montero, José A. Moran, Elisa Martínez: Evaluación continua. Experiencia llevada a cabo en la asignatura de álgebra lineal en Ingeniería La Salle. Proceedings of I Congreso de la RED-U, Castellón de la Plana, (2003).
- [6] Lotfi A. Zadeh: Fuzzy sets. Information and Control, 8, (1965) 338-353.
- [7] L.A. Zadeh. Outline of a new approach to the analysis of complex systems and decision processes. IEEE Transactions Systems Man Cybernetics, 3, (1973) 28-44.
- [8] Chuen Chien Lee. Fuzzy logic in control systems: fuzzy logic controller-part I. IEEE Transactions on systems, man, and cybernetics, Vol.20, (1990) 404-418.
- [9] Chuen Chien Lee. Fuzzy logic in control systems: fuzzy logic controller-part II. IEEE Transactions on systems, man, and cybernetics, Vol.20, (1990) 419-433.
- [10] Li-Xin Wang: A course in fuzzy systems and control. Prentice-Hall. (1997).

- [11] Kinshuk, Alexander Nikov, Ashok Patel. Adaptative tutoring in business education using fuzzy backpropagation approach. Proceedings of the 9th International Conference on Human-Computer Interaction. New Orleans, USA, (2001).
- [12] Ahmad M. Ibrahim. Assessment of distance education quality using fuzzy sets model. Proceedings of International Conference on Engineering Education, Oslo, Norway (2001).
- [13] Alenka Kavcic, Rafael Pedraza, Harold Molina, Francisco Valverde, Jess Cid and Angel Navia. Student modelling based on fuzzy inference mechanisms. Proceedings of EUROCON, Slovenia (2003).
- [14] Bologna process - towards a European Higher Education Area.  
<http://www.bologna-berlin2003.de/index.htm>
- [15] Werner Van Leekwijck, Etienne E. Kerre: Defuzzification: Criteria and classification. Fuzzy Sets and Systems 108, (1999) 159-178.

## Author Index

- Agís, Rodrigo 168, 471  
Aguilar-Ruiz, Jesús S. 742, 804  
Aguirre, Carlos 130  
Ahn, Tae-Chon 391  
Aisbett, Janet 646  
Aler, R. 257  
Alique, José R. 1230  
Allende, Héctor 341  
Alonso, Carlos J. 779  
Alsina, Rosa Maria 1238, 1246  
Andonie, Răzvan 438  
Angulo, Cecilio 709  
Araque, Alfonso 99  
Archambeau, Cédric 820  
Arenas, M.G. 669, 685  
Arenas-García, J. 622  
Arleo, Angelo 107  
Arnold, Mike 471  
Atencia, Miguel 582  
Aunet, Snorre 438, 486  
Awad, Mohammed 280, 289, 399
- Bandera Rubio, Juan Pedro 677  
Barakova, Emilia 122  
Barbati, Giulia 1083  
Barbi, Michele 115  
Barbour, Boris 168  
Becerikli, Yasar 943  
Becerra, J.A. 333, 717  
Bedoya, Guillermo 1059  
Beiu, Valeriu 438, 486  
Belatreche, A. 552  
Bellas, F. 333, 717, 829  
Benítez-Read, Jorge S. 1132  
Bermejo, Sergi 1059  
Berry, Hugues 306  
Berza, J. Andrés 9  
Bezzi, Michele 107  
Blasco, X. 993  
Blesa, Maria 25
- Blum, Christian 25  
Böhm, M. 1075  
Boloix-Tortosa, Rafael 1100  
Bono, Antonio 463  
Boucheny, Christian 136  
Boumeridja, Hafida 582  
Bousquet, Olivier 176  
Bravo, José M. 1
- Cabello, Santiago 1214  
Cabestany, Joan 433, 1059  
Campos, Doris 130  
Cantador, Iván 208  
Carrillo, Richard R. 136, 168, 471  
Cascado, D. 479  
Castillo, P.A. 669, 685  
Celma, Santiago 463  
Çetinkaya, Bahadır 812  
Chakkor, O. 845  
Chen, S.Y. 952  
Chen, Shengyong 661  
Chillemi, Santi 115  
Chung, Duck-Jin 541  
Chung, TaeChoong 217  
Cid, Mariona 1246  
Civit, A. 479, 518, 534  
Claverol-Tinturé, Enric 433  
Coenen, Olivier J.-M.D. 107, 136  
Comas, C. 850  
Cortés, Jesús M. 161  
Cortez, Paulo 59  
Cotta, Carlos 50  
Crespo, J.L. 17
- Danciu, Daniela 366  
D'Angelo, Egidio 107  
Deguchi, Toshinori 612  
del Campo-Ávila, José 795  
del Carmen Vargas-González,  
María 750

- del Castillo-Gomariz, R. 598  
 Demirel, Hasan 935  
 De Moor, Bart 1018  
 Díaz, Francisco 1051  
 Díaz Casás, V. 1188  
 Diaz-Diaz, Norberto 804  
 Di Garbo, Angelo 115  
 Di Stefano, Antonio 590  
 Djupdal, Asbjørn 486  
 Domínguez, E. 67  
 Dorado, Julián 382, 968  
 Dorrnsoro, José R. 208  
 Drias, Habiba 318  
 Du, Haifeng 91  
 Du, Yajun 771  
 Duro, R.J. 17, 333, 717, 829, 1188
- Eickhoff, Ralf 264  
 Eleyan, Alaa 935  
 Ercal, Fikret 881  
 Espinoza, Marcelo 1018
- Fan, Hong 654  
 Farfán, Javier 1214  
 Faundez-Zanuy, Marcos 1172, 1180  
 Ferber, Markus 455  
 Fernández, Eduardo 510, 913  
 Ferrández, J. Manuel 906  
 Fernández, O. 257  
 Fernández-Redondo, Mercedes 604  
 Figueiras-Vidal, Aníbal R. 233, 622  
 Figueiredo, Karla 424  
 Flórez, Francisco 630  
 Fraile-Ardanuy, Jesús 1124  
 Franco, Leonardo 1  
 François, Damien 758  
 Franzmeier, Marc 725  
 Fuente, M.J. 1116
- Galindo Riaño, Pedro 192  
 García, José 630  
 García, Juan Manuel 630  
 Garcia, M. 913
- Garcia, R. 1116  
 García-Herrera, Guillermo 1214  
 García-Laencina, Pedro J. 233  
 García-Pedrajas, N. 598  
 Garrigós, F. Javier 906  
 Gestal, Marcos 382  
 Giaconia, Costantino 590  
 Gibbon, Greg 646  
 Gil-Jiménez, P. 873  
 Gil-Pita, R. 865  
 Giráldez, Raúl 804  
 Glackin, B. 552  
 Gómez, Pedro 1051  
 Gómez-Moreno, H. 873  
 Gomez-Rodriguez, F. 518, 534  
 Gómez-Verdejo, V. 622  
 González, C. 42  
 González, Jesús 272, 289, 399  
 Górriz, J.M. 837, 845, 1043  
 Górriz-Sáez, Juan M. 1035  
 Graña, M. 913  
 Guan, Qiu 661  
 Guerrero Vázquez, Elisa 192  
 Guillén, Alberto 272, 289, 399, 1027  
 Gutiérrez-González, D. 357
- Haber-Guerra, Rodolfo 1230  
 Haber-Haber, Rodolfo 1230  
 Häfliger, Ph. 494  
 Hao, Jin 1010  
 He, Wuhong 91  
 He, Zhenya 857  
 Hernández, Antonio 630  
 Hernández-Espinosa, Carlos 604  
 Herrera, Luis Javier 272, 280, 289, 399, 960, 1027  
 Herrero, Elías 1164  
 Herrero, J.M. 993  
 Hervás-Martínez, César 83, 598  
 Hollmén, Jaakko 1002  
 Hornillo, S. 837  
 Hornillo-Mellado, Susana 1035  
 Huang, Gaoming 857  
 Huo, Hongwei 734

- Ishdorj, Tseoren-Onolt 145  
 Ishii, Naohiro 612
- Jeong, Jong-Geun 889  
 Jerez, José M. 1  
 Ji, Yongnan 1010  
 Jiao, Licheng 91  
 Jiménez, G. 479, 518, 534  
 Joya, Gonzalo 582  
 Jun, Sung-Hae 349  
 Jung, Tobias 960
- Karaboğa, Nurhan 812  
 Karan, Tayfun M. 943  
 Kaulmann, Tim 455  
 Keck, Ingo R. 1067  
 Kelly, P.M. 564  
 Keung Cheung, Wing 75  
 Kim, Dong-Sun 541  
 Kim, Hyeong-Gyun 889  
 Kim, Hyun-Ki 407  
 Kim, Hyun-Sik 541  
 Kim, In-Young 921  
 Kim, Sun I. 921  
 Kim, Yong-Ho 889  
 Kim, Yong-Kab 407  
 Kim, Yong-Soo 391  
 Koklukaya, Etem 1108  
 Kolle Riis, H. 494  
 Kolman, Eyal 1222  
 Krebs, Marie-Odile 1140
- Lafuente-Arroyo, S. 873  
 Lamas, A. 1188  
 Lamela, Horacio 502  
 Lang, Elmar W. 249, 1067, 1075  
 La Rocca, Michele 200  
 Larrañaga, P. 42  
 Lee, Doo-Soo 921  
 Lee, John A. 1091  
 Lee, Jong-Beom 416  
 Lee, Jung Hoon 446  
 Lee, Sang-Min 921  
 Lee, SeungGwan 217  
 Lee, Sungyoung 899
- Lee, Youn-Sung 541  
 Lendasse, Amaury 985, 1002, 1010  
 León-Barranco, Agustín 637  
 Leu, Ming C. 1197  
 Li, Huaqing 927  
 Li, Jing 91  
 Likharev, Konstantin K. 446  
 Linares-Barranco, A. 479, 518, 534  
 Linares-Barranco, B. 479  
 Liu, Lifang 734  
 López, Isaac 1214  
 López-Callejas, Régulo 1132  
 López-Ferrerías, F. 865  
 Lopez-Peña, F. 829, 1188  
 López-Rodríguez, Domingo 1149  
 López-Rubio, Ezequiel 750  
 López-Rubio, José Miguel 750  
 Lourens, Tino 122  
 Lozano, J.A. 42
- Madani, Kurosh 976  
 Maguire, L.P. 552, 564  
 Maldonado-Bascón, S. 873  
 Malti, Rachid 976  
 Malutan, Raul 1051  
 Maravall, D. 1206  
 Margalot, Michael 1222  
 Marro, Joaquín 161  
 Martín-Clemente, Rubén 1035  
 Martín-del-Brío, Bonifacio 463  
 Martín-Vide, Carlos 145  
 Martínez, Antonio 510, 527  
 Martínez, Francisco 1164  
 Martínez, J. Javier 906  
 Martínez, M. 993  
 Martínez-Estudillo, Alfonso 83  
 Martínez-Estudillo, Francisco 83  
 Martín-Vide, Carlos 145  
 McDaid, L.M. 564  
 McGinnity, T.M. 552, 564  
 Medrano, Nicolás 463  
 Merelo, J.J. 669, 685  
 Mérida-Casermeyro, Enrique 1149  
 Mermoud, Grégory 572  
 Miro, L. 534

- Monte-Moreno, E. 850  
 Montero, José Antonio 1246  
 Morales-Bueno, Rafael 795  
 Morán, Jose Antonio 1238, 1246  
 Moreno, Sebastián 341  
 Morillas, Christian A. 510, 527  
 Muñoz, J. 67  
 Muñoz-Romero, S. 622  
 Murillo-Fuentes, Juan José 1100
- Nassabay, Salua 1067  
 Navarro Mérida, Guillermo Mar  
 1172  
 Navia-Vázquez, A. 357  
 Nepomuceno, Isabel 804  
 Neves, José 59  
 Ni, Jie 693  
 Nieu, Thierry 107
- Odinaev, Karina 241  
 Oh, Sung-Kwun 391, 407, 416  
 Okada, Hiroyuki 153  
 Olier, Iván 787  
 Orrite, Carlos 1164  
 Ortega, Julio 297  
 Ortega-Moral, M. 622  
 Ortigosa, Eva M. 168, 471  
 Ortíz-Boyer, Domingo 83, 598  
 Ortiz-de-Lazcano-Lobato,  
 Juan Miguel 750  
 Oster, M. 479  
 Ouali, Abdelaziz 1140  
 Oysal, Yusuf 1108  
 Oz, Cemil 881, 1197
- Pacheco, Marco Aurélio 424  
 Pacheco-Sotelo, Joel O. 1132  
 Panarese, Alessandro 115  
 Pardo, Diego E. 709  
 Park, Keon-Jun 391  
 Parrado-Hernández, E. 357  
 Pascual, Pedro 130  
 Patricio, M.A. 1206  
 Paz, R. 518, 534
- Pazos, Alejandro 99, 968  
 Pedreira, Nieves 382  
 Pedrycz, Witold 407, 416  
 Pei, Zheng 771  
 Pelayo, Francisco J. 510, 527  
 Peña, Carlos-Andres 572  
 Perna, Cira 200  
 Pino, Begoña 527, 845  
 Pizarro Junquera, Joaquin 192  
 Pomares, Héctor 34, 272, 280, 289,  
 399, 1027  
 Porcaro, Camillo 1083  
 Porto, Ana 99  
 Prieto, Alberto 272, 280, 471, 685,  
 829, 1027  
 Prieto, Oscar J. 779  
 Puntonet, Carlos G. 249, 845,  
 1051, 1067, 1075
- Qi, Feihu 927
- Rabuñal, Juan R. 382, 968  
 Raichelgauz, Igal 241  
 Ramdane Cherif, Amar 1140  
 Ramírez, A. 42  
 Ramírez, J. 837  
 Ramos, C. 993  
 Ramos-Jiménez, Gonzalo 795  
 Räsvan, Vladimir 366  
 Rejón, J. 1206  
 Reyes-García, Carlos A. 637  
 Reyhani, Nima 985, 1010  
 Reyneri, Leonardo M. 527  
 Riquelme, José C. 742  
 Rivera, Antonio J. 297  
 Rivero, Daniel 968  
 Rocha, Miguel 59  
 Rodellar, Victoria 1051  
 Rodriguez, Juan José 779  
 Rodriguez, M.A. 518  
 Rogel, Cristian 341  
 Roh, Seok-Beom 416  
 Rojas, Fernando 289, 1043  
 Rojas, Ignacio 272, 280, 297, 399,  
 1027  
 Romero, Enrique 225  
 Romero, G. 669, 685



Romero, Samuel F. 510, 527  
 Ros, Eduardo 136, 168, 471  
 Rosa-Zurera, M. 865  
 Rosell, Xavier 433  
 Ruan, Da 1132  
 Rubio G. 34  
 Rückert, Ulrich 264, 455, 725  
 Ruiz, Roberto 742  
 Ruiz de Angulo, Vicente 701  
 Ruiz-Enciso, Jorge A. 1132  
 Ruiz-Llata, Marta 502

Sadeg, Souhila 318  
 Sainz, G.I. 1116  
 Sakagami, Masamichi 153  
 Salas, Rodrigo 341  
 Sánchez, Antonio 1214  
 Sanchez, Eduardo 572  
 Sancho-Gómez, José-Luis 233  
 Sandoval Hernández, Francisco  
 582, 677  
 Santos, J. 717  
 Sasaki, Hiroshi 612  
 Schölkopf, Bernhard 176, 960  
 Seamari, Yamina 1214  
 Segura, J.C. 837  
 Serrano, Eduardo 130  
 Serrano-García, Jesús 233  
 Socoró, Joan Claudi 1238  
 Solé-Casals, J. 850  
 Song, In-Ho 921  
 Song, Qing 693  
 Sorjamaa, Antti 985  
 Sow, Gustave 976  
 Stadlthanner, Kurt 249, 1075  
 Sun, DonChuan 1157  
 Suykens, Johan A.K. 1018  
 Szeto, Kwok Yip 75

Tecchio, Franca 1083  
 Teixeira, A.R. 1075  
 Téllez, Ricardo A. 709  
 Temam, Olivier 306  
 Tereshko, Valery 326  
 Theis, Fabian J. 249, 1075

Thiaw, Lamine 976  
 Tikka, Jarkko 1002  
 Toledo, F. Javier 906  
 Tomé, A.M. 1075  
 Torras, Carme 701  
 Torrealdea, F.J. 913  
 Torres, Joaquín J. 161  
 Torres-Sospedra, Joaquín 604  
 Townsend, James T. 646

Upegui, Andres 572  
 Usero, L. 1206  
 Utrilla-Manso, M. 865

Valenzuela, Olga 272, 1027, 1043  
 Valls, J.M. 257  
 Vellasco, Marley 424  
 Vellido, Alfredo 787  
 Veredas, Francisco 1214  
 Verleysen, Michel 758, 820, 1091  
 Vicen-Bueno, R. 865  
 Vico, Francisco 1214  
 Vo, Dinh Minh Nhat 899  
 Vrins, Frédéric 1091

Wang, Baoshu 734  
 Wang, Shaoyu 927  
 Wang, W.L. 952  
 Wang, Wan-liang 661  
 Wang, Zhijie 654  
 Wei, Xiaopeng 374  
 Weston, Jason 176  
 Witkowski, Ulf 455, 725  
 Wu, JianJun 1157  
 Wu, Q.X. 552

Xia, Bing-bing 661  
 Xiao, G. 952  
 Xu, Jin 374  
 Xu, Yang 771

Yahi, Safa 318  
 Yamakawa, Hiroshi 153  
 Yáñez Escolano, Andrés 192

- Yang, Luxi 857  
Yao, C.Y. 952  
Yi, Liangzhong 771  
Yilmaz, Ahmet Serdar 1108  
Ying, Y.S. 952  
  
Zappasodi, Filippo 1083  
Zatorre, Guillermo 463  
  
Zawadski, Artur 438  
Zeevi, Yehoshua Y. 241  
Zhang, Qiang 374  
Zhong, YongJun 1157  
Zhou, Changjiu 677  
Zhou, Dongsheng 374  
Zufiria, Pedro J. 9, 1124