Jun Wang
Xiaofeng Liao
Zhang Yi (Eds.)

LNCS 3498

Advances in Neural Networks – ISNN 2005

Second International Symposium on Neural Networks
Chongqing, China, May/June 2005
Proceedings, Part III

3 Part III

Springer

Jun Wang Xiaofeng Liao Zhang Yi (Eds.)

Advances in Neural Networks – ISNN 2005

Second International Symposium on Neural Networks
Chongqing, China, May 30 - June 1, 2005
Proceedings, Part III

Springer

Volume Editors

Jun Wang
The Chinese University of Hong Kong
Department of Automation and Computer-Aided Engineering
Shatin, New Territories, Hong Kong
E-mail: jwang@acae.cuhk.edu.hk

Xiaofeng Liao
Chongqing University, School of Computer Science and Engineering
Chongqing, 400044, China
E-mail: xfliao@cqu.edu.cn

Zhang Yi
University of Electronic Science and Technology of China
School of Computer Science and Engineering
Chengdu, Sichuan, China
E-mail: zhangyi@uestc.edu.cn

Library of Congress Control Number: 2005926239

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

ISSN 0302-9743
ISBN-10 3-540-25914-7 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-25914-5 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Olgun Computergrafik
Printed on acid-free paper SPIN: 11427469 06/3142 5 4 3 2 1 0

Preface

This book and its sister volumes constitute the proceedings of the 2nd International Symposium on Neural Networks (ISNN 2005). ISNN 2005 was held in the beautiful mountain city Chongqing by the upper Yangtze River in southwestern China during May 30–June 1, 2005, as a sequel of ISNN 2004 successfully held in Dalian, China. ISNN emerged as a leading conference on neural computation in the region with increasing global recognition and impact. ISNN 2005 received 1425 submissions from authors on five continents (Asia, Europe, North America, South America, and Oceania), 33 countries and regions (Mainland China, Hong Kong, Macao, Taiwan, South Korea, Japan, Singapore, Thailand, India, Nepal, Iran, Qatar, United Arab Emirates, Turkey, Lithuania, Hungary, Poland, Austria, Switzerland, Germany, France, Sweden, Norway, Spain, Portugal, UK, USA, Canada, Venezuela, Brazil, Chile, Australia, and New Zealand). Based on rigorous reviews, 483 high-quality papers were selected by the Program Committee for presentation at ISNN 2005 and publication in the proceedings, with an acceptance rate of less than 34%. In addition to the numerous contributed papers, 10 distinguished scholars were invited to give plenary speeches and tutorials at ISNN 2005.

The papers are organized into many topical sections under 20 coherent categories (theoretical analysis, model design, learning methods, optimization methods, kernel methods, component analysis, pattern analysis, signal processing, image processing, financial analysis, system modeling, control systems, robotic systems, telecommunication networks, incidence detection, fault diagnosis, power systems, biomedical applications, and industrial applications, and other applications) spanning all major facets of neural network research and applications. ISNN 2005 provided an international forum for the participants to disseminate new research findings and discuss the state of the art. It also created a pleasant opportunity for the participants to interact and exchange information on emerging areas and future challenges of neural network research.

Many people made significant efforts to ensure the success of this event. The ISNN 2005 organizers are grateful to Chongqing University, Southwest Normal University, Chongqing University of Posts and Telecommunications, Southwest Agricultural University, and Chongqing Education College for their sponsorship; grateful to the National Natural Science Foundation of China for the financial support; and to the Asia Pacific Neural Network Assembly, the European Neural Network Society, the IEEE Computational Intelligence Society, and the IEEE Circuits and Systems Society for their technical co-sponsorship. The organizers would like to thank the members of the Advisory Committee for their spiritual support, the members of the Program Committee for reviewing the papers, and the members of the Publication Committee for checking the papers. The organizers would particularly like to thank the publisher, Springer, for their cooperation in publishing the proceedings as three volumes of the Lecture Notes

in Computer Science series. Last but not least, the organizers would like to thank all the authors for contributing their papers to ISNN 2005. Their enthusiastic contributions and participation were essential parts of the symposium with which the organizers were proud to be involved.

May 2005

Jun Wang
Xiaofeng Liao
Zhang Yi

ISNN 2005 Organization

ISNN 2005 was organized and sponsored by Chongqing University, Southwest Normal University, Chongqing University of Posts and Telecommunications, Southwest Agricultural University, and Chongqing Education College in cooperation with the Chinese University of Hong Kong. It was technically cosponsored by the Asia Pacific Neural Network Assembly, the European Neural Network Society, the IEEE Circuits and Systems Society, and the IEEE Computational Intelligence Society. It was financially supported by the National Natural Science Foundation of China and K.C. Wong Education Foundation of Hong Kong.

General Chair

Jun Wang, Hong Kong, China

Advisory Committee Co-chairs

Shun-ichi Amari, Tokyo, Japan
Jacek M. Zurada, Louisville, USA

Advisory Committee Members

Zheng Bao, X'ian, China
Guoliang Chen, Hefei, China
Ruwei Dai, Beijing, China
Chunbo Feng, Nanjing, China
Walter J. Freeman, Berkeley, USA
Toshio Fukuda, Nagoya, Japan
Kunihiko Fukushima, Tokyo, Japan
Aike Guo, Shanghai, China
Zhenya He, Nanjing, China
Okyay Kaynak, Istanbul, Turkey
Frank L. Lewis, Fort Worth, USA
Yanda Li, Beijing, China
Erkki Oja, Helsinki, Finland
Tzyh-Jong Tarn, St. Louis, USA
Shoujue Wang, Beijing, China
Youshou Wu, Beijing, China
Bo Zhang, Beijing, China
Nanning Zheng, Xi'an, China

Steering Committee Chairs

Xiaohong Li, Chongqing, China
Yixin Zhong, Beijing, China

Steering Committee Members

Wlodzislaw Duch, Torun, Poland
Yinguo Li, Chonqing, China
Max Q.H. Meng, Hong Kong, China
Marios M. Polycarpou, Cincinnati, USA
Yuhui Qiu, Chongqing, China
Zhengqi Sun, Beijing, China
DeLiang Wang, Columbus, USA
Zhongfu Wu, Chongqing, China
Zongben Xu, Xi'an, China
Gary G. Yen, Stillwater, USA
Fuliang Yin, Dalian, China
Juebang Yu, Chengdu, China

Program Committee Co-chairs

Xiaofeng Liao, Chongqing, China
Zhang Yi, Chengdu, China

Program Committee Members

Shigeo Abe, Kobe, Japan
Sabri Arik, Istanbul, Turkey
Amit Bhaya, Rio de Janeiro, Brazil
Abdesselam Bouzerdoum, Wollongong, Australia
Jinde Cao, Nanjing, China
Laiwan Chan, Hong Kong, China
Ke Chen, Manchester, UK
Luonan Chen, Osaka, Japan
Tianping Chen, Shanghai, China
Yen-Wei Chen, Kyoto, Japan
Yiu Ming Cheung, Hong Kong, China
Zheru Chi, Hong Kong, China
Hyungsuk Cho, Dae Jeon, Korea
Andrzej Cichocki, Tokyo, Japan
Shuang Cong, Hefei, China
Chuanyin Dang, Hong Kong, China
Meng Joo Er, Singapore
Mauro Forti, Siena, Italy
Jun Gao, Hefei, China
Chengan Guo, Dalian, China
Ping Guo, Beijing, China
Zengguang Hou, Beijing, China
Baogang Hu, Beijing, China
Dewen Hu, Changsha, China
Jinglu Hu, Fukuoka, Japan
Danchi Jiang, Hobart, Australia
Licheng Jiao, Xi'an, China
Nikola Kasabov, Auckland, New Zealand
Hon Keung Kwan, Windsor, Canada
Irwin King, Hong Kong, China
Cees van Leeuwen, Tokyo, Japan
Xiaoli Li, Birmingham, UK
Yangmin Li, Macau, China
Yuanqing Li, Singapore
Yanchun Liang, Changchun, China
Lizhi Liao, Hong Kong, China
Chin-Teng Lin, Hsingchu, Taiwan
Ju Liu, Jinan, China
Qing Liu, Wuhan, China
Baoliang Lu, Shanghai, China
Hongtao Lu, Shanghai, China
Fa-Long Luo, San Jose, USA
Zhiwei Luo, Nagoya, Japan
Qing Ma, Kyoto, Japan
Satoshi Matsuda, Narashino, Japan
Tetsuo Nishi, Fukuoka, Japan
Stanislaw Osowski, Warsaw, Poland
Paul S. Pang, Auckland, New Zealand
Rudy Setiono, Singapore
Yi Shen, Wuhan, China
Daming Shi, Singapore
Peter Sincak, Kosice, Slovakia
Jianbo Su, Shanghai, China
Changyin Sun, Nanjing, China
Fuchun Sun, Beijing, China
Ron Sun, Troy, USA
Johan Suykens, Leuven, Belgium
Ah Hwee Tan, Singapore
Ying Tan, Hefei, China
Dan Wang, Singapore
Lipo Wang, Singapore
Wanliang Wang, Hangzhou, China
Wei Wu, Dalian, China
Michel Verleysen, Louvain, Belgium
Hong Yan, Hong Kong, China
Mao Ye, Chengdu, China
Wen Yu, Mexico City, Mexico
Zhigang Zeng, Hefei, China
Huaguang Zhang, Shenyang, China
Liming Zhang, Shanghai, China
Liqing Zhang, Shanghai, China
Chunguang Zhou, Changchun, China

Special Sessions Chair

Derong Liu, Chicago, USA

Organizing Chairs

Guoyin Wang, Chongqing, China
Simon X. Yang, Guelph, Canada

Finance Chairs

Guangyuan Liu, Chongqing, China
Qingyu Xiong, Chongqing, China
Yu Wu, Chongqing, China

Publication Co-chairs

Yi Chai, Chongqing, China
Hujun Yin, Manchester, UK
Jianwei Zhang, Hamburg, Germany

Publicity Co-chairs

Min Han, Dalian, China
Fengchun Tian, Chongqing, China

Registration Chairs

Yi Chai, Chongqing, China
Shaojiang Deng, Chongqing, China

Local Arrangements Chairs

Wei Zhang, Chongqing, China
Jianqiao Yu, Chongqing, China

Secretariat and Webmaster

Tao Xiang, Chongqing, China

Table of Contents, Part III

12 Control Systems

NN-Based Iterative Learning Control Under Resource Constraints: A Feedback Scheduling Approach 1
Feng Xia and Youxian Sun

Sequential Support Vector Machine Control of Nonlinear Systems by State Feedback 7
Zonghai Sun, Youxian Sun, Xuhua Yang, and Yongqiang Wang

RBFNN-Based Multiple Steady States Controller for Nonlinear System and Its Application 15
Xiugai Li, Dexian Huang, and Yihui Jin

Sliding Mode Control for Uncertain Nonlinear Systems Using RBF Neural Networks 21
Xu Zha and Pingyuan Cui

Adaptive Backstepping Neural Network Control for Unknown Nonlinear Time-Delay Systems 30
Weisheng Chen and Junmin Li

Multiple Models Adaptive Control Based on RBF Neural Network Dynamic Compensation 36
Junyong Zhai and Shumin Fei

Stability Analysis and Performance Evaluation of an Adaptive Neural Controller 42
Dingguo Chen and Jiaben Yang

Adaptive Inverse Control System Based on Least Squares Support Vector Machines 48
Xiaojing Liu, Jianqiang Yi, and Dongbin Zhao

H-Infinity Control for Switched Nonlinear Systems Based on RBF Neural Networks 54
Fei Long, Shumin Fei, and Shiyou Zheng

Neural Networks Robust Adaptive Control for a Class of MIMO Uncertain Nonlinear Systems 60
Tingliang Hu, Jihong Zhu, Chunhua Hu, and Zengqi Sun

Adaptive Critic for Controller Malfunction Accommodation 69
Gary G. Yen

Output Based Fault Tolerant Control of Nonlinear Systems
Using RBF Neural Networks 79
Min Wang and Donghua Zhou

Fault Tolerant Control of Nonlinear Processes
with Adaptive Diagonal Recurrent Neural Network Model 86
Ding-Li Yu, Thoonkhin Chang, and Jin Wang

Dealing with Fault Dynamics in Nonlinear Systems
via Double Neural Network Units 92
Yong D. Song, Xiao H. Liao, Cortney Bolden, and Zhi Yang

Neural Adaptive Singularity-Free Control by Backstepping
for Uncertain Nonlinear Systems 98
Zhandong Yu and Qingchao Wang

Parameter Estimation of Fuzzy Controller
Using Genetic Optimization and Neurofuzzy Networks 107
Sungkwun Oh, Seokbeom Roh, and Taechon Ahn

A Fuzzy CMAC Controller with Eligibility 113
Zhipeng Shen, Chen Guo, Jianbo Sun, and Chenjun Shi

A Novel Intelligent Controller Based on Modulation of Neuroendocrine System . . 119
Bao Liu, Lihong Ren, and Yongsheng Ding

Batch-to-Batch Optimal Control Based on Support Vector Regression Model 125
Yi Liu, Xianhui Yang, Zhihua Xiong, and Jie Zhang

Nonlinear Predictive Control Based on Wavelet Neural Network Applied
to Polypropylene Process 131
Xiaohua Xia, Zhiyan Luan, Dexian Huang, and Yihui Jin

Neural Network Control of Heat Exchanger Plant 137
Mahdi Jalili-Kharaajoo

Remote Controller Design of Networked Control Systems
Based on Self-constructing Fuzzy Neural Network 143
Yi Li, Qinke Peng, and Baosheng Hu

Sliding Mode Control for Cross Beam Simulation System via Neural Network . . . 150
Hongchao Zhao, Qingjiu Xu, Wenjin Gu, and Tingxue Xu

Vibration Suppression of Adaptive Truss Structure
Using Fuzzy Neural Network 155
Shaoze Yan, Kai Zheng, and Yangmin Li

Experimental Investigation of Active Vibration Control
Using a Filtered-Error Neural Network and Piezoelectric Actuators 161
Yali Zhou, Qizhi Zhang, Xiaodong Li, and Woonseng Gan

Compensating Modeling and Control for Friction
Using RBF Adaptive Neural Networks 167
Yongfu Wang, Tianyou Chai, Lijie Zhao, and Ming Tie

Torque Control of Switched Reluctance Motors
Based on Flexible Neural Network 173
Baoming Ge, Aníbal T. de Almeida, and Fernando J.T.E. Ferreira

Position Control for PM Synchronous Motor Using Fuzzy Neural Network 179
Jun Wang, Hong Peng, and Xiao Jian

SVM Based Lateral Control for Autonomous Vehicle 185
Hanqing Zhao, Tao Wu, Daxue Liu, Yang Chen, and Hangen He

Control of Reusable Launch Vehicle Using Neuro-adaptive Approach 192
Yong D. Song, Xiao H. Liao, M.D. Gheorghiu, Ran Zhang, and Yao Li

13 Robotic Systems

A Neural Network Based on Biological Vision Learning
and Its Application on Robot 198
Ying Gao, Xiaodan Lu, and Liming Zhang

Discrete-Time Adaptive Controller Design for Robotic Manipulators
via Neuro-fuzzy Dynamic Inversion 204
Fuchun Sun, Yuangang Tang, Lee Li, and Zhonghang Yin

General Underactuated Cooperating Manipulators
and Their Control by Neural Network 210
S. Murat Yeşiloğlu and Hakan Temeltaş

Intelligent Fuzzy Q-Learning Control of Humanoid Robots 216
Meng Joo Er and Yi Zhou

Performance Analysis of Neural Network-Based Uncalibrated
Hand-Eye Coordination 222
Jianbo Su

Formation Control for a Multiple Robotic System
Using Adaptive Neural Network 228
Yangmin Li and Xin Chen

Tip Tracking of a Flexible-Link Manipulator
with Radial Basis Function and Fuzzy System 234
Yuangang Tang, Fuchun Sun, and Zengqi Sun

Obstacle Avoidance for Kinematically Redundant Manipulators
Using the Deterministic Annealing Neural Network 240
Shubao Liu and Jun Wang

BP Networks Based Trajectory Planning and Inverse Kinematics
of a Reconfigurable Mars Rover 247
Liping Zhang, Shugen Ma, Bin Li, Zheng Zhang, Guowei Zhang, and Binggang Cao

A Novel Path Planning Approach Based on AppART
and Particle Swarm Optimization 253
Jian Tang, Jihong Zhu, and Zengqi Sun

A Neuro-fuzzy Controller for Reactive Navigation
of a Behaviour-Based Mobile Robot 259
Anmin Zhu, Simon X. Yang, Fangju Wang, and Gauri S. Mittal

Research on the Calibration Method for the Heading Errors
of Mobile Robot Based on Evolutionary Neural Network Prediction 265
Jinxia Yu, Zixing Cai, Xiaobing Zou, and Zhuohua Duan

Adaptive Neural-Network Control
for Redundant Nonholonomic Mobile Modular Manipulators 271
Yangmin Li, Yugang Liu, and Shaoze Yan

A Neural Network-Based Camera Calibration Method
for Mobile Robot Localization Problems 277
Anmin Zou, Zengguang Hou, Lejie Zhang, and Min Tan

Abnormal Movement State Detection and Identification
for Mobile Robots Based on Neural Networks 285
Zhuohua Duan, Zixing Cai, Xiaobing Zou, and Jinxia Yu

A Neural Network Based Method for Shape Measurement
in Steel Plate Forming Robot 291
Hua Xu, Peifa Jia, and Xuegong Zhang

Recurrent Networks for Integrated Navigation 297
Jianguo Fu, Yingcai Wang, Jianhua Li, Zhenyu Zheng, and Xingbo Yin

14 Telecommunication Networks

Application of Different Basis and Neural Network Turbo Decoding Algorithm
in Multicarrier Modulation System over Time-Variant Channels 303
Yupeng Jia, Dongfeng Yuan, Haixia Zhang, and Xinying Gao

Blind Detection of Orthogonal Space-Time Block Coding
Based on ICA Schemes 309
Ju Liu, Bo Gu, Hongji Xu, and Jianping Qiao

Improvement of Borrowing Channel Assignment
by Using Cellular Probabilistic Self-organizing Map 315
Sitao Wu and Xiaohong Wang

FPGA Realization of a Radial Basis Function
Based Nonlinear Channel Equalizer 320
Poyueh Chen, Hungming Tsai, ChengJian Lin, and ChiYung Lee

Varying Scales Wavelet Neural Network
Based on Entropy Function and Its Application in Channel Equalization 326
Mingyan Jiang, Dongfeng Yuan, and Shouliang Sun

Robust Direction of Arrival (DOA) Estimation Using RBF Neural Network
in Impulsive Noise Environment 332
Hong Tang, Tianshuang Qiu, Sen Li, Ying Guo, and Wenrong Zhang

Quantum Neural Network for CDMA Multi-user Detection 338
Fei Li, Shengmei Zhao, and Baoyu Zheng

A New QoS Routing Optimal Algorithm in Mobile Ad Hoc Networks
Based on Hopfield Neural Network 343
Jian Liu, Dongfeng Yuan, Song Ci, and Yingji Zhong

Content Filtering of Decentralized P2P Search System
Based on Heterogeneous Neural Networks Ensemble 349
Xianghua Fu and Boqin Feng

Collaborative Filtering Based on Neural Networks Using Similarity 355
Eunju Kim, Myungwon Kim, and Joungwoo Ryu

Using Double-Layer One-Class Classification
for Anti-jamming Information Filtering 361
Qiang Sun, Jianhua Li, Xinran Liang, and Shenghong Li

Remote OS Fingerprinting Using BP Neural Network 367
Wenwei Li, Dafang Zhang, and Jinmin Yang

Emotional Learning Based Intelligent Traffic Control of ATM Networks 373
Mahdi Jalili-Kharaajoo, Mohammadreza Sadri,
and Farzad Habibipour Roudsari

Multi-agent Congestion Control for High-Speed Networks
Using Reinforcement Co-learning 379
Kaoshing Hwang, Mingchang Hsiao, Chengshong Wu, and Shunwen Tan

Multi-scale Combination Prediction Model
with Least Square Support Vector Machine for Network Traffic 385
Zunxiong Liu, Deyun Zhang, and Huichuan Liao

Clustering Algorithm Based on Wavelet Neural Network Mobility Prediction in Mobile Ad Hoc Network 391
Yanlei Shang, Wei Guo, and Shiduan Cheng

Internet Traffic Prediction by W-Boost: Classification and Regression 397
Hanghang Tong, Chongrong Li, Jingrui He, and Yang Chen

Fuzzy Neural Network for VBR MPEG Video Traffic Prediction 403
Xiaoying Liu, Xiaodong Liu, Xiaokang Lin, and Qionghai Dai

15 Incidence Detection

Building an Intrusion Detection System Based on Support Vector Machine and Genetic Algorithm 409
Rongchang Chen, Jeanne Chen, Tungshou Chen, Chunhung Hsieh, Teyu Chen, and Kaiyang Wu

Fusions of GA and SVM for Anomaly Detection in Intrusion Detection System . . 415
Dong Seong Kim, Ha-Nam Nguyen, Syng-Yup Ohn, and Jong Sou Park

A Genetic SOM Clustering Algorithm for Intrusion Detection 421
Zhenying Ma

Intrusion Detection Based on Dynamic Self-organizing Map Neural Network Clustering 428
Yong Feng, Kaigui Wu, Zhongfu Wu, and Zhongyang Xiong

Intrusion Detection Based on MLP Neural Networks and K-Means Algorithm . . . 434
Hongying Zheng, Lin Ni, and Di Xiao

Feature Selection and Intrusion Detection Using Hybrid Flexible Neural Tree 439
Yuehui Chen, Ajith Abraham, and Ju Yang

Detection of Epileptic Spikes with Empirical Mode Decomposition and Nonlinear Energy Operator 445
Suyuan Cui, Xiaoli Li, Gaoxiang Ouyang, and Xinping Guan

Neural Networks for Solving On-Line Outlier Detection Problems 451
Tianqi Yang

Pedestrian Detection by Multiple Decision-Based Neural Networks 457
Chen Huang, Guangrong Tang, and Yupin Luo

A Visual Automatic Incident Detection Method on Freeway Based on RBF and SOFM Neural Networks 463
Xuhua Yang, Qiu Guan, Wanliang Wang, and Shengyong Chen

A Self-organizing Map Method for Optical Fiber Fault Detection and Location . . 470
Yi Chai, Wenzhou Dai, Maoyun Guo, Shangfu Li, and Zhifen Zhang

Anomaly Internet Network Traffic Detection
by Kernel Principle Component Classifier 476
Hanghang Tong, Chongrong Li, Jingrui He, Jiajian Chen, Quang-Anh Tran, Haixin Duan, and Xing Li

Intelligent Hierarchical Intrusion Detection System
for Secure Wireless Ad Hoc Network 482
Peng Fu, Deyun Zhang, Lei Wang, and Zhongxing Duan

A New Approach of Network Intrusion Detection Using HVDM-Based SOM 488
Lei Wang, Yong Yang, and Shixin Sun

A Novel Approach to Corona Monitoring 494
Chiman Kwan, Tao Qian, Zhubing Ren, Hongda Chen, Roger Xu, Weijen Lee, Hemiao Zhang, and Joseph Sheeley

16 Fault Diagnosis

Multi-class Probability SVM Fusion Using Fuzzy Integral for Fault Diagnosis . . . 501
Zhonghui Hu, Yunze Cai, Xing He, Ye Li, and Xiaoming Xu

A Rapid Response Intelligent Diagnosis Network
Using Radial Basis Function Network 508
Guangrui Wen, Liangsheng Qu, and Xining Zhang

An Integrated Approach to Fault Diagnosis
Based on Variable Precision Rough Set and Neural Networks 514
Qingmin Zhou and Chenbo Yin

Hybrid PSO Based Wavelet Neural Networks for Intelligent Fault Diagnosis 521
Qianjin Guo, Haibin Yu, and Aidong Xu

Global-Based Structure Damage Detection
Using LVQ Neural Network and Bispectrum Analysis 531
Guangming Dong, Jin Chen, Xuanyang Lei, Zuogui Ning, Dongsheng Wang, and Xiongxiang Wang

Fault Detection for Plasma Etching Processes Using RBF Neural Networks 538
Yaw-Jen Chang

Detecting Sensor Faults for a Chemical Reactor Rig
via Adaptive Neural Network Model 544
Ding-Li Yu and Dingwen Yu

Optimal Actuator Fault Detection via MLP Neural Network for PDFs 550
Lei Guo, Yumin Zhang, Chengliang Liu, Hong Wang, and Chunbo Feng

Feature Selection and Classification of Gear Faults Using SOM 556
Guanglan Liao, Tielin Shi, Weihua Li, and Tao Huang

Application of Fuzzy SOFM Neural Network
and Rough Set Theory on Fault Diagnosis for Rotating Machinery 561
Dongxiang Jiang, Kai Li, Gang Zhao, and Jinhui Diao

Identification of the Acoustic Fault Sources of Underwater Vehicles
Based on Modular Structure Variable RBF Network 567
Linke Zhang, Lin He, Kerong Ben, Na Wei, Yunfu Pang, and Shijian Zhu

A Dynamic Recurrent Neural Network Fault Diagnosis
and Isolation Architecture for Satellite's Actuator/Thruster Failures 574
Li Li, Liying Ma, and Khashayar Khorasani

Fault Detection in Reaction Wheel of a Satellite
Using Observer-Based Dynamic Neural Networks 584
Zhongqi Li, Liying Ma, and Khashayar Khorasani

Adaptive Wavelet Packet Neural Network Based Fault Diagnosis
for Missile's Amplifier 591
Zhijie Zhou, Changhua Hu, Xiaoxia Han, and Guangjun Chen

Crack Detection in Supported Beams
Based on Neural Network and Support Vector Machine 597
Long Liu and Guang Meng

Early Loosening Fault Diagnosis of Clamping Support
Based on Information Fusion 603
Weixiang Sun, Jin Chen, Xing Wu, Fucai Li, Guicai Zhang, and GM Dong

Insulating Fault Diagnosis of XLPE Power Cables
Using Multi-parameter Based on Artificial Neural Networks 609
Xiaolin Chen, Yonghong Cheng, Zhelei Zhu, Bo Yue, and Xiaojun Xie

17 Power Systems

A Hybrid Method and Its Application for Power System. 616
Xusheng Yang, Yong You, Wanxing Sheng, and Sunan Wang

Fuzzy Neural Very-Short-Term Load Forecasting
Based on Chaotic Dynamics Reconstruction 622
Hongying Yang, Hao Ye, Guizeng Wang, and Tongfu Hu

Application of Neural Networks
for Very Short-Term Load Forecasting in Power Systems 628
Hungcheng Chen, Kuohua Huang, and Lungyi Chang

Next Day Load Forecasting Using SVM 634
Xunming Li, Dengcai Gong, Linfeng Li, and Changyin Sun

Peak Load Forecasting Using the Self-organizing Map 640
Shu Fan, Chengxiong Mao, and Luonan Chen

Ship Power Load Prediction Based on RST and RBF Neural Networks 648
Jianmei Xiao, Tengfei Zhang, and Xihuai Wang

Contingency Screening of Power System
Based on Rough Sets and Fuzzy ARTMAP 654
Youping Fan, Yunping Chen, Wansheng Sun, Dong Liu, and Yi Chai

Intelligent Neuro-fuzzy Based Predictive Control
of a Continuous Stirred Tank Reactor 662
Mahdi Jalili-Kharaajoo and Farzad Habibipour Roudsari

Adaptive Neuro-fuzzy SVC for Multimachine Hybrid Power System
Stability Improvement with a Long of Double Circuit Transmission Lines 668
Chamni Jaipradidtham

Application of BP Network-Based Multi-sensor Fusion Techniques
in Measurement of the Unburned Carbon in Fly Ash 674
Gaowei Yan, Gang Xie, Keming Xie, Zehua Chen, and Hongbing Wang

18 Biomedical Applications

Classification of Nuclear Receptor Subfamilies
with RBF Kernel in Support Vector Machine 680
Jun Cai and Yanda Li

Prediction of Contact Maps in Proteins
Based on Recurrent Neural Network with Bias Units 686
Guixia Liu, Chunguang Zhou, Yuanxian Zhu, and Wengang Zhou

A SVR-Based Multiple Modeling Algorithm
for Antibiotic Fermentation Process Using FCM 691
Yaofeng Xue and Jingqi Yuan

Non-parametric Statistical Tests for Informative Gene Selection 697
Jinwen Ma, Fuhai Li, and Jianfeng Liu

An Information Criterion for Informative Gene Selection 703
Fei Ge and Jinwen Ma

OPTOC-Based Clustering Analysis
of Gene Expression Profiles in Spectral Space 709
Shuanhu Wu, Alan Wee Chung Liew, and Hong Yan

Model the Relationship Between Gene Expression and TFBSs
Using a Simplified Neural Network with Bayesian Variable Selection 719
Xiaobo Zhou, Kuang-Yu Liu, Guangqin Li, and Stephen Wong

Synchrony of Basic Neuronal Network Based on Event Related EEG 725
Xiaotong Wen, Xiaojie Zhao, and Li Yao

Non-negative Matrix Factorizations
Based Spontaneous Electroencephalographic Signals Classification
Using Back Propagation Feedback Neural Networks 731
Mingyu Liu, Jue Wang, and Chongxun Zheng

Neural Networks Preprocessing Based Adaptive Latency Change Estimation
of Evoked Potentials 737
Yongmei Sun, Tianshuang Qiu, Wenhong Liu, Wenqiang Guo, and Hui Li

Blind Estimation of Evoked Potentials
Based on Fractional Lower Order Statistics 742
Daifeng Zha, Tianshuang Qiu, and Xiaobing Li

Wavelet Denoise on MRS Data Based on ICA and PCA 748
Jian Ma, Zengqi Sun, Guangbo Dong, and Guihai Xie

Hard Margin SVM for Biomedical Image Segmentation 754
Chen Pan, Xiangguo Yan, and Chongxun Zheng

Multisensors Information Fusion with Neural Networks
for Noninvasive Blood Glucose Detection 760
Wei Wang, Lanfeng Yan, Baowei Liu, and Heng Zhang

Disease Diagnosis Using Query-Based Neural Networks 767
Ray-I Chang

Study of BP Neural Network and Its Application
in Lung Cancer Intelligent Diagnosis 774
Xuemei Huang, Zhide Tang, and Caixin Sun

New Methodology of Computer Aided Diagnostic System on Breast Cancer 780
HeeJun Song, SeonGu Lee, Dongwon Kim, and GwiTae Park

Spiculated Lesion Detection in Digital Mammogram
Based on Artificial Neural Network Ensemble 790
Ning Li, Huajie Zhou, Jinjiang Ling, and Zhihua Zhou

Classification of Psychiatric Disorders Using Artificial Neural Network 796
Shishir Bashyal

Multilevel Neural Network to Diagnosis Procedure
of Traditional Chinese Medicine 801
Zhanquan Sun, Jianqiang Yi, and Guangcheng Xi

19 Industrial Applications

An Automated Blowing Control System Using the Hybrid Concept of Case Based Reasoning and Neural Networks in Steel Industry 807
Jonghan Kim, Eoksu Sim, and Sungwon Jung

Neural Networks Based Multiplex Forecasting System of the End-Point of Copper Blow Period 813
Lihua Xue, Hongzhong Huang, Yaohua Hu, and Zhangming Shi

Modeling and Prediction of Electric Arc Furnace Based on Neural Network and Chaos Theory 819
Fenghua Wang, Zhijian Jin, and Zishu Zhu

Modeling and Prediction of Violent Abnormal Vibration of Large Rolling Mills Based on Chaos and Wavelet Neural Networks 827
Zhonghui Luo, Xiaozhen Wang, Xiaoning Xue, Baihai Wu, and Yibin Yu

Neural Grey Box Model for Power Estimation in Semiautogenous Mill 833
Tito Valenzuela, Karina Carvajal, Gonzalo Acuña, Max Chacón, and Luis Magne

Neural Network Based On-Line Shrinking Horizon Re-optimization of Fed-Batch Processes 839
Zhihua Xiong, Jie Zhang, Xiong Wang, and Yongmao Xu

Chip Speed Prediction Model for Optimization of Semiconductor Manufacturing Process Using Neural Networks and Statistical Methods 845
Tae Seon Kim

Using ANNs to Model Hot Extrusion Manufacturing Process 851
Kesheng Wang, Per Alvestad, Yi Wang, Qingfeng Yuan, Minglun Fang, and Lingiang Sun

Application Research of Support Vector Machines in Condition Trend Prediction of Mechanical Equipment 857
Junyan Yang and Youyun Zhang

Comparative Study on Engine Torque Modelling Using Different Neural Networks 865
Ding-Li Yu and Michael Beham

A Hybrid Intelligent Soft-Sensor Model for Dynamic Particle Size Estimation in Grinding Circuits 871
Ming Tie, Heng Yue, and Tianyou Chai

Application of Artificial Neural Networks in Abrasive Waterjet Cutting Process . . 877
Yiyu Lu, Xiaohong Li, Binquan Jiao, and Yong Liao

Intelligent Tool Condition Monitoring System for Turning Operations 883
Hongli Gao and Mingheng Xu

A Recurrent Neural Network Modeling
for Automotive Magnetorheological Fluid Shock Absorber 890
Changrong Liao, Honghui Zhang, Miao Yu, Weimin Chen, and Jiansheng Weng

Geometrical Error Compensation of Gantry Stage Using Neural Networks 897
Kok Kiong Tan, Sunan Huang, V. Prahlad, and Tong Heng Lee

Neural Particle Swarm Optimization for Casing Damage Prediction 903
Quansheng Dou, Chunguang Zhou, Guanyu Pan, Hongwen Luo, and Quan Liu

A Novel Chamber Scheduling Method in Etching Tools
Using Adaptive Neural Networks 908
Hua Xu, Peifa Jia, and Xuegong Zhang

CFNN Without Normalization-Based Acetone Product Quality Prediction 914
Jiao Wang and Xiong Wang

Combining Classifiers in Software Quality Prediction:
A Neural Network Approach 921
Qi Wang, Jie Zhu, and Bo Yu

Neural-Network-Driven Fuzzy Reasoning for Product Development Processes . . . 927
Yingkui Gu, Hongzhong Huang, and Yonghua Li

The Integration of the Neural Network and Computational Fluid Dynamics
for the Heatsink Design 933
Yeander Kuan and Hsinchung Lien

The Modeling and Application of Cost Predication Based on Neural Network 939
Xiaoling Huang, Jiansheng Xue, and Liju Dong

Combining SOM and Fuzzy Rule Base for Sale Forecasting
in Printed Circuit Board Industry 947
Pei-Chann Chang and K. Robert Lai

20 Other Applications

Improving Accuracy of Perceptron Predictor Through Correlating Data Values
in SMT Processors 955
Liqiang He and Zhiyong Liu

A Genetic-Algorithm-Based Neural Network Approach
for Short-Term Traffic Flow Forecasting 965
Mingzhe Liu, Ruili Wang, Jiansheng Wu, and Ray Kemp

Self-organizing Map Analysis Consistent with Neuroimaging for Chinese Noun, Verb and Class-Ambiguous Word 971
Minghu Jiang, Huiying Cai, and Bo Zhang

Self-organizing Map Analysis of Conceptual and Semantic Relations for Noun . . 977
Minghu Jiang, Chengqing Zong, and Beixing Deng

Artificial Neural Network for Prediction of Rockburst in Deep-Buried Long Tunnel 983
Xiaohong Li, Xinfei Wang, Yong Kang, and Zheng He

Implementation of Brillouin-Active Fiber Based Neural Network in Smart Structures 987
Yongkab Kim, Sunja Lim, Hwan Y. Kim, Sungkwun Oh, and Chung Yu

Inelastic Simulation of Insect Cuticle Using Artificial Neural Network 992
Bin Chen, Gang Chen, Hongtao Liu, Xianghe Peng, and Jinghong Fan

Applying Neural Networks and Geographical Information Systems to Airport Noise Evaluation 998
Yingjie Yang, David Gillingwater, and Chris Hinde

An Artificial Neural Network Method for Map Correction 1004
Yi Chai, Maoyun Guo, Shangfu Li, Zhifen Zhang, and Dalong Feng

An Effective Two-Stage Neural Network Model and Its Application on Flood Loss Prediction 1010
Li Yang, Chun Zuo, and Yuguo Wang

An Artificial Neural Network Model for Crop Yield Responding to Soil Parameters 1017
Gang Liu, Xuehong Yang, and Minzan Li

Research on Reservation Allocation Decision Method Based on Neural Network 1022
Ancheng Pan, Yongqing Yang, and Hanhui Hu

Wastewater BOD Forecasting Model for Optimal Operation Using Robust Time-Delay Neural Network 1028
Lijie Zhao and Tianyou Chai

A Split-Step PSO Algorithm in Prediction of Water Quality Pollution 1034
Kwokwing Chau

Long-Term Prediction of Discharges in Manwan Reservoir Using Artificial Neural Network Models 1040
Chuntian Cheng, Kwokwing Chau, Yingguang Sun, and Jianyi Lin

Application of Artificial Neural Networks to Predicate Shale Content 1046
Kesheng Wang, Resko Barna, Yi Wang, Maxim Boldin, and Ove R. Hjelmervik

Optimization of Forecasting Supply Chain Management Sustainable Collaboration Using Hybrid Artificial Neural Network 1052
Sehun Lim and Juhee Hahn

Multiple Criteria Inventory Classification Based on Principal Components Analysis and Neural Network 1058
Quansheng Lei, Jian Chen, and Qing Zhou

Author Index 1065

Table of Contents, Part I

1 Theoretical Analysis

Population Coding, Bayesian Inference and Information Geometry 1
Shun-ichi Amari

One-Bit-Matching ICA Theorem, Convex-Concave Programming, and Combinatorial Optimization 5
Lei Xu

Dynamic Models for Intention (Goal-Directedness) Are Required by Truly Intelligent Robots 21
Walter J. Freeman

Differences and Commonalities Between Connectionism and Symbolicism 34
Shoujue Wang and Yangyang Liu

Pointwise Approximation for Neural Networks 39
Feilong Cao, Zongben Xu, and Youmei Li

On the Universal Approximation Theorem of Fuzzy Neural Networks with Random Membership Function Parameters 45
Lipo Wang, Bing Liu, and Chunru Wan

A Review: Relationship Between Response Properties of Visual Neurons and Advances in Nonlinear Approximation Theory 51
Shan Tan, Xiuli Ma, Xiangrong Zhang, and Licheng Jiao

Image Representation in Visual Cortex and High Nonlinear Approximation 57
Shan Tan, Xiangrong Zhang, Shuang Wang, and Licheng Jiao

Generalization and Property Analysis of GENET 63
Youmei Li, Zongben Xu, and Feilong Cao

On Stochastic Neutral Neural Networks 69
Yumin Zhang, Lei Guo, Lingyao Wu, and Chunbo Feng

Eigenanalysis of CMAC Neural Network 75
Chunshu Zhang

A New Definition of Sensitivity for RBFNN and Its Applications to Feature Reduction 81
Xizhao Wang and Chunguo Li

Complexity of Error Hypersurfaces in Multilayer Perceptrons with General Multi-input and Multi-output Architecture 87
Xun Liang

Nonlinear Dynamical Analysis on Coupled Modified Fitzhugh-Nagumo Neuron Model 95
Deepak Mishra, Abhishek Yadav, Sudipta Ray, and Prem K. Kalra

Stability of Nonautonomous Recurrent Neural Networks with Time-Varying Delays 102
Haijun Jiang, Jinde Cao, and Zhidong Teng

Global Exponential Stability of Non-autonomous Neural Networks with Variable Delay 108
Minghui Jiang, Yi Shen, and Meiqin Liu

A Generalized LMI-Based Approach to the Global Exponential Stability of Recurrent Neural Networks with Delay 114
Yi Shen, Minghui Jiang, and Xiaoxin Liao

A Further Result for Exponential Stability of Neural Networks with Time-Varying Delays 120
Jun Zhang, Xiaofeng Liao, Chuandong Li, and Anwen Lu

Improved Results for Exponential Stability of Neural Networks with Time-Varying Delays 126
Deyin Wu, Qingyu Xiong, Chuandong Li, Zhong Zhang, and Haoyang Tang

Global Exponential Stability of Recurrent Neural Networks with Infinite Time-Varying Delays and Reaction-Diffusion Terms 132
Qiankun Song, Zhenjiang Zhao, and Xuedong Chen

Exponential Stability Analysis of Neural Networks with Multiple Time Delays . . 142
Huaguang Zhang, Zhanshan Wang, and Derong Liu

Exponential Stability of Cohen-Grossberg Neural Networks with Delays 149
Wei Zhang and Jianqiao Yu

Global Exponential Stability of Cohen-Grossberg Neural Networks with Time-Varying Delays and Continuously Distributed Delays 156
Yi Shen, Minghui Jiang, and Xiaoxin Liao

Exponential Stability of Stochastic Cohen-Grossberg Neural Networks with Time-Varying Delays 162
Xiaolin Li and Jinde Cao

Exponential Stability of Fuzzy Cellular Neural Networks with Unbounded Delay 168
Tingwen Huang and Linhua Zhang

Global Exponential Stability of Reaction-Diffusion Hopfield Neural Networks with Distributed Delays 174
Zhihong Tang, Yiping Luo, and Feiqi Deng

Global Exponential Stability of Delayed Impulsive Hopfield Type Neural Networks 181
Bingji Xu, Qun Wang, Yi Shen, and Xiaoxin Liao

Global Exponential Stability of Hopfield Neural Networks with Impulsive Effects 187
Zhichun Yang, Jinan Pei, Daoyi Xu, Yumei Huang, and Li Xiang

Global Exponential Stability of Discrete Time Hopfield Neural Networks with Delays 193
Qiang Zhang, Wenbing Liu, and Xiaopeng Wei

Stability Analysis of Uncertain Neural Networks with Linear and Nonlinear Time Delays 199
Hanlin He, Zhongsheng Wang, and Xiaoxin Liao

Robust Stability for Delayed Neural Networks with Nonlinear Perturbation 203
Li Xie, Tianming Liu, Jilin Liu, Weikang Gu, and Stephen Wong

Robust Stability Analysis of a Class of Hopfield Neural Networks with Multiple Delays 209
Huaguang Zhang, Ce Ji, and Derong Liu

Robust Stability of Interval Delayed Neural Networks 215
Wenlian Lu and Tianping Chen

Impulsive Robust Control of Interval Hopfield Neural Networks 222
Yinping Zhang and Jitao Sun

Global Attractivity of Cohen-Grossberg Model with Delays 229
Tao Xiang, Xiaofeng Liao, and Jian Huang

High-Order Hopfield Neural Networks 235
Yi Shen, Xiaojun Zong, and Minghui Jiang

Stability Analysis of Second Order Hopfield Neural Networks with Time Delays 241
Jinan Pei, Daoyi Xu, Zhichun Yang, and Wei Zhu

Convergence Analysis of Genetic Regulatory Networks Based on Nonlinear Measures 247
Hongtao Lu, Zhizhou Zhang, and Lin He

Stability Conditions for Discrete Neural Networks in Partial Simultaneous Updating Mode 253
Runnian Ma, Shengrui Zhang, and Sheping Lei

Dynamic Behavior Analysis of Discrete Neural Networks with Delay 259
Runnian Ma, Sheping Lei, and Shengrui Zhang

Existence and Stability of Periodic Solution
in a Class of Impulsive Neural Networks 265
Xiaofan Yang, David J. Evans, and Yuanyan Tang

Globally Attractive Periodic Solutions of Continuous-Time Neural Networks
and Their Discrete-Time Counterparts 271
Changyin Sun, Liangzhen Xia, and Chunbo Feng

Globally Stable Periodic State of Delayed Cohen-Grossberg Neural Networks . . 276
Chaojin Fu, Hanlin He, and Xiaoxin Liao

Globally Attractive Periodic State of Discrete-Time Cellular Neural Networks
with Time-Varying Delays 282
Zhigang Zeng, Boshan Chen, and Zengfu Wang

An Analysis for Periodic Solutions of High-Order BAM Neural Networks
with Delays 288
Jianlong Qiu and Jinde Cao

Periodic Oscillation and Exponential Stability
of a Class of Competitive Neural Networks 294
Boshan Chen

Synchronous Behaviors of Two Coupled Neurons 302
Ying Wu, Jianxue Xu, and Wuyin Jin

Adaptive Synchronization of Delayed Neural Networks
Based on Parameters Identification 308
Jin Zhou, Tianping Chen, and Lan Xiang

Strength and Direction of Phase Synchronization of Neural Networks 314
Yan Li, Xiaoli Li, Gaoxiang Ouyang, and Xinping Guan

Hopf Bifurcation in a Single Inertial Neuron Model:
A Frequency Domain Approach 320
Shaorong Li, Shaowen Li, Xipeng Sun, and Jie Li

Hopf Bifurcation in a Single Inertial Neuron Model with a Discrete Delay 327
Shaowen Li and Shaorong Li

Stability and Bifurcation of a Neuron Model with Delay-Dependent Parameters . 334
Xu Xu and Yanchun Liang

Stability and Chaos of a Neural Network with Uncertain Time Delays 340
Shangbo Zhou, Hua Li, and Zhongfu Wu

Chaotic Synchronization of Delayed Neural Networks 346
Fenghua Tu, Xiaofeng Liao, and Chuandong Li

Chaos Synchronization for Bi-directional Coupled Two-Neuron Systems with Discrete Delays 351
Xiaohong Zhang and Shangbo Zhou

Complex Dynamics in a Simple Hopfield-Type Neural Network 357
Qingdu Li and Xiaosong Yang

Adaptive Chaotic Controlling Method of a Chaotic Neural Network Model 363
Lidan Wang, Shukai Duan, and Guangyuan Liu

2 Model Design

Modeling Cortex Network: A Spatio-temporal Population Approach 369
Wentao Huang, Licheng Jiao, Maoguo Gong, and Chuang Guo

A Special Kind of Neural Networks: Continuous Piecewise Linear Functions . . . 375
Xusheng Sun and Shuning Wang

A Novel Dynamic Structural Neural Network with Neuron-Regeneration and Neuron-Degeneration Mechanisms 380
Yingtung Hsiao, Chenglong Chuang, Joeair Jiang, Chiang Wang, and Chengchih Chien

A New Adaptive Ridgelet Neural Network 385
Shuyuan Yang, Min Wang, and Licheng Jiao

Designing Neural Networks Using Hybrid Particle Swarm Optimization 391
Bo Liu, Ling Wang, Yihui Jin, and Dexian Huang

A New Strategy for Designing Bidirectional Associative Memories 398
Gengsheng Zheng, Sidney Nascimento Givigi, and Weiyu Zheng

Genetically Optimized Hybrid Fuzzy Neural Networks Based on TSK Fuzzy Rules and Polynomial Neurons 404
Sungkwun Oh, Byoungjun Park, and Hyunki Kim

Genetically Optimized Self-organizing Fuzzy Polynomial Neural Networks Based on Information Granulation 410
Hosung Park, Daehee Park, and Sungkwun Oh

Identification of ANFIS-Based Fuzzy Systems with the Aid of Genetic Optimization and Information Granulation 416
Sungkwun Oh, Keonjun Park, and Hyungsoo Hwang

Design of Rule-Based Neurofuzzy Networks by Means of Genetic Fuzzy Set-Based Granulation 422
Byoungjun Park and Sungkwun Oh

Design of Genetic Fuzzy Set-Based Polynomial Neural Networks
with the Aid of Information Granulation 428
Sungkwun Oh, Seokbeom Roh, and Yongkab Kim

A Novel Self-organizing Neural Fuzzy Network for Automatic Generation
of Fuzzy Inference Systems 434
Meng Joo Er and Rishikesh Parthasarathi

Constructive Fuzzy Neural Networks and Its Application 440
Lunwen Wang, Ying Tan, and Ling Zhang

A Novel CNN Template Design Method Based on GIM 446
Jianye Zhao, Hongling Meng, and Daoheng Yu

A Novel Generalized Congruence Neural Networks 455
Yong Chen, Guoyin Wang, Fan Jin, and Tianyun Yan

A SOM Based Model Combination Strategy 461
Cristofer Englund and Antanas Verikas

Typical Sample Selection and Redundancy Reduction
for Min-Max Modular Network with GZC Function 467
Jing Li, Baoliang Lu, and Michinori Ichikawa

Parallel Feedforward Process Neural Network
with Time-Varying Input and Output Functions 473
Shisheng Zhong, Gang Ding, and Daizhong Su

A Novel Solid Neuron-Network Chip
Based on Both Biological and Artificial Neural Network Theories 479
Zihong Liu, Zhihua Wang, Guolin Li, and Zhiping Yu

Associative Memory Using Nonlinear Line Attractor Network
for Multi-valued Pattern Association 485
Ming-Jung Seow and Vijayan K. Asari

Associative Chaotic Neural Network
via Exponential Decay Spatio-temporal Effect 491
Shukai Duan and Lidan Wang

On a Chaotic Neural Network with Decaying Chaotic Noise 497
Tianyi Ma, Ling Wang, Yingtao Jiang, and Xiaozong Yang

Extension Neural Network-Type 3 503
Manghui Wang

Pulsed Para-neural Networks (PPNN) Based on MEXORs and Counters 509
Junquan Li and Yixin Yin

Using Ensemble Information in Swarming Artificial Neural Networks 515
Jian Tang, Zengqi Sun, and Jihong Zhu

Negatively Correlated Neural Network Ensemble
with Multi-population Particle Swarm Optimization 520
Zheng Qin, Yu Liu, Xingchen Heng, and Xianhui Wang

Wrapper Approach for Learning Neural Network Ensemble
by Feature Selection 526
Haixia Chen, Senmiao Yuan, and Kai Jiang

Constructive Ensemble of RBF Neural Networks
and Its Application to Earthquake Prediction 532
Yue Liu, Yuan Li, Guozheng Li, Bofeng Zhang, and Genfeng Wu

3 Learning Methods

The Bounds on the Rate of Uniform Convergence for Learning Machine 538
Bin Zou, Luoqing Li, and Jie Xu

Supervised Learning on Local Tangent Space 546
Hongyu Li, Li Teng, Wenbin Chen, and I-Fan Shen

Study Markov Neural Network by Stochastic Graph 552
Yali Zhao, Guangcheng Xi, and Jianqiang Yi

An Efficient Recursive Total Least Squares Algorithm
for Training Multilayer Feedforward Neural Networks 558
Nakjin Choi, JunSeok Lim, and KoengMo Sung

A Robust Learning Algorithm for Feedforward Neural Networks
with Adaptive Spline Activation Function 566
Lingyun Hu and Zengqi Sun

A New Modified Hybrid Learning Algorithm for Feedforward Neural Networks . 572
Fei Han, Deshuang Huang, Yiuming Cheung, and Guangbin Huang

Robust Recursive TLS (Total Least Square) Method
Using Regularized UDU Decomposed
for FNN (Feedforward Neural Network) Training 578
JunSeok Lim, Nakjin Choi, and KoengMo Sung

An Improved Backpropagation Algorithm Using Absolute Error Function 585
Jiancheng Lv and Zhang Yi

An Improved Relative Criterion Using BP Algorithm 591
Zhiyong Zhang, Jingang Liu, and Zhongzhi Shi

Solving Hard Local Minima Problems Using Basin Cells
for Multilayer Perceptron Training 597
Younggui Yoon and Jaewook Lee

Enhanced Fuzzy Single Layer Perceptron 603
Kwangbaek Kim, Sungshin Kim, Younghoon Joo, and Am-Sok Oh

A New Training Algorithm for a Fuzzy Perceptron and Its Convergence 609
Jie Yang, Wei Wu, and Zhiqiong Shao

Stochastic Fuzzy Neural Network
and Its Robust Parameter Learning Algorithm 615
Junping Wang and Quanshi Chen

Applying Neural Network to Reinforcement Learning in Continuous Spaces 621
Dongli Wang, Yang Gao, and Pei Yang

Multiagent Reinforcement Learning Algorithm
Using Temporal Difference Error 627
SeungGwan Lee

A Foremost-Policy Reinforcement Learning Based ART2 Neural Network
and Its Learning Algorithm 634
Jian Fan and Gengfeng Wu

A Reinforcement Learning
Based Radial-Bassis Function Network Control System 640
Jianing Li, Jianqiang Yi, Dongbin Zhao, and Guangcheng Xi

Structure Pruning Strategies for Min-Max Modular Network 646
Yang Yang and Baoliang Lu

Sequential Bayesian Learning for Modular Neural Networks 652
Pan Wang, Zhun Fan, Youfeng Li, and Shan Feng

A Modified Genetic Algorithm for Fast Training Neural Networks 660
Dongsun Kim, Hyunsik Kim, and Duckjin Chung

Immunity Clonal Synergetic Learning of Unbalanced Attention Parameters
in Synergetic Network 666
Xiuli Ma and Licheng Jiao

Optimizing Weights of Neural Network
Using an Adaptive Tabu Search Approach 672
Yi He, Yuhui Qiu, Guangyuan Liu, and Kaiyou Lei

Semi-supervised Learning for Image Retrieval Using Support Vector Machines . 677
Ke Lu, Jidong Zhao, Mengqin Xia, and Jiazhi Zeng

A Simple Rule Extraction Method Using a Compact RBF Neural Network 682
Lipo Wang and Xiuju Fu

Automatic Fuzzy Rule Extraction Based on Fuzzy Neural Network 688
Li Xiao and Guangyuan Liu

4 Optimization Methods

Neural Networks for Nonconvex Nonlinear Programming Problems:
A Switching Control Approach 694
Changyin Sun and Chunbo Feng

Deterministic Global Optimization with a Neighbourhood
Determination Algorithm Based on Neural Networks 700
Weitao Sun, Jiwu Shu, and Weimin Zheng

A Neural Network Methodology of Quadratic Optimization
with Quadratic Equality Constraints 706
Yongqing Yang, Jinde Cao, and Daqi Zhu

A Hopfiled Neural Network for Nonlinear Constrained Optimization Problems
Based on Penalty Function 712
Zhiqing Meng and Chuangyin Dang

A Neural Network Algorithm for Second-Order Conic Programming 718
Xuewen Mu, Sanyang Liu, and Yaling Zhang

Application of Neural Network to Interactive Physical Programming 725
Hongzhong Huang and Zhigang Tian

Application of the "Winner Takes All" Principle
in Wang's Recurrent Neural Network for the Assignment Problem 731
Paulo Henrique Siqueira, Sergio Scheer, and Maria Teresinha Arns Steiner

Theoretical Analysis and Parameter Setting of Hopfield Neural Networks 739
Hong Qu, Zhang Yi, and XiaoLin Xiang

Solving Optimization Problems Based on Chaotic Neural Network
with Hysteretic Activation Function 745
Xiuhong Wang, Qingli Qiao, and Zhengqu Wang

An Improved Transiently Chaotic Neural Network
for Solving the K-Coloring Problem 750
Shenshen Gu

A Sweep-Based TCNN Algorithm for Capacity Vehicle Routing Problem 756
Huali Sun, Jianying Xie, and Yaofeng Xue

Transient Chaotic Discrete Neural Network for Flexible Job-Shop Scheduling . . 762
Xinli Xu, Qiu Guan, Wanliang Wang, and Shengyong Chen

Integration of Artificial Neural Networks and Genetic Algorithm for Job-Shop Scheduling Problem 770
Fuqing Zhao, Yi Hong, Dongmei Yu, Xuhui Chen, and Yahong Yang

An Effective Algorithm Based on GENET Neural Network Model for Job Shop Scheduling with Release Dates and Due Dates 776
Xin Feng, Hofung Leung, and Lixin Tang

Fuzzy Due Dates Job Shop Scheduling Problem Based on Neural Network 782
Yuan Xie, Jianying Xie, and Jie Li

Heuristic Combined Artificial Neural Networks to Schedule Hybrid Flow Shop with Sequence Dependent Setup Times 788
Lixin Tang and Yanyan Zhang

A Neural Network Based Heuristic for Resource-Constrained Project Scheduling 794
Yongyi Shou

Functional-Link Net Based Multiobjective Fuzzy Optimization 800
Ping Wang, Hongzhong Huang, Ming J. Zuo, Weidong Wu, and Chunsheng Liu

Optimizing the Distributed Network Monitoring Model with Bounded Bandwidth and Delay Constraints by Neural Networks 805
Xianghui Liu, Jianping Yin, Zhiping Cai, Xicheng Lu, and Shiming Chen

Stochastic Nash Equilibrium with a Numerical Solution Method 811
Jinwu Gao and Yankui Liu

5 Kernel Methods

Generalized Foley-Sammon Transform with Kernels 817
Zhenzhou Chen and Lei Li

Sparse Kernel Fisher Discriminant Analysis 824
Hongjie Xing, Yujiu Yang, Yong Wang, and Baogang Hu

Scaling the Kernel Function to Improve Performance of the Support Vector Machine 831
Peter Williams, Sheng Li, Jianfeng Feng, and Si Wu

Online Support Vector Machines with Vectors Sieving Method 837
Liangzhi Gan, Zonghai Sun, and Youxian Sun

Least Squares Support Vector Machine Based on Continuous Wavelet Kernel . . . 843
Xiangjun Wen, Yunze Cai, and Xiaoming Xu

Multiple Parameter Selection for LS-SVM Using Smooth Leave-One-Out Error . 851
Liefeng Bo, Ling Wang, and Licheng Jiao

Trajectory-Based Support Vector Multicategory Classifier 857
Daewon Lee and Jaewook Lee

Multi-category Classification by Least Squares Support Vector Regression 863
Jingqing Jiang, Chunguo Wu, and Yanchun Liang

Twi-Map Support Vector Machine for Multi-classification Problems 869
Zhifeng Hao, Bo Liu, Xiaowei Yang, Yanchun Liang, and Feng Zhao

Fuzzy Multi-class SVM Classifier Based on Optimal Directed Acyclic Graph
Using in Similar Handwritten Chinese Characters Recognition 875
Jun Feng, Yang Yang, and Jinsheng Fan

A Hierarchical and Parallel Method for Training Support Vector Machines 881
Yimin Wen and Baoliang Lu

Task Decomposition Using Geometric Relation for Min-Max Modular SVMs . . . 887
Kaian Wang, Hai Zhao, and Baoliang Lu

A Novel Ridgelet Kernel Regression Method 893
Shuyuan Yang, Min Wang, Licheng Jiao, and Qing Li

Designing Nonlinear Classifiers Through Minimizing VC Dimension Bound . . . 900
Jianhua Xu

A Cascaded Mixture SVM Classifier for Object Detection 906
Zejian Yuan, Nanning Zheng, and Yuehu Liu

Radar High Range Resolution Profiles Feature Extraction
Based on Kernel PCA and Kernel ICA 913
Hongwei Liu, Hongtao Su, and Zheng Bao

Controlling Chaotic Systems via Support Vector Machines
Without Analytical Model 919
Meiying Ye

Support Vector Regression for Software Reliability Growth Modeling
and Prediction 925
Fei Xing and Ping Guo

SVM-Based Semantic Text Categorization
for Large Scale Web Information Organization 931
Peng Fu, Deyun Zhang, Zhaofeng Ma, and Hao Dong

Fuzzy Support Vector Machine and Its Application to Mechanical Condition Monitoring 937
Zhousuo Zhang, Qiao Hu, and Zhengjia He

6 Component Analysis

Guided GA-ICA Algorithms 943
Juan Manuel Górriz, Carlos García Puntonet, Angel Manuel Gómez, and Oscar Pernía

A Cascaded Ensemble Learning for Independent Component Analysis 949
Jian Cheng, Kongqiao Wang, and Yenwei Chen

A Step by Step Optimization Approach to Independent Component Analysis . . . 955
Dengpan Gao, Jinwen Ma, and Qiansheng Cheng

Self-adaptive FastICA Based on Generalized Gaussian Model 961
Gang Wang, Xin Xu, and Dewen Hu

An Efficient Independent Component Analysis Algorithm for Sub-Gaussian Sources 967
Zhilin Zhang and Zhang Yi

ICA and Committee Machine-Based Algorithm for Cursor Control in a BCI System 973
Jianzhao Qin, Yuanqing Li, and Andrzej Cichocki

Fast Independent Component Analysis for Face Feature Extraction 979
Yiqiong Xu, Bicheng Li, and Bo Wang

Affine Invariant Descriptors for Color Images Based on Independent Component Analysis 985
Chengming Liu, Xuming Huang, and Liming Zhang

A New Image Protection and Authentication Technique Based on ICA 991
Linhua Zhang, Shaojiang Deng, and Xuebing Wang

Locally Spatiotemporal Saliency Representation: The Role of Independent Component Analysis 997
Tao Jiang and Xingzhou Jiang

A Multistage Decomposition Approach for Adaptive Principal Component Analysis 1004
Dazheng Feng

A New Kalman Filtering Algorithm for Nonlinear Principal Component Analysis 1010
Xiaolong Zhu, Xianda Zhang, and Ying Jia

An Improvement on PCA Algorithm for Face Recognition 1016
Vo Dinh Minh Nhat and Sungyoung Lee

A Modified PCA Neural Network to Blind Estimation of the PN Sequence in Lower SNR DS-SS Signals 1022
Tianqi Zhang, Xiaokang Lin, Zhengzhong Zhou, and Aiping Mu

A Modified MCA EXIN Algorithm and Its Convergence Analysis 1028
Dezhong Peng, Zhang Yi, and XiaoLin Xiang

Robust Beamforming by a Globally Convergent MCA Neural Network 1034
Mao Ye

Author Index 1043

Table of Contents, Part II

7 Pattern Analysis

A New Approach for Classification: Visual Simulation Point of View 1
Zongben Xu, Deyu Meng, and Wenfeng Jing

A Novel Classifier with the Immune-Training Based Wavelet Neural Network . . 8
Lei Wang, Yinling Nie, Weike Nie, and Licheng Jiao

Fisher Subspace Tree Classifier Based on Neural Networks 14
Dongyue Chen, Xiaodan Lu, and Liming Zhang

Classification Algorithms Based on Fisher Discriminant and Perceptron Neural Network 20
Hu Yang and Jianwen Xu

Robust Classification of Immunity Clonal Synergetic Network Inspired by Fuzzy Integral 26
Xiuli Ma, Shuang Wang, and Licheng Jiao

An Improved Optimal Pairwise Coupling Classifier 32
Roger Xu, Tao Qian, and Chiman Kwan

Improvement on Response Performance of Min-Max Modular Classifier by Symmetric Module Selection 39
Hai Zhao and Baoliang Lu

Principle for Outputs of Hidden Neurons in CC4 Network 45
Zhenya Zhang, Shuguang Zhang, Xufa Wang, Shuangping Chen, and Hongmei Cheng

Chunk Incremental LDA Computing on Data Streams 51
Shaoning Pang, Seiichi Ozawa, and Nikola Kasabov

A Novel Clustering Method Based on SVM 57
Jie Li, Xinbo Gao, and Licheng Jiao

Clustering High-Dimensional Data Using Growing SOM 63
Junlin Zhou and Yan Fu

A Novel Clustering Algorithm Based upon a SOFM Neural Network Family . . . 69
Junhao Wen, Kaiwen Meng, Hongyan Wu, and Zhongfu Wu

Advanced Visualization Techniques for Self-organizing Maps with Graph-Based Methods . . . 75
Georg Pölzlbauer, Andreas Rauber, and Michael Dittenbach

Selection of Optimal Features for Iris Recognition . . . 81
Hongying Gu, Zhiwen Gao, and Fei Wu

Application of Multi-weighted Neuron for Iris Recognition . . . 87
Wenming Cao, Jianhui Hu, Gang Xiao, and Shoujue Wang

Robust Precise Eye Location by Adaboost and SVM Techniques . . . 93
Xusheng Tang, Zongying Ou, Tieming Su, Haibo Sun, and Pengfei Zhao

Classification-Based Face Detection Using Compound Features . . . 99
Linlin Huang, Akinobu Shimizu, and Hidefumi Kobatake

Face Recognition Using RBF Neural Networks and Wavelet Transform . . . 105
Bicheng Li and Hujun Yin

Face Recognition Using Fisher Non-negative Matrix Factorization with Sparseness Constraints . . . 112
Xiaorong Pu, Zhang Yi, Ziming Zheng, Wei Zhou, and Mao Ye

Gabor Features-Based Classification Using SVM for Face Recognition . . . 118
Yixiong Liang, Weiguo Gong, Yingjun Pan, Weihong Li, and Zhenjiang Hu

An Experimental Evaluation of Linear and Kernel-Based Classifiers for Face Recognition . . . 124
Congde Lu, Taiyi Zhang, Wei Zhang, and Guang Yang

A Study on Illumination Invariant Face Recognition Methods Based on Multiple Eigenspaces . . . 131
Wujun Li, Chongjun Wang, Dianjiang Xu, Bin Luo, and Zhaojian Chen

Boosted Independent Features for Face Expression Recognition . . . 137
Lianghua He, Jianzhong Zhou, Die Hu, Cairong Zou, and Li Zhao

Intelligent Immigration Control System by Using Passport Recognition and Face Verification . . . 147
Kwangbaek Kim

Recognition of Finger Spelling of American Sign Language with Artificial Neural Network Using Position/Orientation Sensors and Data Glove . . 157
Cemil Oz and Ming C. Leu

Fingerprint Minutia Recognition with Fuzzy Neural Network . . . 165
Guang Yang, Daming Shi, and Chai Quek

Fingerprint Classification Based on Curvature Sampling and RBF Neural Networks . . . 171
Xuchu Wang, Jianwei Li, and Yanmin Niu

Palmprint Recognition Based on Translation Invariant Zernike Moments and Modular Neural Network 177
Yanlai Li, Kuanquan Wang, and David Zhang

Gait Recognition Using Independent Component Analysis 183
Jiwen Lu, Erhu Zhang, Zhigang Zhang, and Yanxue Xue

Nighttime Pedestrian Detection with a Normal Camera Using SVM Classifier . . 189
Qiming Tian, Hui Sun, Yupin Luo, and Dongcheng Hu

Signature Recognition and Verification with Artificial Neural Network Using Moment Invariant Method 195
Cemil Oz

Handwritten Digit Recognition with Kernel-Based LVQ Classifier in Input Space 203
Hairong Lv and Wenyuan Wang

Recognition of English Business Cards Using Enhanced Hybrid Network 209
Kwangbaek Kim, Jaehyun Cho, and Amsuk Oh

A Novel Approach for License Plate Recognition Using Subspace Projection and Probabilistic Neural Network 216
Yafeng Hu, Feng Zhu, and Xianda Zhang

Automatic Authentication Technique Based on Supervised ART-2 and Polynomial Spline Pyramid Algorithm 222
Ning Chen, Boqin Feng, Haixiao Wang, and Hao Zhang

Neural Network Based Online Feature Selection for Vehicle Tracking 226
Tie Liu, Nanning Zheng, and Hong Cheng

TextCC: New Feed Forward Neural Network for Classifying Documents Instantly 232
Zhenya Zhang, Shuguang Zhang, Enhong Chen, Xufa Wang, and Hongmei Cheng

A Neural Network Model for Hierarchical Multilingual Text Categorization 238
Rowena Chau, Chunghsing Yeh, and Kate A. Smith

Chinese Syntactic Category Disambiguation Using Support Vector Machines . . . 246
Lishuang Li, Lihua Li, Degen Huang, and Heping Song

A Clustering Algorithm for Chinese Text Based on SOM Neural Network and Density 251
Zhiqing Meng, Hongcan Zhu, Yihua Zhu, and Gengui Zhou

Automatic Caption Detection in Video Frames Based on Support Vector Machine 257
Jianfeng Xu and Shaofa Li

Selection of ICA Features for Texture Classification 262
Xiangyan Zeng, Yenwei Chen, Deborah van Alphen, and Zensho Nakao

Feature Selection and Fusion for Texture Classification 268
Shutao Li and Yaonan Wang

Scene Classification Using Adaptive Processing of Tree Representation of Rectangular-Shape Partition of Images 274
Wei Sun, Ken Lo, and Zheru Chi

Shape Recognition Based on Radial Basis Probabilistic Neural Network and Application to Plant Species Identification 281
Jixiang Du, Deshuang Huang, Xiaofeng Wang, and Xiao Gu

Image Recognition Using Synergetic Neural Network. 286
Shuiping Gou and Licheng Jiao

Content Based Retrieval and Classification of Cultural Relic Images 292
Na Wei, M. Emre Celebi, and Guohua Geng

Obscene Image Recognition Based on Model Matching and BWFNN 298
Xiaohua Liu, Zhezhou Yu, Libiao Zhang, Miao Liu, Chunguang Zhou, Chunxia Li, Catitang Sun, and Li Zhang

Classification of SAR Imagery Using Multiscale Self-organizing Network 304
Xianbin Wen

Mixture of Experts for Stellar Data Classification 310
Yugang Jiang and Ping Guo

A Neural Network Model for Extraction of Salient Contours 316
Qiling Tang, Nong Sang, and Tianxu Zhang

A Mechanism for Extracting Optical Virtual Contours of Discrete Dot Stimuli . . 321
Eunhwa Jeong and Keongho Hong

Using Self-organizing Map for Mental Tasks Classification in Brain-Computer Interface 327
Hailong Liu, Jue Wang, and Chongxun Zheng

Speech Recognition Using Stereo Vision Neural Networks with Competition and Cooperation 333
Sung-III Kim

Speech Recognition of Finite Words Based on Multi-weight Neural Network . . . 339
Yan Wu, Hongbo Wang, Mingxi Jin, and Shoujue Wang

Continuous Speech Research Based on Two-Weight Neural Network 345
Wenming Cao, Xiaoxia Pan, and Shoujue Wang

Two-Domain Feature Compensation for Robust Speech Recognition 351
Haifeng Shen, Gang Liu, Jun Guo, and Qunxia Li

On Kernel Discriminant Analyses Applied to Phoneme Classification 357
András Kocsor

Automatic News Audio Classification Based on Selective Ensemble SVMs 363
Bing Han, Xinbo Gao, and Hongbing Ji

A Compound Statistical Model Based Radar HRRP Target Recognition 369
Lan Du, Hongwei Liu, Zheng Bao, and Junying Zhang

A Radar Target Multi-feature Fusion Classifier
Based on Rough Neural Network 375
Yinshui Shi, Hongbing Ji, and Xinbo Gao

Automatic Digital Modulation Recognition Based on ART2A-DWNN 381
Zhilu Wu, Xuexia Wang, Cuiyan Liu, and Guanghui Ren

Recognition of Radiated Noises of Ships Using Auditory Features
and Support Vector Machines 387
Xinhua Zhang, Chunyu Kang, and Zhijun Xia

Feature Selection and Identification of Underground Nuclear Explosion
and Natural Earthquake Based on Gamma Test and BP Neural Network 393
Daizhi Liu, Xihai Li, and Bin Zhang

An Adaptive Neural Network Classifier for Tropical Cyclone Prediction
Using a Two-Layer Feature Selector 399
Bo Feng and James N.K. Liu

Feature Point Matching of Affine Model Images Using Hopfield Network 405
Jinsi Tian and Jianbo Su

8 System Modeling

Nonlinear System Modeling Using Wavelet Networks 411
Seda Postalcioglu and Yasar Becerikli

Robust Modeling for Nonlinear Dynamic Systems
Using a Neurofuzzy Approach with Iterative Optimization 418
Shirong Liu, Simon X. Yang, and Jinshou Yu

Modelling of Chaotic Systems with Recurrent Least Squares Support Vector
Machines Combined with Stationary Wavelet Transform 424
Jiancheng Sun, Lun Yu, Guang Yang, and Congde Lu

Adding Value to System Dynamics Modeling
by Using Artificial Neural Network 430
Changrui Ren, Yueting Chai, and Yi Liu

Least Squares Wavelet Support Vector Machines
for Nonlinear System Identification 436
Zhenhua Yu and Yuanli Cai

Wavelet Support Vector Machines and Its Application
for Nonlinear System Identification 442
Xiangjun Wen, Yunze Cai, and Xiaoming Xu

Comparative Assessment of Interval and Affine Arithmetic
in Neural Network State Prediction 448
Marcela Jamett and Gonzalo Acuña

Identification of Duffing's Equation with Dynamic Recurrent Neural Network . . 454
Shan Liang, Qin Zhu, and Mitsuaki Ishitobi

An Intelligent System for Dynamic System State Forecasting 460
Wilson Wang

9 Signal Processing

Sequential Extraction Algorithm for BSS Without Error Accumulation 466
Qiang Liu and Tianping Chen

A Learning Framework for Blind Source Separation
Using Generalized Eigenvalues 472
Hailin Liu and Yiuming Cheung

Post-nonlinear Blind Source Separation Using Neural Networks
with Sandwiched Structure 478
Chunhou Zheng, Deshuang Huang, Zhanli Sun, and Li Shang

A Novel Approach for Underdetermined Blind Sources Separation
in Frequency Domain 484
Ming Xiao, Shengli Xie, and Yuli Fu

A Neural Network Blind Separation Method Based on Special Frequency Bins . . 490
Anqing Zhang, Xuxiu Zhang, Tianshuang Qiu, and Xinhua Zhang

Application of Blind Source Separation to Time Delay Estimation
in Interference Environments 496
Gaoming Huang, Luxi Yang, and Zhenya He

Blind Identification and Deconvolution
for Noisy Two-Input Two-Output Channels 502
Yuanqing Li, Andrzej Cichocki, and Jianzhao Qin

A Novel Blind Deconvolution Method
for Single-Output Chaotic Convolution Mixed Signal 508
Xiefeng Cheng, Yong Zhang, Zhiquan Feng, Ju Liu, and Huibo Hu

Stability Analysis of Multichannel Blind Deconvolution 514
Bin Xia and Liqing Zhang

Joint Diagonalization of Power Spectral Density Matrices
for Blind Source Separation of Convolutive Mixtures 520
Tiemin Mei, Jiangtao Xi, Fuliang Yin, and Joe F. Chicharo

A Block-Adaptive Subspace Method Using Oblique Projections
for Blind Separation of Convolutive Mixtures 526
Chunyi Peng, Xianda Zhang, and Qutang Cai

FIR Convolutive BSS Based on Sparse Representation 532
Zhaoshui He, Shengli Xie, and Yuli Fu

Blind Separation Combined Frequency Invariant Beamforming and ICA for
Far-field Broadband Acoustic Signals 538
Qi Lv, Xianda Zhang, and Ying Jia

Blind Source Separation-Based Encryption of Images and Speeches 544
Qiuhua Lin, Fuliang Yin, and Hualou Liang

A Digital Audio Watermarking Scheme Based on Blind Source Separation 550
Xiaohong Ma, Chong Wang, Xiangping Cong, and Fuliang Yin

Lidar Signal Processing for Under-water Object Detection 556
Vikramjit Mitra, Chiajiu Wang, and Satarupa Banerjee

Ultra-wideband Nearfield Adaptive Beamforming
Based on a RBF Neural Network 562
Min Wang, Shuyuan Yang, and Shunjun Wu

Automatic Digital Modulation Recognition Using Support Vector Machines
and Genetic Algorithm 568
Jie Li, Jing Peng, Heng Chu, and Weile Zhu

A Unified Framework for Synthesis of Cosine-Modulated Filter Banks
and Corresponding Wavelets 574
Ying Tan

A Systematic Chaotic Noise Reduction Method Combining
with Neural Network 580
Min Han, Yuhua Liu, Jianhui Xi, and Zhiwei Shi

A New Speech Enhancement Method for Adverse Noise Environment 586
Xiaohong Ma, Yu Wang, Wenlong Liu, and Fuliang Yin

A Subband Adaptive Learning Algorithm for Microphone Array
Based Speech Enhancement 592
Dongxia Wang and Fuliang Yin

A Spiking Neuron Model of Auditory Neural Coding 598
Guoping Wang and Misha Pavel

Blind Feature Extraction for Time-Series Classification
Using Haar Wavelet Transform 605
Hui Zhang, Tubao Ho, and Wei Huang

Prediction of Time Sequence
Using Recurrent Compensatory Neuro-fuzzy Systems 611
ChiYung Lee and ChengJian Lin

Study of Nonlinear Multivariate Time Series Prediction
Based on Neural Networks 618
Min Han, Mingming Fan, and Jianhui Xi

Improving Ability of Passive Attacks of Chaotic Encryption
by Using Neural Network 624
Xin Yang, Xiyue Huang, and Hanmin Huang

Chosen-Plaintext Cryptanalysis of a Clipped-Neural-Network-Based
Chaotic Cipher 630
Chengqing Li, Shujun Li, Dan Zhang, and Guanrong Chen

A Novel Identity-Based Key Issuing Scheme
Based on Interacting Neural Network 637
Tieming Chen, Bo Chen, and Jiamei Cai

The Projection Pursuit Learning Network
for Nonlinear Time Series Modeling and Forecasting 643
Zheng Tian, Zi Jin, Fang He, and Wei Ling

10 Image Processing

A New Scheme for Blind Decomposition of Mixed Pixels
Based on Non-negative Matrix Factorization 651
Hao Zhou, Bin Wang, and Liming Zhang

Representing Image Matrices: Eigenimages Versus Eigenvectors 659
Daoqiang Zhang, Songcan Chen, and Jun Liu

A SIMD Neural Network Processor for Image Processing 665
Dongsun Kim, Hyunsik Kim, Hongsik Kim, Gunhee Han, and Duckjin Chung

MRF-MBNN: A Novel Neural Network Architecture for Image Processing 673
Nian Cai, Jie Yang, Kuanghu Hu, and Haitao Xiong

Using LM Artificial Neural Networks and η-Closest-Pixels
for Impulsive Noise Suppression from Highly Corrupted Images 679
Pınar Çivicioğlu

Two Novel Image Filters Based on Canonical Piecewise Linear Networks 685
Xusheng Sun, Shuning Wang, and Yuehong Wang

A New Effective and Powerful Image Segmentation Method 690
Yalin Miao, Xianglin Miao, Zhengzhong Bian, Kai Chen, and Gang Yu

A Novel Image Interpolator Based on Probabilistic Neural Network
with Shapeness/Smoothness Adaptation 698
Chinghan Chen and Shenghsien Hsieh

Contrast Enhancement for Image with Simulated Annealing Algorithm
and Wavelet Neural Network 707
Changjiang Zhang, Xiaodong Wang, and Haoran Zhang

Adaptive Constructive Neural Networks Using Hermite Polynomials
for Image Compression 713
Liying Ma and Khashayar Khorasani

Compression of Remote Sensing Images
Based on Ridgelet and Neural Network 723
Shuyuan Yang, Min Wang, and Licheng Jiao

The SAR Image Compression with Projection Pursuit Neural Networks 730
Jian Ji, Zheng Tian, Wei Lin, and Yanwei Ju

Image Restoration Using Hopfield Neural Network
Based on Total Variational Model 735
Hongying Zhang, Yadong Wu, and Qicong Peng

Pulse Coupled Neural Network Based Image Fusion 741
Min Li, Wei Cai, and Zheng Tan

A Novel Image Fusion Method Based on SGNN 747
Zheng Qin, Fumin Bao, and Aiguo Li

Multifocus Image Fusion Using Spatial Features and Support Vector Machine . . 753
Shutao Li and Yaonan Wang

A New Scheme for Fusion of Multispectral and Panchromatic Images
Based on Residual Error 759
Zhirong Ge, Bin Wang, and Liming Zhang

Binocular 3D Reconstruction Based on Neural Network 765
*Mingxing Lin, Yongrui Zhao, Zhiguang Guan, Fenghua Ding,
Qingxin Xu, and Xiaohua Wang*

A Neural Network Based Lossless Digital Image Watermarking
in the Spatial Domain 772
Jun Sang and Mohammad S. Alam

A Copy Attack Resilient Blind Watermarking Algorithm
Based on Independent Feature Components 777
Ju Liu, Huibo Hu, Jiande Sun, and Yu Huang

Watermarking Capacity Analysis Based on Neural Network 783
Fan Zhang and Hongbin Zhang

SVR-Based Oblivious Watermarking Scheme 789
Yonggang Fu, Ruimin Shen, Hongtao Lu, and Xusheng Lei

An Audio Watermarking Scheme with Neural Network 795
Chong Wang, Xiaohong Ma, Xiangping Cong, and Fuliang Yin

Subsampling-Based Robust Watermarking Using Neural Network Detector 801
Wei Lu, Hongtao Lu, and FuLai Chung

Image Feature Extraction Based on an Extended Non-negative Sparse
Coding Neural Network Model 807
Li Shang, Deshuang Huang, Chunhou Zheng, and Zhanli Sun

Evolving Optimal Feature Set by Interactive Reinforcement Learning
for Image Retrieval 813
Jianbo Su, Fang Liu, and Zhiwei Luo

Perception-Oriented Prominent Region Detection in Video Sequences
Using Fuzzy Inference Neural Network 819
Congyan Lang, De Xu, Xu Yang, Yiwei Jiang, and Wengang Cheng

The Application of Neural Network
and Wavelet in Human Face Illumination Compensation 828
Zhongbo Zhang, Siliang Ma, and Danyang Wu

Global Icons and Local Icons of Images Based Unit-Linking PCNN
and Their Application to Robot Navigation 836
Xiaodong Gu and Liming Zhang

A Neural Model for Extracting Occluding Subjective Surfaces 842
Keongho Hong and Eunhwa Jeong

Hopfield Neural Network Image Matching Based on Hausdorff Distance
and Chaos Optimizing 848
Zhenghao Shi, Yaning Feng, Linhua Zhang, and Shitan Huang

Neural Network Based Fairing of Digitized Curves and Its Application 854
Jianhua Hou, Zongying Ou, and Mingen Guo

A Digital Image Encryption Scheme Based on the Hybrid
of Cellular Neural Network and Logistic Map 860
Wei Zhang, Jun Peng, Huaqian Yang, and Pengcheng Wei

Image Encryption Scheme Based on Chaotic Neural System 868
Shaojiang Deng, Linhua Zhang, and Di Xiao

11 Financial Analysis

Effectiveness of Different Target Coding Schemes on Networks in Financial Engineering 873
Kidong Lee, Junghee Park, and Sangjae Lee

Select the Size of Training Set for Financial Forecasting with Neural Networks . 879
Wei Huang, Yoshiteru Nakamori, Shouyang Wang, and Hui Zhang

Estimating the Yield Curve Using Calibrated Radial Basis Function Networks . . 885
Gyusik Han, Daewon Lee, and Jaewook Lee

Fast ICA for Online Cashflow Analysis 891
Shangming Yang and Zhang Yi

Impacts of Internet Stock News on Stock Markets Based on Neural Networks . . . 897
Xun Liang

Coherent Risk Measure Using Feedfoward Neural Networks 904
Hyoseok Lee, Jaewook Lee, Younggui Yoon, and Sooyoung Kim

Application of Evidence Theory and Neural Network in Warning System of Financial Risk 910
Qingyu Xiong, Yinlin Huang, Shan Liang, Weiren Shi, Songsong Tan, and Yinhua Lin

Novel Questionnaire-Responded Transaction Approach with SVM for Credit Card Fraud Detection 916
Rongchang Chen, Tungshou Chen, Yuer Chien, and Yuru Yang

Learning of Neural Networks for Fraud Detection Based on a Partial Area Under Curve 922
Lae-Jeong Park

Customer Churning Prediction Using Support Vector Machines in Online Auto Insurance Service 928
Yeon Hur and Sehun Lim

Author Index 935

NN-Based Iterative Learning Control Under Resource Constraints: A Feedback Scheduling Approach

Feng Xia and Youxian Sun

National Laboratory of Industrial Control Technology,
Institute of Modern Control Engineering
Zhejiang University, Hangzhou 310027, China
{xia,yxsun}@iipc.zju.edu.cn

Abstract. The problem of neural network based iterative learning control (NNILC) in a resource-constrained environment with workload uncertainty is examined from the real-time implementation perspective. Thanks to the iterative nature of the NNILC algorithm, it is possible to abort the optimization routine before it reaches the optimum. Taking into account the impact of resource constraints, a feedback scheduling approach is suggested, with the primary goal of maximize the control performance. The execution time of the NNILC task is dynamically adjusted to achieve a desired CPU utilization level. Thus a tradeoff is done between the available CPU time and the control performance. For the sake of easy implementation, a practical solution with a dynamic iteration stop criterion is proposed. Preliminary simulation results argue that the proposed approach is efficient and delivers better performance in the face of workload variations than the traditional NNILC algorithm.

1 Introduction

Neural network based iterative learning control (NNILC) has attracted significant attention in many control applications thanks to the successful combination of neural networks and iterative learning control. As perfect nonlinear function approximators, neural networks have good potential for identification and control applications. Many works [1-4] argue that ILC based on neural networks delivers wonderful performance e.g. for nonlinear system trajectory tracking. However, little has been done to develop efficient real-time implementations of NNILC. The control community generally assumes that the real-time platform used to implement the control algorithms can provide deterministic timing behaviors (e.g. constant execution time) and sufficient computing resources (CPU time) as needed. It is quite clear that this is not always the truth, especially in an embedded control system that features resource constraints and workload uncertainty [5]. It is well known that NNILC searches an "optimal" control input based on iterations. Intuitively more iterations lead to higher accuracy of the solution. This can be held under the conditions that the computing resources are sufficient all the time and the impact of computation delays is neglectable. All works mentioned above are done under this assumption. Under resource constraints, however, things will not go in this way. Increase in iterations may cause worse performance when the computing resources are scarce. On the other hand, if the CPU time is sufficient, it will be a waste of resource to computing control input with a small number of iterations that can only result in a sub-optimal solution.

For the first time in the literature, we consider the impact of computing resource constraints on NNILC. Instead of developing fast learning algorithms, we intend to propose a feedback scheduling approach to optimize the control performance of an iterative controller with resource constraints, from the real-time implementation perspective. Feedback scheduling is an emerging technique for integrating control and scheduling. The goal is to maximize the performance of NNILC in the face of resource constraints and workload uncertainty through controlling the CPU utilization. Using feedback scheduling, this can be achieved by manipulating the sampling period [6] or execution time [7] of the control task (which implements the NNILC algorithm). Thanks to the iterative nature of NNILC, the execution time is chosen to be the manipulated variable to control the CPU demand. Since the quality of control signal is gradually refined for each iteration in NNILC, it is possible to abort the optimization routine before it has reached the optimum. In this way, a tradeoff is done between the available computation time, i.e., how long time the controller may spend calculating the new control signal, and the control performance.

The remainder of this paper is structured as follows. A mathematical description of NNILC is given in Section 2. We present the feedback scheduling approach in Section 3. And its performance is evaluated via preliminary simulation studies in Section 4. Section 5 concludes this paper.

2 Formulation of NNILC

In the context of NNILC, a neural network, e.g., feedforward or Elman NN, is trained to model the dynamics of the plant. Arbitrary accuracy can be guaranteed since a NN is capable of approximating any continuous nonlinear functions. And then, certain control laws in an iterative learning manner can be built upon the NN model. Over the years, different architectures and learning algorithms have been suggested for NNILC [2]. For the sake of simplicity of description, we employ the most general structure and iterative learning algorithm as follows (see [8,9] for more details).

Consider a SISO (single-input and single-output) nonlinear system described as:

$$y(k+1) = f[y(k),...,y(k-n+1),u(k),...,u(k-m+1)] \tag{1}$$

where $y(k) \in R^n$ and $u(k) \in R^m$ are the system output and input respectively, $f(\bullet)$ is an unknown nonlinear function. The reference model is given by:

$$y_m(k+1) = A_m y_m(k) + B_m u_m(k) \tag{2}$$

where $A_m(z) = a_0 + a_1 z^{-1} + ... + a_{n-1} z^{n-1}$, $B_m(z) = b_0 + b_1 z^{-1} + ... + b_{m-1} z^{m-1}$. For the output tracking problem, the control objective is to find a control input $u(k)$ that satisfy $\lim_{k\to\infty} | y_m(k) - y(k) |_{u(k)} = 0$. Let $u_m(k)$ denote the reference input, and the NN to approximate the SISO system is described as $y_n(k+1) = NN_f(W, X(k))$. The NNILC operates as follows.

0) Train the NN model offline to provide good approximation of the plant dynamics. Initialize the system with the upper bound of the number of iterations I_{max} and the allowed maximums control error E_{max}.
1) At any instant k, let the iteration $i = 0$, and sample $y(k)$.
2) Calculate $u_m(k)$ and $y_m(k+1)$.
3) Calculate $u(k, i)$ according to

$$\begin{aligned} &u(k,i+1) = u(k,i) + \lambda[y_m(k+1) - NN_f(k,i)] \times [NN'_f(k,i)]^{-1} \\ &u(k,0) = u(k-1) \end{aligned} \tag{3}$$

where $\lambda > 0$ is the iterative learning factor, and $NN'_f(k,t) = \partial y_n / \partial u$.

4) For the updated input $u(k, i+1)$, evaluate $y_n(k+1, i+1)$ via the NN model.
5) Let $i = i + 1$. If $i > I_{max}$, go to next step. Otherwise, if $\| y_n(k+1, i+1) - y_m(k+1) \| > E_{max}$, return to step 3).
6) Issue the control input onto the plant, and wait for the next instant $k = k + 1$.

It can be seen from the above mathematical description that traditional NNILC utilizes the same maximum number of iterations for each sampling interval, regardless of the availability of computing resources. In a resource sufficient environment, good performance may be achieved, e.g., with an adequately large I_{max}. For a resource-constrained environment with time-varying workloads, however, such a prespecified maximum iterations may result in bad performance even if it is well-chosen, which can be seen from our simulation study in Section 4. The reason behind this is that traditional NNILC says little about the real-time implementation, treating control and scheduling separately.

3 A Feedback Scheduling Approach

Under resource constraints, the main goal of a feedback scheduler is to maximize the control performance through dynamically adjusting the execution time of the NN based iterative controller. From the control perspective, the scheduler can be viewed as a controller. The controlled variable is the CPU utilization while the execution time of the control task acts as the manipulated variable. Hence, the objective of the feedback scheduling can be formulated as

$$\min_C \ J = \eta(C) \quad \text{subject to} \ \ C/h \le U_{sp} \tag{4}$$

where C and h are the execution time and sampling period of the iterative controller, J is the cost function with respect to C, and U_{sp} is the available CPU utilization that changes over time. Under NNILC, it is typically that $\eta(\bullet)$ is a monotonic decreasing function, i.e., more iterations lead to better control performance in terms of less costs. This is mainly because that the output accuracy of NNILC is progressively increased with respect to iterations. Therefore, for a given sampling period, the control cost will be simply minimized by setting C to be h^*U_{sp}. However, this is not so easy as it seems due to the impact of computational delay on the control performance, especially for a resource constrained controller featuring workload variations. It is well known in the control community that time delays degrade the performance, even causing instability. With the execution time increasing, the controller performs worse. While the cost decreases with each new iteration, it is also possible that the resulting cost will increase because of larger latency.

For a specified system and a well-trained NN model, the execution time for each iteration is approximately constant (which is denoted by C_0), and hence the execution time of a control task only depends on the number of iterations. Consequently, the problem becomes determining the number of iterations so that the cost is minimized when experiencing workload variations. It is ideal that the decrease in cost thanks to iterations and the increase in cost due to latency for a given NNILC task should be computed analytically. In such case, in order to achieve the maximum control performance, the iteration should stop exactly when the total cost starts to increase instead of decrease. As far as the real-time implementation concerned, however, calculating the delay-induced cost turns out to be a very difficult problem, because of the complex and nonlinear nature of the NNILC. Inspired by the work in [7], we turn to a simpler stop criterion based on the formulation of feedback scheduling in (4). A maximum allowed delay is specified according to (4) as $C_{max} = h*U_{sp}$. Using the feedback scheduling approach, the pseudo-code for the NNILC in each sampling interval is given as follows.

```
Dynamic Iteration Stop Algorithm ()
  t = currentTime;
  repeat
    perform one optimization iteration;
    computationalDelay = currentTime - t;
  until computationalDelay >= $C_{max}$ - $\epsilon$
```

If the execution time is indirectly determined by stopping iterations as long as it exceeds the maximum allowed delay, the utilization limitation in (4) will be violated in most cases. This may degrade the performance of other tasks that concurrently run on the same CPU since their CPU time is stolen. In order to address this problem, a delay margin ϵ is introduced in the above algorithm. The typical value of ϵ falls between zero and C_0. Thus the absolute deviation between the actual and optimal execution time will be no more than max{ϵ, $C_0 - \epsilon$ }, which is considerably small (always less than C_0). Although occasional overruns may still exist, it is too slight to actually impact the control performance.

4 Example

In this section, we present preliminary simulations of the proposed approach. Consider the following nonlinear system [1] to be controlled using NNILC:

$$y(k+1) = \frac{y(k)}{1+y^2(k)} + u^3(k) \tag{5}$$

The reference model is described by

$$y_m(k+1) = 0.5y_m(k) + 0.5u_m(k) \tag{6}$$

The reference input $u_m(k)$ is given as a square wave input with a frequency of 0.5 rad/s and amplitude of 1. The neural network used to identify the system dynamics is of Elman type with $1\times6\times1$ nodes. Each layer's weights and biases are initialized with the Nguyen-Widrow layer initialization method. The Elman network is trained pre-runtime, providing a good approximation of the controlled system. The NNILC task is implemented as a real-time task, which is released periodically with a sampling period of 10ms. It is assumed that the underlying system schedules the task in a way that a new instance is not allowed to start execution until the previous instance has completed. For all simulations, the allowed CPU utilization U_{sp} changes from 0.6 to 0.5 at time t = 1s, and from 0.5 to 0.9 at time t = 2s.

In the first simulation, the system is controlled by the traditional NNILC algorithm given in Section 2. Related parameters are chosen as $\lambda = 0.02$, I_{max} = 30, and E_{max} = 0.05. The resulting control performance is shown in Fig.1. The control task works in the same way along the whole simulation process, regardless of the variations in computing resource availability. As a consequence of the prolonged computational delay, the performance degrades at t = 1s.

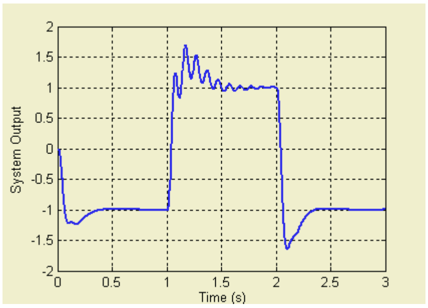


Fig. 1. Control performance under traditional NNILC

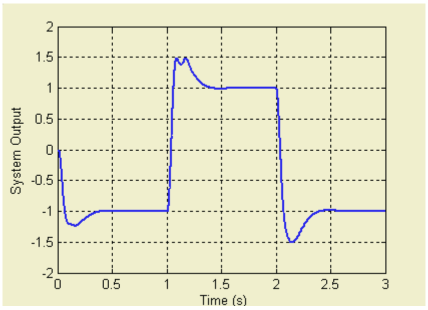


Fig. 2. Control performance under NNILC with the proposed approach

Next, the feedback scheduling approach we propose in Section 3 is implemented. The feedback scheduler acts as a periodic task with a period of 200 ms and the scheduling overhead is ignored since it is relatively small. As shown in Fig.2, the control performance is significantly improved, especially when the available computing resources become scare (from t = 1 to 2s). Note that NNILC with the feedback scheduling approach does not need the parameter I_{max} anymore. According to dynamic stop criterion in Section 3, the feedback scheduler may choose to terminate the execution of the iterative algorithm in order to avoid serious overruns in the face of heavy workload as well as to maximize the use of resource in the face of light workload. Note that small U_{sp} implies heavy workload on the CPU and large U_{sp} indicates the light case. The results show that the proposed approach to NNILC is efficient in the face of workload variations and outperforms the traditional NNILC algorithm, both when the workload increases and decreases.

5 Conclusions

From the viewpoint of real-time implementation, this paper presents a novel approach to NNILC with resource constraints. The emerging technique in the field of integrated control and scheduling, feedback scheduling, is exploited. A dynamic stop criterion for iteration routine is proposed to obtain online tradeoffs between the execution time and control performance. It is found that better performance can be achieved through dynamically terminating the execution of iterative control algorithms under resource constraints and workload uncertainty. Inspired by the promising results the feedback scheduling approach delivers, we are now working on applying it in multitasking cases where more than one NNILC task resides in the same CPU.

Acknowledgement

This work is supported by National Natural Science Foundation of China (NSFC 60434030, 60304018), National Basic Research Program of China (2002CB312200) and Advance Research Program of France-China (PRA SI03-02).

References

1. Narendra, K.S., Parthasarathy, K.: Identification and Control of Dynamical Systems Using Neural Networks. IEEE Trans. on Neural Networks, **1** (1990) 4-27
2. Chow, T.W.S., Li, X.D., Fang, Y.: A Real-Time Learning Control Approach for Nonlinear Continuous-Time System Using Recurrent Neural Networks. IEEE Trans. on Industrial Electronics, **47** (2000) 478-486
3. Xiong, Z., Zhang, J.: A Batch-to-batch iterative Optimal Control Strategy Based on Recurrent Neural Network Models. Journal of Process Control, **15** (2005) 11–21
4. Wang, F., and Li, M.: Optimal Iterative Control of Unknown Nonlinear Systems Using Neural Networks. Proc. 36th CDC, San Diego, California USA (1997) 2201-2206
5. Årzen, K-E., et al: Integrated Control and Scheduling. Research report: ISSN 0820-5316, Lund Institute of Technology (1999)
6. Cervin, A., Eker, J., Bernhardsson, B., and Årzén, K-E.: Feedback-Feedforward Scheduling of Control Tasks, Real-Time Systems, **23** (2002)
7. Henriksson, D, Cervin, A., Åkesson, J., Årzén, K-E.: Feedback Scheduling of Model Predictive Controllers. In Proc. 8th IEEE RTAS, San Jose, CA (2002) 207-216
8. Xu, J.X., Tan, Y.: Linear and Nonlinear Iterative Learning Control. In: Lecture Notes in Control and Information Sciences 291 (Edited by M. Thoma). Springer-Verlag, Heidelberg (2003)
9. He, Y., Li, X.: Neural Network Control Technology and Its Applications. Beijing: Science Press (2000)

Sequential Support Vector Machine Control of Nonlinear Systems by State Feedback

Zonghai Sun[1], Youxian Sun[1], Xuhua Yang[2], and Yongqiang Wang[1]

[1] National Laboratory of Industrial Control Technology,
Zhejiang University, Hangzhou 310027, China
zhsun@iipc.zju.edu.cn

[2] College of Information Engineering,
Zhejiang University of Technology,Hangzhou 310032, China

Abstract. Support vector machine is a new and promising technique for pattern classification and regression estimation. The training of support vector machine is characterized by a convex optimization problem, which involves the determination of a few additional tuning parameters. Moreover, the model complexity follows from that of this convex optimization problem. In this paper we introduce the sequential support vector machine for the regression estimation. The support vector machine is trained by the Kalman filter and particle filter respectively and then we design a controller based on the sequential support vector machine. Support vector machine controller is designed in the state feedback control of nonaffine nonlinear systems. The results of simulation demonstrate that the sequential training algorithms of support vector machine are effective and sequential support vector machine controller can achieve a satisfactory performance.

1 Introduction

Recently, controller design for systems having complex nonlinear dynamics becomes an important and challenging research area. Many remarkable results in this area have been obtained owing to the advances in geometric nonlinear control theory. Applications of these approaches are quite limited because they rely on the exact knowledge of the plant nonlinearities. In order to relax some of the exact model restrictions, several adaptive schemes have recently been introduced to solve the problem of parametric uncertainties. At present stage the existing controller are usually given locally and/or require additional prior knowledge about systems. Other problems of current adaptive control techniques such as nonlinear control laws which are difficult to derive, geometrically increasing complexity with the number of unknown parameters and general difficulty for real time applications have compelled researcher to look for more applicable methods.

Support vector machine (SVM) [1] is an important methodology in the area of neural and nonlinear modeling. The theory of SVM is based on the idea of structural risk minimization (SRM). SVM maps input data into high dimensional feature space in which an optimal separating hyperplane is built. In many applications it provides high generalization ability and overcomes the overfitting problem experienced by the other learning technique such as neural network. The training of SVM is done by quadratic programming (QP) possessing a global solution, which overcomes the problem of local minima suffered by classical neural network. An interesting and important property of SVM's solutions is that one obtains a sparse approximation, in the sense that many elements in the QP solution vector are equal to zeros.

In this paper, we discuss the SVM controller design of nonaffine nonlinear systems by state feedback. We adopt a sequential strategy for SVM training. Thus each item of data is used only once and then discarded. In the classic state-space model estimation by a Kalman filter setting, one defines a Gaussian probability density over the parameters to be estimated. The mean and covariance matrix are propagated using recursive update equations each time new data are received. The use of the matrix inversion lemma allows an elegant update of the inverse covariance matrix without actually having to invert a matrix for each sample. This is a widely studied topic, optimal in the case of linear Gaussian models. For nonlinear models the common trick is Taylor series expansion around the operating point, leading to the extended Kalman filter (EKF) [2], [3].

But Taylor series expansion leading to the EKF makes gross simplification of the probabilistic specification of the model. With the assumption of Gaussian probability density over the parameters to be estimated, the solutions of sequential SVM are not optimal. Particle filter [4],[5],[6],[7] techniques that provide a class of algorithms to address this issue. It is well suitable for applications involving on-line, nonlinear, and nongaussian signal processing. In this paper we focus on SVM, constraining ourselves to sequential training and its application in control [8].

2 The Presentation of State Space for SVM

Suppose we have a set of training samples $\{x_i, y_i\}_{i=1}^N$, where $x_i \in R^n$, $y_i \in R$. For a integer i, the following regression model defines the relation between x_i and y_i,

$$y_i = w^T \varphi(x_i) + b, i = 1, \cdots, N \tag{1}$$

where b denotes the threshold value, and w denotes the weight. In order to estimate the y_i for x_i, we should solve the following optimization problem

$$\min \quad L = \frac{1}{2} w^T w + C \sum_{i=1}^{N} (\xi_i + \xi_i^*) \tag{2}$$

s.t.

$$\begin{cases} y_i - w^T \varphi(x_i) - b \le \varepsilon + \xi_i^*, & i = 1, \cdots, N \\ w^T \varphi(x_i) + b - y_i \le \varepsilon + \xi_i, & i = 1, \cdots, N \\ \xi_i, \xi_i^* \ge 0, & i = 1, \cdots, N \end{cases} \tag{3}$$

where C denotes balance term, and ξ_i, ξ_i^* denote the slack variables.

By forming the Lagrangian, optimization problem (2), (3) can be translated into a problem of minimizing

$$\begin{aligned} \min L = {} & \frac{1}{2} w^T w + C \sum_{i=1}^{N} (\xi_i + \xi_i^*) - \sum_{i=1}^{N} (\gamma_i \xi_i + \gamma_i^* \xi_i^*) - \sum_{i=1}^{N} \alpha_i (y_i - w^T \varphi(x_i) - b + \varepsilon + \xi_i) \\ & - \sum_{i=1}^{N} \alpha_i^* (w^T \varphi(x_i) + b - y_i + \varepsilon + \xi_i^*) \end{aligned} \tag{4}$$

with respect to w, ξ_i, ξ_i^* and b. Setting the derivative of L to zeros gives:

$$\begin{cases} w=\sum_{i=1}^{N}(\alpha_i^*-\alpha_i)\varphi(x_i), & i=1,\cdots,N \\ C=\alpha_i+\gamma_i, & i=1,\cdots,N \\ C=\alpha_i^*+\gamma_i^*, & i=1,\cdots,N \\ \sum_{i=1}^{N}\alpha_i=\sum_{i=1}^{N}\alpha_i^* \end{cases} \tag{5}$$

The α_i, α_i^*, γ_i, γ_i^* are the Lagrange multipliers, there are two Lagrange multipliers for each training sample.

When a new data sample arrives, the output of SVM is given by:

$$\begin{aligned} y_i &= \sum_{j=1}^{N}(\alpha_j^*-\alpha_j)\varphi(x_j)^T\varphi(x_i)+b \\ &= \sum_{j=1}^{N}(\alpha_j^*-\alpha_j)K(x_i,x_j)+b, \quad i=1,\cdots,N \end{aligned} \tag{6}$$

where $K(x_i,x_j)$ denotes the kernel function which may be linear, polynomial, radial basis function and others that satisfy Mercer's condition [1]. We will discuss the sequential SVM paradigm for regression estimation. We re-express equation (6) in terms of a moving window over the data and compute the output of SVM each time new data are received:

$$y_{k+1}=\sum_{j=0}^{L}(\alpha_{k+1,k-L+j}^*-\alpha_{k+1,k-L+j})K(x_{k+1},x_{k-L+j})+b_{k+1} \tag{7}$$

We re-write equation (7) in an equivalent form:

$$y_{k+1}=[1\ K(x_{k+1},x_{k-L})\cdots K(x_{k+1},x_k)]\bullet\begin{bmatrix} b_{k+1} \\ \alpha_{k+1,0}^*-\alpha_{k+1,0} \\ \vdots \\ \alpha_{k+1,L}^*-\alpha_{k+1,L} \end{bmatrix} \tag{8}$$

In order to estimate b, $\alpha^*-\alpha$ sequential, we adopt following state-space Markovian representation:

$$\theta_{k+1}=\theta_k+\eta_k \tag{9}$$

$$y_{k+1}=C_{k+1}\theta_{k+1}+\delta_{k+1} \tag{10}$$

where $C_{k+1}=[1\ K(x_{k+1},x_{k-L})\ K(x_{k+1},x_{k-L+1})\cdots K(x_{k+1},x_k)]$, $\theta_k=\left[b_k\ \alpha_{k,0}^*-\alpha_{k,0}\ \alpha_{k,1}^*-\alpha_{k,1}\cdots\alpha_{k,L}^*-\alpha_{k,L}\right]^T$, η_k and δ_{k+1} denote the process and measurement noise respectively.

3 The Kalman Filtering Realization of SVM

We assume η_k to be zero mean Gassian distributed with covariance R and δ_{k+1} to be zero mean Gaussian distributed with covariance Q. The assumption is motivated by the fact that if we have a linear Gaussian state space model, the optimal solution (minimum variance, maximum a posteriori et al) is given by the Kalman filter [3].

The parameter θ may be estimated recursively by the Kalman filter

$$G_{k+1} = (P_k + R)C_{k+1}^T [Q + C_{k+1}(P_k + R)C_{k+1}^T]^{-1} \tag{11}$$

$$\hat{\theta}_{k+1} = \hat{\theta}_k + G_{k+1}(y_{k+1} - C_{k+1}\hat{\theta}_k) \tag{12}$$

$$P_{k+1} = (I - G_{k+1}C_{k+1})(P_k + R) \tag{13}$$

where P_k and $\hat{\theta}$ are known as the covariance and mean of θ, G_{k+1} denotes the Kalman gain and I denotes identity matrix.

4 The Particle Filter Realization of SVM

We assume η_k to be non-Gaussian distributed with covariance R and δ_{k+1} to be non-Gaussian distributed with covariance Q. For this type distribution of noise the parameter θ may be estimated recursively by the particle filter.

The method of particle filter [10] provides an approximative solution for the problem of estimating recursively the posterior density function $p(\theta_k \mid y_{k-L:k})$, for the discrete time system on the form (9), (10). In other words it provides an approximate solution to the optimal recursive Bayesian filter given by

$$p(\theta_{k+1} \mid y_{k-L:k}) = \int p(\theta_{k+1} \mid \theta_k) p(\theta_k \mid y_{k-L:k}) d\theta_k \tag{14}$$

$$p(\theta_k \mid y_{k-L:k}) = p(y_k \mid \theta_k) p(\theta_k \mid y_{k-L:k-1}) / p(y_k \mid y_{k-L:k-1}) \tag{15}$$

For expression on $p(\theta_{k+1} \mid \theta_k)$ and $p(y_k \mid \theta_k)$ in (14) and (15) we use the known probability densities p_{η_k} and p_{δ_k}. The main idea is to approximate the (15) with a sum of delta-Dirac functions located in the samples, $\theta_k^{(i)}$. Using the importance weights the posterior can be written as

$$p(\theta_k \mid y_{k-L:k}) \approx \sum_{i=1}^{N} \bar{q}_k^{(i)} \delta(\theta_k^{(i)} - \theta_k) \tag{16}$$

where $\delta(\cdot)$ denotes the delta-Dirac function. A straightforward way to recursive update the particles $\theta_k^{(i)}$ and weights $\bar{q}_k^{(i)}$ is given by

$$\theta_{k+1}^{(i)} \sim p(\theta_{k+1} \mid \theta_k^{(i)}), i = 1, \cdots, M \tag{17}$$

$$\overline{q}_{k+1}^{(i)} = \frac{p(y_{k+1} \mid \theta_{k+1}^{(i)})\overline{q}_k^{(i)}}{\sum_{j=1}^{M} p(y_{k+1} \mid \theta_{k+1}^{(i)})\overline{q}_k^{(j)}}, i = 1, \cdots, M \tag{18}$$

initiated at time $k = 0$ with

$$\theta_0^{(i)} \sim p(\theta_0), \ \overline{q}_0^{(i)} = 1/N, \ i = 1, \cdots, M \tag{19}$$

The discussion above is formalized in algorithm below. All the operations in the algorithm have to be done for all particles $\theta_k^{(i)}$.

Algorithm 1. The particle filter

(1) Sample $\theta_0 |_{-1} \sim p(\theta_0)$;

(2) Calculate the weights $q_k = p(y_k \mid \theta_k)$ and normalize, i. e.,

$$\overline{q}_k^{(i)} = \frac{q_k^{(i)}}{\sum_{j=1}^{M} q_k^{(j)}}, i = 1, \cdots, M \ ;$$

(3) Generate a new set $\{\theta_k^{(i*)}\}_{i=1}^{M}$ by resampling with replacement M times from $\{\theta_k^{(i)}\}_{i=1}^{M}$, with probability $P\{\theta_k^{(i*)} = \theta_k^j\} = \overline{q}_k^j$;

(4) Predict (simulate) new particles, i.e., $\theta_{k+1|k}^{(i)} = \theta_{k|k}^{(i)} + \eta_k^{(i)}, i = 1, \cdots, M$;

(5) Increase k and iterate from step 2.

5 Sequential SVM Control of Nonlinear Systems

We consider a single-input-single-out nonlinear system [10], [11]

$$y^{(n)} = f(y, y^{(1)}, y^{(2)}, \cdots y^{(n-1)}, u) \tag{20}$$

where u is the control input, y is the measured output, $(\cdot)^{(i)}$ denotes the i th derivative of $(\cdot)$, $f(\cdot)$ is a smooth function with respect to input u. The objective is to enable the output y to follow a desired trajectory y_d . Let $x = [x_1, x_2, \cdots x_n]^T = [y, y^{(1)}, \cdots, y^{(n-1)}]^T \in R^n$ be the state vector, we may represent system (20) in a state space model form

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = x_3 \\ \quad \vdots \\ \dot{x}_n = a(x) + b(x)u \\ y = x_1 \end{cases} \tag{21}$$

Assumption 1: $b(x) \neq 0$, $a(x)$ are bounded and the sign of $b(x)$ is known. The objective trajectory y_d is n-order derivable and bounded.

Define vector x_d, e and a filtered tracking error s as

$$x_d = [y_d, \dot{y}_d, \cdots, y_d^{(n-1)}]^T,\ e = x - x_d = [e_1, e_2, \cdots, e_n]^T,\ s = [\Lambda \quad 1]^T e \tag{22}$$

where $\Lambda = [\lambda_1, \lambda_2, \cdots \lambda_{n-1}]^T$ is an appropriately chosen coefficient vector so that $e \to 0$ as $s \to 0$. Then the time derivative of the filtered tracking error can be written as

$$\dot{s} = a(x) + b(x)u - y_d^{(n)} + [0 \quad \Lambda^T]e \tag{23}$$

Define a saturation function [10]

$$sat(s) = \begin{cases} 1 - \exp(-s/\gamma), & s > 0 \\ -1 + \exp(s/\gamma) & s \le 0 \end{cases} \tag{24}$$

where γ denotes any small positive constant.

Without losing generality, we will assume that $b(x) > 0$. We have the following lemma to present an ideal control, u^*, such that under the ideal control, the output y can follow the objective trajectory y_d asymptotically.

Lemma 1: For the system (21) satisfying Assumption 1, if the ideal control is designed as

$$u^* = -k(t)s - k(t)sat(s) - (a(x) + v)/b(x) \tag{25}$$

where $k(t) > 0$, for all t, is a design parameter, then the filtered tracking error converges to zeros.

The lemma can be proven easily by introducing a Lyapunov function $V = (1/2)s^2$. Following the (25), the ideal control u^* can be written as

$$u^* = -k(t)s - k(t)sat(s) - u_1^* \tag{26}$$

where $u_1^* = (a(x) + v)/b(x)$. However, the nonlinear function $a(x)$, $b(x)$ are unknown, u^* is unknown. In what follows, the SVM is used to approximate the unknown function u_1^*. So the following approximation holds,

$$u_1^*(z) = (a(x) + v)/b(x) = w^T \varphi(z) + b + \varsigma,\ z = [x^T \quad v]^T \tag{27}$$

where ς denotes the approximation error.

6 Numerical Simulations

In this section we give two examples to illustrate the effectiveness of the proposed SVM control of nonlinear systems by filtered training algorithms of SVM. The first nonlinear plant [10] is

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= x_1^2 + 0.15u^3 + 0.1(1 + x_2^2)u + \sin(0.1u) \\ y &= x_1 \end{aligned} \tag{28}$$

It is impossible to get an explicit controller for system feedback linearization because the nonlinearities in the plant (28) are an implicit function with respect to u. In this example the objective is to make the output y track a desired trajectory $y_d = \sin(t) + \cos(0.5t)$. The initial conditions are $x(0) = [0.6 \;\; 0.6]^T$.

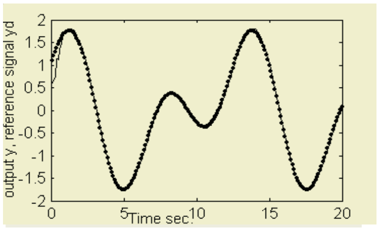


Fig. 1. Tracking performance of state feedback control based on SVM trained by Kalman filter.

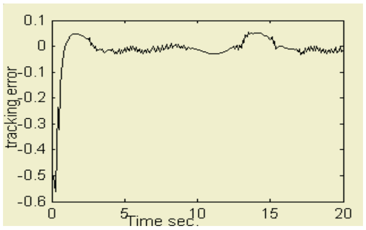


Fig. 2. Tracking error curve, SVM is trained by Kalman filter

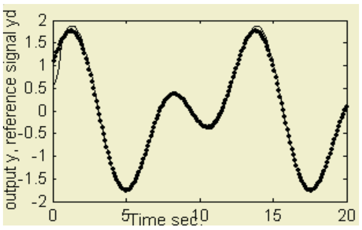


Fig. 3. Tracking performance of state feedback control based on SVM trained by particle filter

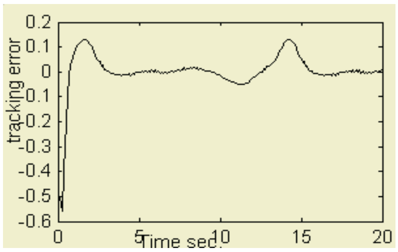


Fig. 4. Tracking error, SVM is trained by particle filter

The sequential SVM controller $u_1^*(z) = w^T\varphi(z) + b$ has been chosen. The kernel function is the Gaussian function. The SVM is trained online by Kalman filter with $L = 50$, $R = 1$, $Q = 2$. Other controller parameters are chosen as $\Lambda = 3.2$, $k(t) = 2.2$, and $\gamma = 0.03$ in simulation. Figure 1 shows that the output y tracks the desired trajectory y_d effectively. Figure.2 is the tracking error curve. The dot linedenotes desired trajectory y_d, while the solid line denotes the output curve y. Figure 1-2 are simulation results of state feedback control based on SVM trained by Kalman filter. Figure 3-4 are simulation results of state feedback control based on SVM trained by particle filter that the number of particles is chosen as $M = 200$. Comparing the figure 2 with figure 4, we may know the simulation results that is obtained by SVM controller, which is trained by Kalman filter, overwhelm that of SVM controller trained by particle filter, because SVM's solutions of Kalmam filtering estimation are optimal while that of the particle filter estimation are suboptimal.

The numerical simulation results above show good transient performance and the tracking error is small. This demonstrates that the sequential SVM controller can achieve a satisfactory performance.

7 Conclusions

In this paper we discussed sequential SVM for regression estimation and sequential SVM control of nonlinear systems. Firstly we provide the representation of state-space for SVM. The sequential SVM is realized by Kalman filter and particle filter respectively. The advantage of the proposed method is that the computational complexity decreases and thus the on-line training and control become feasible. Finally we present a new method of controller designing and demonstrate that this method is feasible for nonaffine nonlinear systems. Interesting questions for future research include: (1) the method of training SVM in this paper decreases the generalization ability, then how to train the SVM on-line without decreasing the generalization ability should be studied; (2) in this paper we discussed sequential SVM controller of nonlinear systems by state feedback. In what follows, the approximation property of SVM and output feedback control problem based on SVM need to be studied.

Acknowledgments

China 973 Plan supported this work under grant NO. 2002CB312200.

References

1. Vapnik, V. N.: Statistical Learning Theory. John Wiley and Sons, New York (1998)
2. de Freitas, J. F. G., Niranjan, M., Gee, A. H., Doucet, A.: Sequential Monte Carlo Methods to Train Neural Network Models. Neural Computation, **12** (2000) 955-993
3. Kalman, R. E and Bucy, R S: New Results of in Linear Filtering and Prediction Theory. Transaction of the ASME (Journal of Basic Engineering), **83** (1961) 95-108
4. Storvik G.: Particle Filters in State Space Models with the Presence of Unknown Static Parameters. IEEE Transactions on Signal Processing, **50** (2002) 281-289
5. Arnaud Doucet, Nando de Freitas, Kevin Murphy, Stuart Russell: Rao-Blackwellised Particle Filtering for Dynamic Bayesian Networks. Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence, Stanford (2000) 176-183
6. Liu, S., Chen, R.: Sequential Monte Carlo Methods for Dynamic Systems. Journal of the American Statistical Association, **93** (1998) 1032-44
7. Crisan, D., Doucet, A.: A Survey of Convergence Results on Particle Filtering Methods or Practitioners. IEEE Transactions on Signal Processing, **50** (2002) 736-746
8. Sun, Z.: Study on Support Vector Machine and Its Application in Control. PhD. Thesis, Zhejiang University, Hangzhou, China (2003)
9. Arulampalam, S., Maskell, S., Gordon, N., and Clapp, T.: A Tutorial on Particle Filters for On-Line Nonlinear /Non-Gaussian Bayesian Tracking. IEEE Trans. Signal Process., **50** (2002) 174-189
10. Ge, S.S., Hang, C. C. and Zhang, T.: Adaptive Neural Network Control of Nonlinear Systems by State and Output Feedback. IEEE Trans. SMC-part B: Cybernetics, **29** (1999) 818-828
11. Calise, A.J., Hovakimyan, N., and Idan, M.: Adaptive Output Feedback Control of Nonlinear Systems Using Neural Networks. Automatica, **37** (2001) 1201-1211

RBFNN-Based Multiple Steady States Controller for Nonlinear System and Its Application

Xiugai Li[1,2], Dexian Huang[1], and Yihui Jin[1]

[1] Department of Automation, Tsinghua University, Beijing 100084, China
{li-xg,huangdx}@mail.tsinghua.edu.cn
[2] Key Lab of Complex Systems and Intelligent Science, Institute of Automation, Chinese Academy of Sciences, Beijing 100080, China

Abstract. On-line Radial Basis Function (RBF) neural network based multiple steady states controller for nonlinear system is presented. The unsafe upper steady states can be prevented with the optimizer for Constrained General Model Controller (CGMC).Process simulator package is used to generate a wide range of operation data and the dynamic simulator is built as the real plant. The effectiveness is illustrated with a Continuous Stirred Tank Reactor (CSTR) and OPC tools are developed for on-line data acquisition and computation.

1 Introduction

The main problem concerning the safety operation of a chemical reactor is the analysis of the existence of multiple steady states. It means that different conversations can be reached under the same conditions [1],[2]. Although a lot of researches have been made on the chemical reactor, most of them focused on tracking the setpoints of reactor temperature with various disturbance [3],[4]. For the classic approach for CSTR, which involves a single homogenous exothermic reaction of the first order in a well-stirred continuously fed reactor, some theoretical papers have proven it also exhibit multiple steady state and sustained oscillatory outputs [1],[2],[5]. It is well know that in many reaction systems the upper steady-state may be sufficiently high and is undesirable or even dangerous to operate under these conditions. In this paper on-line RBF neural network controller for multiplicity steady state of a CSTR is presented. CGMC based optimizer is presented to prevent the upper steady states from reaching the unsafe upper limit.

2 Plant Description and the Multiple Steady States Characteristic

In this section, an exothermic reaction involving the propylene oxide to propylene glycol in a CSTR with jacked cooling is discussed. Reaction between water and propylene is $H_2O+C_3H_6O \rightarrow C_3H_8O_2$.The physical constants of the CSTR are shown in Table 1, and the kinetic model of the CSTR with irreversible reaction is described as:

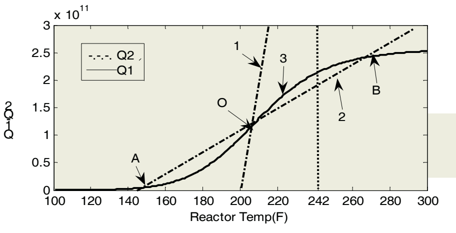
$$\frac{dC_A}{dt} = \frac{F}{V}(C_{A0} - C_A) - kC_A \quad . \tag{1}$$

$$\frac{dT}{dt} = -\frac{F}{V}(T - T_{in}) + \frac{\Delta H}{\rho C_p} kC_A - \frac{UC_R}{\rho C_p V}(T - T_c) \, . \tag{2}$$

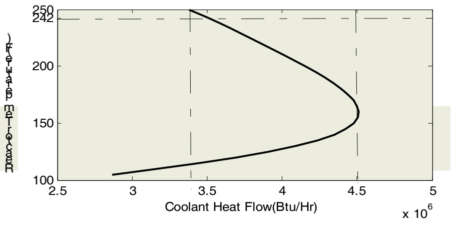
Table 1. Physical Constants and Steady State Operation Range of the CSTR

Parameters	Definition	Value
$-(\Delta H)$	Heat of the reaction	$9e^4$ kJ/kgmole
ρ	Mole Density of reactant	14.173 kgmole/m^3
C_p	Heat capacity	4.2 J/g.K
E	Activation energy	75362kJ/kgmole
R	Gas Constant	8.3196 KJ/mole.K
F_2	Water Feed Mass Flow	9000~11000 (Lb/Hr)
Q	Coolant Heat Flow	$2\times10^6 \sim 3.2\times10^6$ (Btu/Hr)
F_1	Propylene Oxide Mass Flow	8000~9300(Lb/Hr)

Fig.1 (a) shows the existence of multiplicity steady states of this reaction. The cooling heat flow (Q_2 dash line) shows the heat withdraw curve of the reactant and coolant. The S-shaped heating curve (Q_1 solid line) shows the relation between generated heat by the reaction and the reactor temperature T. The possible steady state operating points are the points where the two curves intersect [6]. The points are assumed to be stable only in the case when slope of Q_2 is higher than that of Q_1. Relation between coolant heat flow and the reactor temperature is shown in Fig.1 (b) .Two points located at coolant heat flow of 3.4413e^6 Btu/Hr and 4.51e^6 Btu/Hr bound a region of multiple steady states. Existence of multiple steady states makes it is very challenging to design controller for this nonlinear system.



(a)



(b)

Fig. 1. Multiplicity steady states of the CSTR Process

Main aim of the controller is to prevent the reactor temperature from entering the unstable or unsafe upper steady states when the manipulated variables are acquired.

3 RBFNN-Based Multiple Steady States Controller

RBFNN have been well known for its better approximation capacity, simple structure, and fast learning, is becoming increasingly popular in many scientific areas, especially in chemical engineering [7]. The structure of RBFNN based multiple steady states controller is shown in Fig.2.

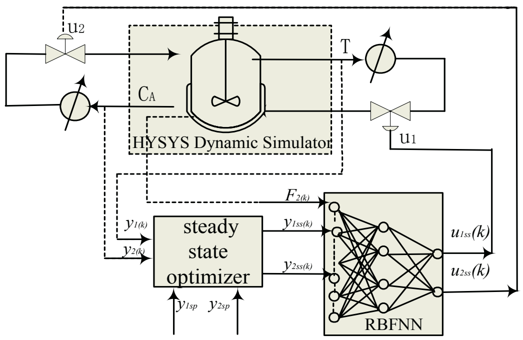


Fig. 2. Structure of Multi-Steady-State Controller for CSTR

3.1 Off-Line Training of the Multi-steady-State Inverse Characteristic

HYSYS is a process simulation environment designed to serve many processing industrials especially in the Oil& Gas and Refining [8]. With HYSYS, We can create steady state and dynamic models for plant design, performance monitoring, etc. Steady state data of the CSTR for training and testing the RBFNN are comprised of 45 data points separately. All 90 data points were obtained by considering 3 data points for both the propylene oxide mass flow and the water mass flow and 10 data points for coolant heat flow, thus giving $3\times10\times3=90$ steady state data points which bridge a wide range operating condition (Table (1)). Gaussian function output $\varphi_{i,j}$ is given by:

$$\varphi_{i,j}(X)=\exp(\frac{-\left\|X_i-C_j\right\|^2}{2\sigma_j^2})((i=1,2,\cdots M,j=1,2,\cdots h)\,. \tag{3}$$

Hidden output is $X1_{i,j}=\varphi_{i,j}$, and output of the RBFNN is $Y=\begin{bmatrix}y_1 & y_2 & \cdots & y_N\end{bmatrix}^T$ defined as: $Y=X1\times W$, where: $x_i\in R^m$ and $y_i\in R^n$. The objective function is defined as:

$$E_c^s=\frac{1}{2}\left\|\hat{U}(k)-U^{ss}(k)\right\|^2\,. \tag{4}$$

The steepest descent algorithm is used to calculate the gradient by the chain rule. The following learning rule can be inferred:

$$\frac{\partial E(k)}{\partial W(k)}=-X1(k)^T(\hat{U}(k)-U^{ss}(k))\,. \tag{5}$$

$$\frac{\partial E(k)}{\partial\sigma_j(k)}=-(\hat{U}(k)-U^{ss}(k))W(k)^T(\sum_{l=1}^{N}X1_{jl}\frac{\left\|X_l-C_j\right\|^2}{\sigma_j^3})\,. \tag{6}$$

$$\frac{\partial E(k)}{\partial C_{ij}(k)} = \frac{1}{\sigma_j^2}\sum_{l=1}^{N}[(\frac{\partial E_l}{\partial X1_{li}} X1_{li})(X0_{li} - C(k)_{ij})]\,. \tag{7}$$

Training and testing the RBF can be done off-line according to the above learning rules. Fig.3 shows the comparison with testing data and output of the RBFNN.

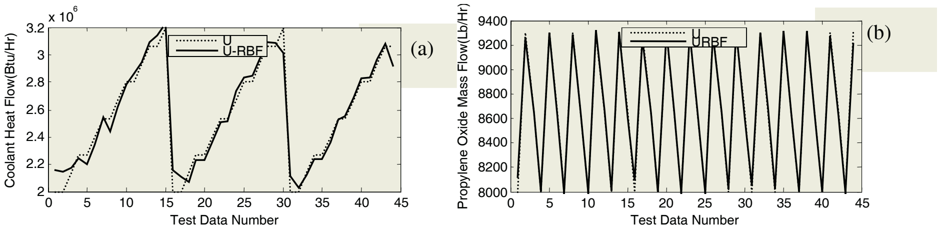


Fig. 3. (a) Coolant heat flow (Btu/Hr), (b) Propylene Oxide mass flow(Lb/Hr)

3.2 CGMC Based Multi-steady-state Optimizer

GMC had got many successfully in nonlinear system [9].By presuming that the controlled variables have first order responses and forcing output variables toward steady state at a specified time rate, Eqs. (8) can be obtained by omitting the bias:

$$Y_{ss} = Y(k) + K_p(Y_{sp}(k) - Y(k)) + K_2\sum_{j=0}^{k} Y_{sp} - Y(j)\,. \tag{8}$$

Where Y_{ss} is the steady state target value, and Y_{sp} is the desired setpoints for output variables, K_p and K_2 are the tuning parameters. CGMC is presented to make the output steady states are within the safe operation limit. Giving the following criterion:

$$\min_{Y_{ss}(k)} J = \frac{1}{2}\left\| Y_{sp}(k) - Y(k) \right\|_Q^2 \,. \qquad \text{s.t.} \quad Y_{\min} \le Y_{ss}(k) \le Y_{\max} \tag{9}$$

By substituting (8) to (9) and omitting the integer terms, the standard quadratic programming (QP) problem can be derived with the tuning parameters K_p

$$\min_{Y_{ss}(k)} J = \frac{\widetilde{K}_p^{-1}}{2}(Y_{ss}(k)^T QY_{ss}(k) - 2Y_{sp}(k)^T QY_{ss}(k) + Y_{sp}(k)^T QY_{sp}(k))\,. \tag{10}$$

where $\widetilde{K}_p = K_p - I$. Steps of controller can be summarized as follows:

Step1: set Y_{sp}, K_p, and the timer for optimizer, sample time, weight matrix Q, Y_{min},Y_{max}. Off-line training inverse steady states model: $RBF(Y_{ss}, C, \sigma, W)$ and load all the parameters of the RBFNN for on-line application.
Step2: (on-line) compute $Y_{ss}(k)$ according to the CGMC based optimizer.
Step3: compute the $U_{ss}(k)$ with RBFNN based multiple steady states controller $U_{ss}(k) = RBF(Y_{ss}(k), C, \sigma, W)$, and write $U_{ss}(k)$ to the HYSYS dynamic plant via OPC asynchronous write operation .
Step4: when k=$k+1$, go to step2.

An OPC data communication tools [10] was developed to accomplish the online transmission of process data from HYSYS simulator into Matlab (MATLAB Version 7.0.0). In OPC server (HYSYS_OS.DAServer), a group of variables had been defined to be monitored. OPC client was build with Matlab script used for communicating between OPC Server and Matlab.

4 Application to CSTR Plant

Fig.4 shows the system response of the setpoints tracking with PID and multiple steady state (MSS) controller. At the time instant 106,204s, controller was switched from PID to MSS control, response of the reactor temperature and the reactant mole concentration can be seen from Fig.4(a) and (b).

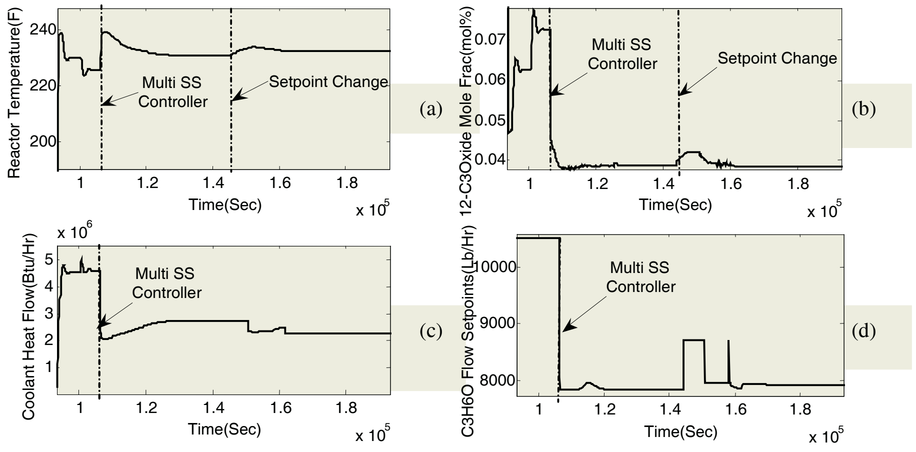


Fig. 4. Setpoints tracking with PID controller and MSS controller (a) reactor temperature (b) mole concentration of the reactant in reactor (c) coolant heat flow (d) setpoints for reactant mass flow

The setpoints for PID controller is switched from 230F to 225.5F for reactor temperature, and from 0.062% to 0.072% for reactant concentration in reactor. When switched to MSS controller, a simultaneous setpoints change to 231F and 0.039% were given on the *T* and C_A, respectively. After the changes are made, MSS controller can get the value of the manipulated variables for the new steady state operation condition immediately, and drive the plant to it. At time instant 143987s, a setpoints change to 232F is given on the *T* for MSS controller. Fig. 4 (c) and (d) shows the change of manipulated variables during the run. At 157967s, disturbance is given to the propylene oxide mass flow to test the controller performance. Compared with the PID controller for tracking problem, the MSS controller shows a more stable and little overshooting performance.

5 Conclusions

Multiplicity steady states for nonlinear system of the chemical plant is a challenging research topic and attract a lot research interest in recent years. There remain a lot of problems wait to be solved: such as, which steady states are stable? which new steady state we want the system move to if process conditions change? The on-line RBFNN based multiple steady states controller for nonlinear system is presented aim at the last problem. The unsafe upper steady states can be prevented by CGMC based optimizer. Dynamic and steady state process of a CSTR is build with process simulator HYSYS. OPC tools are developed for on-line data acquisition and processing. The further research will focus on the identification of undesirable or unsafe steady states for the nonlinear system and its application in distillation column.

Acknowledgements

The research is supported by National 973 program (Grant No.2002CB312203) and 863 program (Grant No.2004AA412050) . The use of an academic licensed of HYSYSTM provided by Hyprotech is also appreciated.

References

1. Molnar, J.M.: Safety Analysis of CSTR Towards Changes in Operating Conditions. J.Loss Prev. Proc. Ind, **16** (2003) 373-380
2. Sivakumar,S.C.: Steady-state Operability Characteristics of Reactors. Computers Chem. Engng, **24** (2000) 1563-1568
3. Knapp, T.D., Buddman, H.M.: Adaptive Control of a CSTR with a Neural Network Model. J.Proc.Cont, **1** (2001) 53-68
4. Galvan, I.M., Zaldivar, J.M.: Application of Recurrent Neural Networks in Batch Reactors Part ∏ :Nonlinear Inverse and Predictive Control of the Heat Transfer Fluid Temperature. Chem. Engineering Processing, **37** (1998) 149-161
5. Fogler, H. S.: Elements of Chemical Reaction Engineering. Prentice Hall, London (1999)
6. Ka[illegible]apoB, B.B.: Application of Cybernetics in Chemic and Chemical Engineering. Chemical Industry Publishing Company, Beijing, China (1983)
7. Pottmann, M., Seborg, D.E.: A Nonlinear Predictive Control Strategy Based on Radial Basis Function Models. Computers Chem. Engng, **21** (1997) 965-980
8. Aspen Technology, Inc.: HYSYS 3.2 Documentation, User Guide. Ten Canal Park, Cambridge, MA02141, U.S.A. (2003)
9. Lee, P.L., Sullivan, G.R.: Generic Model Control (GMC). Computers Chem. Engng, **12** (1988) 573-580
10. OPC Foundation: OPC Data Access Specification 1.0 (1997)

Sliding Mode Control for Uncertain Nonlinear Systems Using RBF Neural Networks

Xu Zha and Pingyuan Cui

Astronautics school, Harbin Institute of Technology,
Harbin 150001, China
tacitrainbow@netease.com

Abstract. A robust sliding mode adaptive tracking controller using RBF neural networks is proposed for uncertain SISO nonlinear dynamical systems with unknown nonlinearity. The Lyapunov synthesis approach and sliding mode method are used to develop a state-feedback adaptive control algorithm by using RBF neural networks. Furthermore, the H_∞ tracking design technique and the sliding mode control method are incorporated into the adaptive neural networks control scheme so that the derived controller is robust with respect to disturbances and approximate errors. Compared with conventional methods, the proposed approach assures closed-loop stability and guarantees an H_∞ tracking performance for the overall system. Simulation results verify the effectiveness of the designed scheme and the theoretical discussions.

1 Introduction

In recent years, the analytical study of adaptive nonlinear control systems using universal function approximators has received much attention [1]. Typically, these methods use neural networks as approximation models for the unknown system nonlinearity [2], [3]. The control systems based on a fuzzy-neural control scheme are augmented with sliding mode control (SMC) [4] to ensure global stability and robustness to disturbances. With the use of the adaptive fuzzy-neural control [5], [6], and the sliding mode control [7], two objectives can be achieved. First, modeling impression and bounded disturbances are effectively compensated. Secondly, the stability and robustness of the system can be verified.

Based on the initial results of SISO nonlinear systems [8], we intend to develop sliding mode adaptive neural networks control for SISO nonlinear systems with unknown nonlinearities. The sliding mode control method is incorporated into the adaptive neural networks control scheme to derive the control law. The proposed controller guarantees that tracking error converges in the small neighborhood of zero under less restrictive assumptions and that the states and estimated parameters are all bounded. Subsequently, a novel sliding mode adaptive neural networks controller with H_∞ tracking performance for uncertain nonlinear systems is proposed. The overall system by using the H_∞ tracking-based sliding mode adaptive controller using RBF neural networks is robust with respect to disturbances and approximate errors.

2 Problem Formulation

2.1 System Description

Consider an nth-order uncertain nonlinear system of the following form:

$$\begin{aligned} \dot{x}_i &= x_{i+1} \qquad 1 \le i \le n-1 \\ \dot{x}_n &= f(x) + g(x)u + d(x,t) \\ y &= x_1 \end{aligned} \tag{1}$$

where $u \in R$ and $y \in R$ represent the control input and the output of the system, respectively. $x = (x_1, x_2, \ldots, x_n)^T \in R^n$ is comprised of the states which are assumed to be available, the integer n denotes the dimension of the system. $f(x)$ is an unknown smooth uncertain system function and $g(x)$ is an unknown smooth uncertain input gain function. $d(x,t)$ is disturbance of the system.

2.2 Function Approximation Using RBF Neural Networks

The control design presented in this paper employs RBF NNs to approximate the nonlinear functions in system (1). RBF networks are of the general form $\hat{F}(\cdot) = \theta^T \xi(\cdot)$, where $\theta \in R^p$ is a vector of adjustable weights and $\xi(\cdot)$ a vector of Gaussian basis functions. Their ability to uniformly approximate smooth functions over compact sets is well documented in the literature [9]. In general, it has been shown that given a smooth function $F : \Omega \to R$, where Ω is a compact subset of R^m (m is an appropriate integer) and $\varepsilon > 0$, there exists a RBF vector $\xi : R^m \to R^p$ and a weight vector $\theta^* \in R^p$ such that $\left|F(x) - \theta^{*T}\xi(x)\right| \le \varepsilon, \forall x \in \Omega$ .The quantity $F(x) - \theta^{*T}\xi(x) = d_F(x)$ is called the network reconstruction error and obviously $|d_F(x)| \le \varepsilon$ .The optimal weight vector θ^* defined above is a quantity only for analytical purposes. Typically θ^* is chosen as the value of θ that minimizes $d_F(x)$ over Ω , that is [2]

$$\theta^* = \arg\min_{\theta \in R^p} \left[\sup_{x \in \Omega} \left| F(x) - \theta^T \xi(x) \right| \right] \tag{2}$$

3 Sliding Mode Adaptive Tracking Controller Using RBF Neural Networks

For a given reference trajectory y_d , we define the tracking error vector $e = \left(e_1, \dot{e}_1, \ldots, e_1^{(n-1)}\right)^T$ with $e_1 = x_1 - y_d$, and the system (1) can be written as

$$\dot{e}_i = e_{i+1}, \qquad 1 \le i \le n-1$$

$$\dot{e}_n = f(x) + g(x)u(t) + d(x,t) - y_d^{(n)} \tag{3}$$

We define that

$$S(e,t) = e_n + k_1 e_1 + \cdots + k_{n-1} e_{n-1} = e_n + \sum_{i=1}^{n-1} k_i e_i \tag{4}$$

where k_i is chosen such that all roots of $s^{n-1} + k_{n-1}s^{n-2} + \cdots + k_1 = 0$ are in the open left-half complex plane. We choose

$$\dot{S}(e,t) = \dot{e}_n + \sum_{i=1}^{n-1} k_i e_{i+1} \tag{5}$$

The unknown functions f and g are expressed as

$$f(x) = f_0(x) + \Delta f(x),\ g(x) = g_0(x) + \Delta g(x) \tag{6}$$

where f_0 fn and g_0 are known functions (representing some nominal estimates of f and g, respectively) and $\Delta f(x)$ and $\Delta g(x)$ are unknown functions representing the system uncertainties. The unknown functions $\Delta f(x)$ and $\Delta g(x)$, representing the system uncertainties, are approximated by neural networks as follows:

$$\Delta\hat{f}(x,\theta_f) = \theta_f^T \xi_f(x),\ \Delta\hat{g}(x,\theta_g) = \theta_g^T \xi_g(x) \tag{7}$$

where θ_f and θ_g are adjustable parameter vectors with appropriate dimensions, and $\xi_f(x)$, $\xi_g(x)$ are vectors of RBFs. The uncertainties Δf and Δg can be expressed as

$$\Delta f(x) = \Delta\hat{f}(x,\theta_f^*) + \omega_f,\ \Delta g(x) = \Delta\hat{g}(x,\theta_g^*) + \omega_g \tag{8}$$

where θ_f^* and θ_g^* are some unknown optimal parameter vectors, and ω_f and ω_g represent the reconstruction errors for each neural network system. The optimal parameter vectors θ_f^* and θ_g^* are analytical quantities required only for analytical purposes. Typically, θ_f^* and θ_g^* are chosen as the values of θ_f and θ_g, respectively that minimize the reconstruction errors, i.e.,

$$\theta_f^* = \arg\min_{\theta_f}\left[\sup_{x\in U_c}\left|\Delta\hat{f}(x,\theta_f) - \Delta f(x)\right|\right],\ \theta_g^* = \arg\min_{\theta_g}\left[\sup_{x\in U_c}\left|\Delta\hat{g}(x,\theta_g) - \Delta g(x)\right|\right] \tag{9}$$

*Assumption 1:*On the compact region U

$$\left|\omega_f(x)\right| \le \psi_f^* \rho_f(x)\ ,\ \left|\omega_g(x)\right| \le \psi_g^* \rho_g(x), \qquad \forall x \in U \tag{10}$$

where $\psi_f^* \ge 0$ and $\psi_g^* \ge 0$ are unknown bounding parameters and $\rho_f(x): U \to R^+$ and $\rho_g(x): U \to R^+$ are known smooth bounding functions.

Assumption 2: The disturbance is assumed such that $d(x,t) \in L_2[0,T], \forall T \in [0,\infty)$, and $d(x,t)$ is bounded by some positive constant $\rho_d : |d(x,t)| \le \rho_d$.

Substitute (5) into (3), and we obtain [10]

$$\dot{e}_i = e_{i+1}, \dot{e}_{n-1} = -k_1 e_1 - \cdots - k_{n-1} e_{n-1} + S \quad 1 \le i \le n-1$$
$$\dot{S} = f_0(x) + \Delta f(x) + [g_0(x) + \Delta g(x)]u(t) + d - y_d^{(n)} + \sum_{i=1}^{n-1} k_i e_{i+1} \tag{11}$$

Suppose $\xi = [e_1, e_2, \cdots, e_{n-1}]^T$, and multiply (4) by k_{n-1}. Then (11) can be rewritten as

$$\dot{\xi} = A\xi + BS$$
$$\dot{S} = f_0(x) + \Delta f(x) + [g_0(x) + \Delta g(x)]u(t) + d - y_d^{(n)} + k_{n-1}S - C\xi \tag{12}$$

where

$$A = \begin{bmatrix} 0 & 1 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ -k_1 & -k_2 & \cdots & -k_{n-1} \end{bmatrix}, B = [0,0,\cdots,1]^T, C = \left[k_{n-1}k_1, (k_{n-1}k_2 - k_1), \cdots, (k_{n-1}^2 - k_{n-2})^2\right]^T.$$

Because A is stable, there exists a real symmetric positive definite matrix P satisfying the Lyapunov equation (13). Let the control law be chosen as

$$A^T P + PA + Q = 0 \tag{13}$$
$$u = u_c + u_N \tag{14}$$

where u_c is the certainty equivalent controller , and u_N is the bounding controller and is needed to improve robustness with respect to the bounding uncertainty. We choose u_c and u_N as follows:

$$u_c = u_{equ} = \left[-f_0(x) - \Delta\hat{f} - (k_{n-1} + k/2)S + C\xi + y_d^{(n)} - B^T P\xi\right] / \hat{g}(x, \theta_g) \tag{15}$$
$$u_N = -(\alpha + \beta) / \hat{g}(x, \theta_g) \tag{16}$$
$$\alpha = \rho_d \tanh(S\rho_d / \varepsilon) \tag{17}$$
$$\beta = (\psi^T w) / \left(1 - \psi_g \rho_g / \left|\hat{g}(x, \theta_g)\right|\right) \tag{18}$$
$$\hat{g}(x, \theta_g) = [g_0(x) + \Delta\hat{g}(x)] \tag{19}$$

where $\psi = [\psi_f, \psi_g]^T$ is an estimation for the bounding constants of Assumption 1 and w will be defined later. Vector ψ is the estimated bounding parameter. Substitute (7), (8), (14) and (15) into (12), and we obtain

$$\dot{\xi} = A\xi + BS$$
$$\dot{S} = -\frac{1}{2}kS + \omega_f + \omega_g u + \tilde{\theta}_f^T \xi_f + \tilde{\theta}_g^T \xi_g u - B^T P\xi + d + \hat{g}(x, \theta_g)u_N \tag{20}$$

Consider the following Lyapunov candidate

$$V = \frac{1}{2}S^2 + \frac{1}{2}\xi^T P\xi + \frac{1}{2\gamma_f}\tilde{\theta}_f^T \tilde{\theta}_f + \frac{1}{2\gamma_g}\tilde{\theta}_g^T \tilde{\theta}_g + \frac{1}{2\gamma_\psi}\tilde{\psi}^T \tilde{\psi} \tag{21}$$

where $\tilde{\theta}_f = \hat{\theta}_f - \theta_f^*, \tilde{\theta}_g = \hat{\theta}_g - \theta_g^*$ and $\tilde{\psi} = \hat{\psi} - \psi^*$. The derivative of V is

$$\dot{V} = S\dot{S} + \frac{1}{2}\dot{\xi}^T P\xi + \frac{1}{2}\xi^T P\dot{\xi} + \frac{1}{\gamma_f}\tilde{\theta}_f^T\dot{\theta}_f + \frac{1}{\gamma_g}\tilde{\theta}_g^T\dot{\theta}_g + \frac{1}{\gamma_\psi}\tilde{\psi}^T\dot{\psi} \tag{22}$$

The adaptive laws for θ_f and θ_g are chosen as

$$\dot{\theta}_f = -\gamma_f\left[S\xi_f + \sigma\left(\theta_f - \theta_f^0\right)\right], \dot{\theta}_g = -\gamma_g\left[S\xi_g u + \sigma\left(\theta_g - \theta_g^0\right)\right] \tag{23}$$

where θ_f^0 , θ_g^0 and σ_f, σ_g are design constants. These adaptive laws incorporate a leakage term based on a variant of the σ -modification, which prevents parameter drift of the system. Using the above adaptive laws, we obtain

$$\dot{V} = -\frac{1}{2}\xi^T Q\xi - \frac{1}{2}kS^2 - \sigma_f\left(\theta_f - \theta_f^0\right) - \sigma_g\left(\theta_g - \theta_g^0\right) + \varepsilon_v \tag{24}$$

$$\varepsilon_v = S\omega_f + S\omega_g u - S\beta + \frac{1}{r_\psi}\tilde{\psi}^T\dot{\psi} + \kappa\varepsilon \tag{25}$$

where $\kappa = 0.2785$ .We define $z = -S$ and using (10), and (14)-(19), (25) becomes

$$\begin{aligned}\varepsilon_v &= -z\omega_f - z\omega_g u_c + z\beta - z\omega_g\frac{\beta}{\hat{g}} + \frac{1}{\gamma_\psi}\tilde{\psi}^T\dot{\psi} + \kappa\varepsilon \\ &\le \frac{1}{\gamma_\psi}\tilde{\psi}^T\dot{\psi} + |z|\psi^{*T}\rho + |z\beta|\frac{\psi_g^*\rho_g}{|\hat{g}|} + z\beta + \kappa\varepsilon\end{aligned} \tag{26}$$

where $\rho = \left[\rho_f, \rho_g|u_c|\right]^T$. We choose w as

$$w = [w_1, w_2]^T = \left[\rho_f \tanh\left(z\rho_f/\varepsilon\right), \rho_g|u_c|\tanh\left(z\rho_g|u_c|/\varepsilon\right)\right]^T \tag{27}$$

where $\varepsilon > 0$ is a small positive design constant. Using the inequality $z\beta \le 0$, (26) and the adaptation law (28)[8], we obtain

$$\dot{\psi} = \gamma_\psi\left[zw_1 - \sigma\left(\psi_f - \psi_f^0\right), zw_2 - z\beta\frac{\psi_g}{|\hat{g}|} - \sigma\left(\psi_g - \psi_g^0\right)\right]^T \tag{28}$$

$$\begin{aligned}\varepsilon_v &\le \frac{1}{\gamma_\psi}\tilde{\psi}^T\dot{\psi} + |z|\psi^{*T}\rho - z\psi^T w + z\beta\frac{\psi_g\rho_g}{|\hat{g}|} + |z\beta|\frac{\psi_g^*\rho_g}{|\hat{g}|} + \kappa\varepsilon \\ &\le -\sigma\tilde{\psi}^T\left(\psi - \psi^0\right) + \kappa\varepsilon\left(\left\|\psi^*\right\| + 1\right)\end{aligned} \tag{29}$$

where $\psi^0 = \left[\psi_f^0, \psi_g^0\right]^T$ .Substituting (29) into (24) and then completing the square give

$$\begin{aligned}\dot{V} &\le -\frac{1}{2}\xi^T Q\xi - \frac{1}{2}kS^2 - \frac{\sigma}{2}\left(\left\|\tilde{\theta}_f\right\|^2 + \left\|\tilde{\theta}_g\right\|^2 + \left\|\tilde{\psi}\right\|^2\right) \\ &+ \frac{\sigma}{2}\left(\left\|\theta_f^* - \theta_f^0\right\|^2 + \left\|\theta_g^* - \theta_g^0\right\|^2 + \left\|\psi^* - \psi^0\right\|^2\right) + \kappa\varepsilon\left(\left\|\psi^*\right\| + 1\right)\end{aligned} \tag{30}$$

Choosing $Q = I$, and considering the Lyapunov function (30), we obtain

$$\dot{V} \le -cV + \lambda \tag{31}$$

where c and λ are positive constants given by

$c = \min\{1/\lambda_{\max}(P), k, \sigma\gamma_f, \sigma\gamma_g, \sigma\gamma_\psi\}$, $\lambda = \frac{\sigma}{2}\left(\left\|\theta_f^* - \theta_f^0\right\|^2 + \left\|\theta_g^* - \theta_g^0\right\|^2 + \left\|\psi^* - \psi^0\right\|^2\right) + \kappa\varepsilon\left(\left\|\psi^*\right\| + 1\right)$.

Now, if we let $p = \lambda/c$ then (31) satisfies

$$0 \le V(t) \le p + (V(0) - p)\exp(-ct) \tag{32}$$

Therefore $\xi, S, \theta_f, \theta_g$ and ψ are uniformly bounded. Furthermore, using (21) and (30) we obtain that given any $\mu > \sqrt{2p/\lambda_{\min}(P)}$, there exists T such that for all $t \ge T$ error $e(t) = y(t) - y_d(t)$ satisfies $|e| \le \mu, \forall t \ge T$.

Theorem 1. Consider nonlinear system (1) with the sliding mode adaptive tracking design described by the control law (14), the parameter adaptive laws (23), and the bounding parameter adaptive law (28). Suppose Assumption 1 and Assumption 2 hold globally, the following properties can be guaranteed:

(a) all the signals and parameter estimates in the on-line approximation based control scheme are uniformly bounded, and

(b) given any $\mu > \sqrt{2p/\lambda_{\min}(P)}$ there exists T such that for all $t \ge T$, $|e| \le \mu$.

4 H_∞ Tracking-Based Sliding Mode Adaptive Controller Using RBF Neural Networks

In this section, to relax the assumption, we adopt the tracking design technique because the disturbances are bounded rather than explicitly known. The neural networks are used to approximate the unknown nonlinear functions of the dynamical systems through tuning by the adaptive laws. Subsequently, the H_∞ tracking design technique and the sliding mode control method are incorporated into the adaptive neural networks control scheme to derive the control law. Compared with conventional sliding mode control approaches which generally require prior knowledge on the upper bound of the uncertainties, the proposed approach not only assures closed-loop stability, but also guarantees a desired tracking performance for the overall system based on a much relaxed assumption.

We choose u_c and u_N in (14) as follows [11], [12]

$$u_c = u_{equ} + u_{H_\infty} \tag{33}$$

$$u_N = -\beta/\hat{g}(x, \theta_g) \tag{34}$$

where $u_{H_\infty} = \left[S/(2\gamma^2)\right]/\hat{g}(x)$. Since $\rho_f \ge 0$, $\rho_g \ge 0$, letting $\varepsilon = 0$ makes w in (27) as

$$w = [w_1, w_2]^T = \left[\rho_f \operatorname{sgn}(z), \rho_g |u_c| \operatorname{sgn}(z)\right]^T \tag{35}$$

If we set $\sigma = 0$, we can easily check from (22) that

$$\dot{V} \le -\frac{1}{2}\xi^T Q\xi - \frac{1}{2}kS^2 - \frac{S^2}{2\gamma^2} - Sd \le -\frac{1}{2}X^T\widetilde{Q}X + \frac{1}{2}\gamma^2 d^2 \tag{36}$$

where $\widetilde{Q} = \begin{bmatrix} Q & 0 \\ 0 & k \end{bmatrix}$, $X = [\xi, S]^T$. We integrate (36) from $t = 0$ to $t = T$, and obtain

$$\frac{1}{2}\int_0^T X^T(\xi,S)\widetilde{Q}X(\xi,S)dt \le V(0) + \frac{1}{2}\gamma^2\int_0^T d^2(t)dt \tag{37}$$

From (37), we have the H_∞ tracking performance.

Theorem 2. Consider nonlinear system (1) with the sliding mode adaptive tracking design described by the control law (14), the parameter adaptive laws (23) ($\sigma = 0$), control laws (33) and (34), and the bounding parameter adaptive law (28) ($\sigma = 0$). Suppose Assumption 1 and Assumption 2 hold globally, the following property can be guaranteed:

For a prescribed attenuation level γ, the sliding mode adaptive neural networks output feedback control can maintain the H_∞ tracking performance (37) achieved under the state feedback controller.

5 Simulation

Here, a simple is presented to show the effectiveness of the approach proposed above. The robust sliding mode adaptive neural networks controller is applied to control the inverted pendulum to track a sinewave trajectory. The dynamic equations of the system are given by [8], [13]

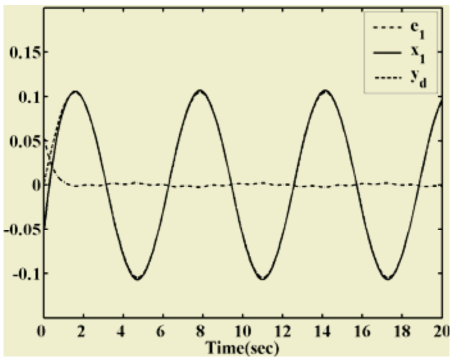
$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = \frac{g\sin(x_1) - mlx_2^2\cos(x_1)\sin(x_1)/(m_c+m)}{l(4/3 - m\cos^2(x_1)/(m_c+m))} + \frac{\cos(x_1)/(m_c+m)}{l(4/3 - m\cos^2(x_1)/(m_c+m))}u \tag{38}$$

We choose $m_c = 1kg$, $m = 0.1kg$ and $l = 0.5m$. The reference signal is chosen as $y_d = (\pi/30)\sin(t)$. Disturbance is chosen as $d = 0.5\sin(5t)$ in the following simulations. The nominal functions $f_0(x)$ and $g_0(x)$ are simply set to zero. In this paper, all the basis functions of the neural networks have the form

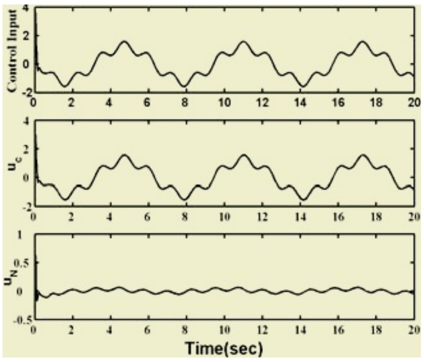
$$G(x_i) = \exp\left[-(x_i - c_i)^2/v_i^2\right] \tag{39}$$

c_i is the center of the receptive field and v_i is the width of the Gaussian function. The neural networks $\theta_f^T\xi_f(x)$ and $\theta_g^T\xi_g(x)$ all contain 25 nodes, with centers c_i evenly spaced in $[-\pi/6, \pi/6]$, and $v_i = \pi/24$ widths. The initial weights θ_f and θ_g are given arbitrarily in [0, 1] and [1, 1.2] respectively. The initial values are $\theta_{fk}^0 = 0, \theta_{gk}^0 = 1$ and $\psi^0 = 0$ for all $k = 1, \ldots, 25$ where θ_{fk}^0 and θ_{gk}^0 denote the kth element of the vector θ_f^0 and θ_g^0, respectively. The initial state is $x(0) = [-0.05, 0]^T$. The controller parameters are selected as $k = 1$, $k_1 = 2$, $k_2 = 1$, $Q = 2$, $P = 0.5$. The adaptation rate constants γ_f, γ_g, and γ_ψ are set to 1000, 10,1 respectively. For comparison purposes, sliding mode adaptive neural networks controller and H_∞ tracking-based sliding mode adaptive neural networks controller for this inverted pendulum tracking problem are simulated. The first proposed controller with $\varepsilon = 0.05$ and $\sigma = 0.01$ is used. The second method is also used the proposed controller but with $\varepsilon = \sigma = 0$, $\gamma = 0.1$

Fig. 1.a shows the simulation results of the state x_1 , reference signal y_d , tracking errors e_1 using the first control method. The control input u , control terms, i.e. u_c , u_N in the first control method are depicted in Fig. 1.b. From the results, it can be inferred that the system output tracks the desired output well by the first proposed controller. Fig. 2.a shows the simulation results of the state x_1 , reference signal y_d , tracking errors e_1 using the second control method. The control input u , control terms, i.e., u_c , u_N in the second control method are depicted in Fig. 2.b. Fig. 2.a also shows that H_∞ tracking performance is achieved.

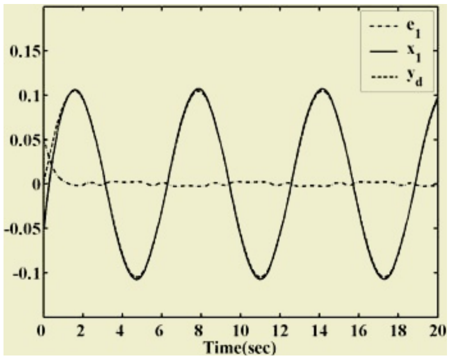


a)

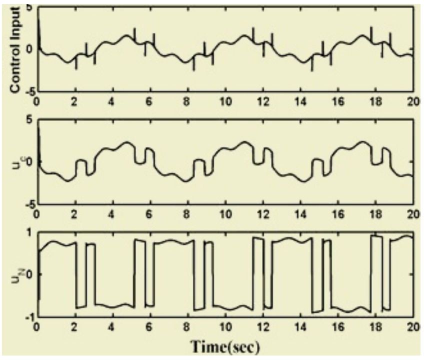


b)

Fig. 1. a) Results of the state x_1, reference signal y_d, tracking errors e_1 in the first method b) Control input and control terms u_c , u_N in the first method



a)



b)

Fig. 2. a) Results of the state x_1, reference signal y_d , tracking errors e_1 in the second method b) Control input and control terms u_c , u_N in the second method

6 Conclusions

A robust sliding mode adaptive tracking controller using RBF neural networks is proposed for uncertain single-input single-output nonlinear dynamical systems with unknown nonlinearities. The control systems can guarantee that the tracking errors converge in the small neighborhood of zero and that all signals involved are uniformly bounded. Furthermore, to facilitate the design process, an H_∞ design algorithm, which can be computerized to derive the H_∞ tracking-based sliding mode adaptive tracking controller using neural networks for the uncertain nonlinear system, is also presented. The proposed approach not only assures closed-loop stability, but also guarantees the H_∞ tracking performance for the overall system based on a much relaxed assumption without prior knowledge on the upper bound of the uncertainties.

References

1. Polycarpou, M.: Stable Adaptive Neural Control Scheme for Nonlinear Systems. IEEE Trans. Automat. Contr., **41** (1996) 447–451
2. Li, Y.H., Qang, Y., Zhuang, X.Y., and Kaynak, O.: Robust and Adaptive Backstepping Control for Nonlinear Systems Using RBF Neural Networks. IEEE Trans. Neural Networks, **15** (2004) 693–701
3. Leu, Y. G., Wang, W.Y., and Lee, T.T.: Robust Adaptive Fuzzy-Neural Controllers for Uncertain Nonlinear Systems. IEEE Trans. Robot. Automat., **15** (1999) 805–817
4. Tzirkel-Hancock, E., and Fallside, F.: Stable Control of Nonlinear Systems Using Neural Networks. Int. J. Robust Nonlinear Control, **2** (1992) 63–86
5. Lue, Y.G., Lee, T.T., and Wang, W. Y.: Observer-Based Adaptive Fuzzy Neural Control for Unknown Nonlinear Dynamical Systems. IEEE Trans. Syst., Man, Cybern. B, **29** (1999) 583–591
6. Ge, S. S., and Wang, C.: Direct Adaptive NN Control of a Class of Nonlinear Systems. IEEE Trans. Neural Networks, **13** (2002) 214–221
7. Yoo, B. and Ham, W.: Adaptive Fuzzy Sliding Mode Control of Nonlinear System. IEEE Trans. Fuzzy Syst., **6** (1998) 315–321
8. Park, J.H., Seo, S.J., and Park, G.T.: Robust Adaptive Fuzzy Controller for Nonlinear System Using Estimation of Bounds for Approximation Errors. Fuzzy Sets and Systems, **133** (2003) 19–36
9. Sanner, R.M., and Slotine, J.J.E.: Gaussian Networks for Direct Adaptive Control. IEEE Trans. Neural Networks, **3** (1992) 837–863
10. Yang, Y.S., Zhou, C.J., and Ren, J.S.: Model Reference Adaptive Robust Fuzzy Control for Ship Steering Autopilot with Uncertain Nonlinear Systems. Applied Soft Computing, **3** (2003) 305–316
11. Chen, B. S., Lee, C.H., and Chang, Y.C.: H_∞ Tracking Design of Uncertain Nonlinear SISO Systems: Adaptive Fuzzy Approach. IEEE Trans. Fuzzy Syst., **4** (1996) 32–43
12. Wang, W.Y., Chan, M.L., Hsu, C.C.J., and Lee, T.T.: H_∞ Tracking-Based Sliding Mode Control for Uncertain Nonlinear Systems via an Adaptive Fuzzy-Neural Approach. IEEE Trans. Syst., Man, Cybern. B, **32** (2002) 483–491
13. Wang, L.X.: Stable Adaptive Fuzzy Controllers with Application to Inverted Pendulum Tracking. IEEE Trans. Syst., Man Cybern. B, **26** (1996) 677–691

Adaptive Backstepping Neural Network Control for Unknown Nonlinear Time-Delay Systems[⋆]

Weisheng Chen and Junmin Li

Department of Applied Mathematics, Xidian University, Xi'an, 710071, China
wshchen@126.com

Abstract. For a class of unknown nonlinear time-delay systems, an adaptive neural network (NN) control design is proposed in this paper. Bacsteppping, domination and adaptive bounding design technique are combined to construct an adaptive NN controller. Unknown time-delay functions are approximated by NNs. Based on the Lyapnov-Krasovikii functional, the sem-globally uniformly ultimately boundedness (SGUUB) of all the signals in the closed-loop systems is proved. The arbitrary output tracking accuracy is achieved by tuning the design parameters. The feasibility is investigated by an illustrative simulation example.

1 Introduction

Neural network control has made great progress in past decades[1-4]. In [1], adaptive bounding technique is applied to adaptive NN control for a class of strict-feedback nonlinear systems. By introducing an integral Lyapunov function in [2], an adaptive NN control approach is proposed for strict-feedback nonlinear systems with unknown nonlinearies, where the controller singularity problem is avoided. In [3], the state feedback and output feedback adaptive NN controllers are presented for a class of strict-feedback discrete-time nonlinear systems, where long standing noncausal problem is avoided. Adaptive NN control approach is extended to output-feedback nonlinear systems in [4].

Recently robust control of nonlinear systems with time delays has been addressed by many researchers. Some interesting results are obtained [5-6]. In [5], output feedback stabilization algorithm for a class of time-delay nonlinear systems is presented. In [6], an adaptive NN control approach is presented for a class of strict-feedback nonlinear time-delay systems with unknown virtual control coefficients, where the NNs are only used to approximate delay-independent functions and the nonlinear time-delay functions are assumed to have known upper bound functions. However, to the author's knowledge, it is merely considered that the NNs are nonlinear functions in adaptive NN control.

Motivated by previous works on both adaptive NN control and robust control for time-delay systems, we extend the adaptive NN control to a class of unknown nonlinear time-delay systems in this paper. Unknown time-delay functions are approximated by NNs. Based on Lyapunov-Krasovskii functional, the SGUUS of all the signals in the closed-loop system is proved. The feasibility is investigated by an illustrative simulation example.

[⋆] This work is supported by NSFC (60374015).

2 Problem Statement and Preliminaries

Consider the following unknown nonlinear time-delay system with the structure

$$\begin{cases} \dot{x}_i = x_{i+1} + f_i(\bar{x}_i) + h_i(y(t-\tau)) + \mu_i(t), \quad 1 \le i \le n-1, \\ \dot{x}_n = g(x)u + f_n(x) + h_n(y(t-\tau)) + \mu_n(t), \\ y = x_1. \end{cases} \tag{1}$$

where $x \in R^n, u \in R$ and $y \in R$ represent the system state, control input and output respectively. $\bar{x}_i = [x_1, ..., x_i]^T \in R^i, (1 \le i \le n-1).g(x)$ is known smooth function. $f_i(.), h_i(.), (1 \le i \le n)$ are unknown smooth functions. The uncertain terms $\mu_i(t), (1 \le i \le n)$ are bounded by an unknown constant. Time delay τ is assumed to be known.

The control objective is to design an adaptive NN controller to track a given reference signal $y_r(t)$.

On compact sets $\Omega_i \subset R^i, (1 \le i \le n)$ and Ω_1, $f_i(\bar{x}_i)$ and $h_i(y(t-\tau))$ can be approximated by the following linearly parameterized neural networks respectively.

$$\begin{cases} f_i(\bar{x}_i, \theta_{f_i}) = \xi_i(\bar{x}_i)^T \theta_{f_i} + \varepsilon_{f_i}(\bar{x}_i), \\ h_i(y(t-\tau), \theta_{h_i}) = \zeta_i(y(t-\tau))^T \theta_{h_i} + \varepsilon_{h_i}(y(t-\tau)). \end{cases} \tag{2}$$

where $\xi_i(.) : \Omega_i \to R^{p_i}$ and $\zeta_i(.) : \Omega_1 \to R^{q_i}$ are known smooth basis function vectors, $p_i, q_i, (1 \le i \le n)$ are the NN node numbers. $\theta_{f_i \in R^{p_i}}$ and $\theta_{h_i \in R^{q_i}}$ are the optimal weights defined as

$$\begin{aligned} \theta_{f_i} &= \arg \min_{\hat{\theta}_{f_i} \in R^{p_i}} \{ \sup_{\bar{x}_i \in \Omega_i} |f_i(\bar{x}_i) - \xi_i(\bar{x}_i)^T \hat{\theta}_{f_i}| \} \\ \theta_{h_i} &= \arg \min_{\hat{\theta}_{h_i} \in R^{q_i}} \{ \sup_{y(t-\tau)) \in \Omega_1} |h_i(y(t-\tau)) - \zeta_i(y(t-\tau))^T \hat{\theta}_{h_i}| \} \end{aligned} \tag{3}$$

Substituting (2) into (1), we obtain

$$\begin{cases} \dot{x}_i = x_{i+1} + \phi_i(\bar{x}_i)^T \theta_1 + \varphi_i(y(t-\tau))^T \theta_2 + \upsilon_i, \quad 1 \le i \le n-1, \\ \dot{x}_n = g(x)u + \phi_n(x)^T \theta_1 + \varphi_n(y(t-\tau))^T \theta_2 + \upsilon_n, \\ y = x_1. \end{cases} \tag{4}$$

where, the network reconstruction errors are defined as $\upsilon_i := \varepsilon_{f_i} + \varepsilon_{h_i} + \mu_i$,

$$\begin{aligned} \theta_1 &= [\theta_{f_1}^T, ..., \theta_{f_n}^T]^T, \theta_2 = [\theta_{h_1}^T, ..., \theta_{h_n}^T]^T, \\ \phi_i &= [0_{1 \times (\sum_{j=1}^{i-1} p_j)} | \xi_{i(1 \times p_i)}^T | 0_{1 \times (\sum_{j=I+1}^{n} p_j)}]^T, \\ \varphi_i &= [0_{1 \times (\sum_{j=1}^{i-1} q_j)} | \xi_{i(1 \times q_i)}^T | 0_{1 \times (\sum_{j=I+1}^{n} q_j)}]^T. \end{aligned}$$

In this paper, we make the following Assumptions

Assumption 1[1]: On the compact sets $\Omega_i \subset R^i$ and Ω_1, the network reconstruction errors $|\upsilon_i| \le \psi, i = 1, ...n$, where $\psi \ge 0$ is the unknown constant.

Assumption 2[7]: The reference signal $y_r(t)$ and its first n derivatives are known and uniformly bounded in the interval $[-\tau, +\infty)$.

Remark 2.1 According to the mean value theorem, the assumption 2 implies the following inequality holds

$$|\varphi_i(y) - \phi_i(y - y_r)| \leq |y - y_r| s_i(y - y_r), 1 \leq i \leq n. \tag{5}$$

where $s_i(.)$ are known smooth nonnegative functions. The inequality(5) will be used to be deal with the nonlinear time delay terms.

Throughout this paper, $\hat{*}$ denotes the estimate of $*$, and $\tilde{*} := * - \hat{*}$, $|x| := (\sum_{i=1}^n x_i^2)^{\frac{1}{2}}, x \in R^n$.

3 Adaptive NN Controller Design

In the section, we will employ backstepping, domination and adaptive bounding technique to design the adaptive NN tracking controller.

We define a change of coordinates as follows

$$\begin{cases} z_1 = y - y_r, \\ z_i = x_i - \alpha_{i-1} - y_r^{(i-1)}, i = 2, ..., n. \end{cases} \tag{6}$$

From (4) and (6), we select the stabilizing functions

$$\begin{cases} \alpha_1 = -a_{11} z_1 - \omega_{\theta_1,1}^T \hat{\theta}_1 - \omega_{\theta_2,1}^T \hat{\theta}_2 - \omega_{\psi,1} \hat{\psi}, \\ \alpha_i = -z_{i-1} - a_{ii} z_i - \Delta_i - \sum_{j=2}^{i-1} \sigma_{j,i} z_j - \omega_{\theta_1,i}^T \hat{\theta}_1 - \omega_{\theta_2,i}^T \hat{\theta}_2 - \omega_{\psi,i} \hat{\psi}, \\ \qquad i = 2, ..., n. \end{cases} \tag{7}$$

where $a_{11} = c_1 + W(z_1) + \frac{1}{2}\hat{\Theta}\omega_{\Theta,1}$, $a_{ii} = c_i + \frac{1}{2}\hat{\Theta}\omega_{\Theta,i}$, $c_i, d_i \geq 0$ are the design parameters,$W(z_1) = \frac{1}{2}\sum_{i=1}^n (\sum_{j=1}^i s_j^2(z_1))$, Θ denotes $|\theta_2|^2$, and

$$\begin{aligned}
&\omega_{\theta_1,1} = \phi_1(x_1), \qquad \omega_{\theta_1,i} = \phi_i(\bar{x}_i) - \sum_{j=1}^{i-1} \frac{\partial \alpha_{i-1}}{\partial x_j} \phi_j(\bar{x}_i), \quad 2 \leq i \leq n, \\
&\omega_{\theta_2,1} = \varphi_1(y_r(t-\tau)), \omega_{\theta_2,i} = \varphi_i(y_r(t-\tau)) - \sum_{j=1}^{i-1} \frac{\partial \alpha_{i-1}}{\partial x_j} \varphi_j(y_r(t-\tau)), 2 \leq i \leq n, \\
&\omega_{\Theta,1} = 1, \qquad \omega_{\Theta,i} = 1 + \sum_{k=1}^{i-1} (\frac{\partial \alpha_{i-1}}{\partial x_k})^2, \quad 2 \leq i \leq n, \\
&\omega_{\psi,1} = \beta_1 \tanh(\tfrac{\beta_1 z_1}{\varepsilon}), \ \omega_{\psi,i} = \beta_i \tanh(\tfrac{\beta_i z_i}{\varepsilon}), \quad \varepsilon > 0, \quad 2 \leq i \leq n, \\
&\beta_1 = 1, \qquad , \beta_i = 1 + \sum_{j=1}^{i-1} (\tfrac{1}{2} + \tfrac{1}{2}(\tfrac{\partial \alpha_{i-1}}{\partial x_j})^2), \quad 2 \leq i \leq n, \\
&\Delta_i = -\sum_{k=1}^{i-1} \frac{\partial \alpha_{i-1}}{\partial x_k} x_{k+1} - \sum_{j=1}^{i-1} \frac{\partial \alpha_{i-1}}{\partial y_r^{(j-1)}(t-\tau)} y_r^{(j)}(t-\tau) - \sum_{j=1}^{i-1} \frac{\partial \alpha_{i-1}}{\partial y_r^{(j-1)}(t)} y_r^{(j)}(t) \\
&\qquad - \sum_{k=1}^{2} \frac{\partial \alpha_{i-1}}{\partial \hat{\theta}_k} \Gamma_{\theta_k}(\tau_{\theta_k} - \iota \hat{\theta}_k) - \frac{\partial \alpha_{i-1}}{\partial \hat{\psi}} \gamma_\psi (\tau_{\psi,i} - \iota \hat{\psi}) - \frac{\partial \alpha_{i-1}}{\partial \hat{\Theta}} \gamma_\Theta (\tau_{\Theta,i} - \iota \hat{\Theta}), \\
&\qquad\qquad 2 \leq i \leq n, \quad \Gamma_{\theta_k} > 0, \quad \gamma_\psi > 0, \quad \gamma_\Theta > 0, \quad \iota > 0, \\
&\sigma_{i,j} = -\sum_{k=1}^{2} \frac{\partial \alpha_{i-1}}{\partial \hat{\theta}_k} \Gamma_{\theta_k} \omega_{\theta_k,j} - \frac{\partial \alpha_{i-1}}{\partial \hat{\psi}} \gamma_\psi \omega_{\psi,j} - \frac{1}{2} \frac{\partial \alpha_{i-1}}{\partial \hat{\Theta}} \gamma_\Theta \omega_{\Theta,j} z_j, \quad 2 \leq i \leq n,
\end{aligned}$$

We select the tuning functions

$$\begin{cases} \tau_{\theta_1,1} = \omega_{\theta_1,1} z_1, & \tau_{\theta_1,i} = \tau_{\theta_1,i-1} + \omega_{\theta_1,i} z_i, \quad 2 \le i \le n, \\ \tau_{\theta_2,1} = \omega_{\theta_2,1} z_1, & \tau_{\theta_2,i} = \tau_{\theta_2,i-1} + \omega_{\theta_2,i} z_i, \quad 2 \le i \le n, \\ \tau_{\psi,1} = \omega_{\psi,1} z_1, & \tau_{\psi,i} = \tau_{\psi,i-1} + \omega_{\psi,i} z_i, \quad 2 \le i \le n, \\ \tau_{\Theta,1} = \frac{1}{2}\omega_{\Theta,1} z_1^2, & \tau_{\Theta,i} = \tau_{\Theta,i-1} + \frac{1}{2}\omega_{\Theta,i} z_i^2, \quad 2 \le i \le n. \end{cases} \tag{8}$$

and the last tuning functions $\tau_{\theta_1,n}, \tau_{\theta_2,n}, \tau_{\psi,n}, \tau_{\Theta,n}$ are used in the adaptive law

$$\begin{cases} \dot{\hat{\theta}}_1 = \Gamma_{\theta_1}(\tau_{\theta_1,n} - \iota\hat{\theta}_1), \\ \dot{\hat{\theta}}_2 = \Gamma_{\theta_2}(\tau_{\theta_2,n} - \iota\hat{\theta}_2), \\ \dot{\hat{\psi}} = \gamma_\psi(\tau_{\psi,n} - \iota\hat{\psi}), \\ \dot{\hat{\Theta}} = \gamma_\Theta(\tau_{\Theta,n} - \iota\hat{\Theta}). \end{cases} \tag{9}$$

The last stabilizing function α_n is used in the actual control law

$$u = \frac{1}{g(x)}(\alpha_n + y_r^{(n)}(t)). \tag{10}$$

From (4),(6),(7) and (10),the closed-loop system is written as

$$\begin{cases} \dot{z}_1 = -a_{11} + z_2 + \omega_{\theta_1,1}^T\tilde{\theta}_1 + \omega_{\theta_2,1}^T\tilde{\theta}_2 - \omega_{\psi,1}\tilde{\psi} + \Lambda_1 + \varsigma_1 \\ \dot{z}_i = -z_{i-1} - a_{ii}z_i + z_{i+1} - \sum\limits_{j=2}^{i-1}\sigma_{j,i}z_j + \sum\limits_{j=i+1}^{n}\sigma_{i,j}z_j \\ \qquad + \sum\limits_{k=1}^{2}\omega_{\theta_k,i}^T\tilde{\theta}_k - \omega_{\psi,i}\hat{\psi} + \Lambda_i + \varsigma_i, \qquad 2 \le i \le n. \end{cases} \tag{11}$$

where $\alpha_n := g(x)u - y_r^n$, $z_{n+1} := 0$,

$\varsigma_1 = \upsilon_1, \quad \varsigma_i = \upsilon_i - \sum\limits_{j=1}^{i-1}\frac{\partial\alpha_{i-1}}{\partial x_j}\upsilon_j, \quad \Lambda_1 = (\varphi_1(y(t-\tau)) - \varphi_1(y_r(t-\tau)))^T\theta_2,$

$\Lambda_i = [\varphi_i(y(t-\tau)) - \varphi_i(y_r(t-\tau)) - \sum\limits_{j=1}^{i-1}\frac{\partial\alpha_{i-1}}{\partial x_j}(\varphi_j(y(t-\tau)) - \varphi_j(y_r(t-\tau)))]^T\theta_2$

Theorem 1 The cosed-loop adaptive system consisting of the plant (1), the adaptive Laws (9) and the control law (10) has following properties:

(I) The tracking error satisfies

$$\lim_{t\to\infty}[(\int_0^t z_1^2(\sigma)d\sigma)/t] \le \frac{\lambda}{c_0} \tag{12}$$

where $\lambda = n\kappa\psi\varepsilon + \frac{1}{2}\iota(|\theta_1|^2 + |\theta_2|^2 + \Theta^2 + \psi^2)$, $c_0 = \min\{c_1, ..., c_n\}$.

(II) All the signals are sem-glabally uniformly ultimately bounded.

Proof For system (11), by Young'Inequality, the time-delay terms Λ_i can be dealt with as follows

$$z_i\Lambda_i \le \frac{1}{2}|\theta_2|^2 z_i^2(1 + \sum_{j=1}^{i-1}(\frac{\partial\alpha_{i-1}}{\partial x_j})) + \frac{1}{2}z_1^2(t-\tau)\sum_{j=1}^{i}s_j^2(z_1(t-\tau)), \quad 1 \le i \le 2. \tag{13}$$

where $\frac{\partial\alpha_0}{\partial x_j} := 0$

By using inequality $|\eta| \leq \eta \tanh(\eta/\varepsilon) + \kappa\varepsilon$ [1],$\kappa = 0.2785, \varepsilon > 0$, the network reconstruction terms ς_i can be dealt with as follows

$$\begin{aligned} z_i\varsigma_i &\leq |z_i|(|\upsilon_i| + \sum_{j=1}^{i-1} |\tfrac{\partial \alpha_{i-1}}{\partial x_j}||\upsilon_j|) \\ &\leq \psi z_i \beta_i \tanh(z_i\beta_i/\varepsilon) + \psi\kappa\varepsilon \\ &= \psi z_i \omega_{\psi,i} + \psi\kappa\varepsilon, \quad 1 \leq i \leq n. \end{aligned} \tag{14}$$

Defined a Lyapunov-Krasowiskii functional as

$$V = \frac{1}{2}z^T z + \frac{1}{2}\tilde{\theta}_1^T \Gamma_{\theta_1}^{-1}\tilde{\theta}_1 + \frac{1}{2}\tilde{\theta}_2^T \Gamma_{\theta_2}^{-1}\tilde{\theta}_2 + \frac{1}{2}\gamma_\psi^{-1}\tilde{\psi}^2 + + \frac{1}{2}\gamma_\Theta^{-1}\tilde{\Theta}^2 + \int_{t-\tau}^{t} S(z_1(\sigma))d\sigma \tag{15}$$

where $z = [z_1, ..., z_n$, $S(z_1(\sigma)) = z_1^2(\sigma)W(z_1(\sigma))$.

From (9),(11),(13),(14)and(15), and observing $\iota\tilde{\theta}_1^T\hat{\theta}_1 + \iota\tilde{\theta}_2^T\hat{\theta}_2 + \iota\tilde{\psi}\psi + \iota\tilde{\Theta}\hat{\Theta} \leq -\frac{1}{2}\iota|\tilde{\theta}_1|^2 - \frac{1}{2}\iota|\tilde{\theta}_2|^2 - \frac{1}{2}\iota\tilde{\psi}^2 - \frac{1}{2}\iota\tilde{\Theta}^2 + \frac{1}{2}\iota|\theta_1|^2 + \frac{1}{2}\iota|\theta_2|^2 + \frac{1}{2}\iota\psi^2 + \frac{1}{2}\iota\Theta^2$, we have

$$\begin{aligned} \dot{V} \leq &- \sum_{i=1}^{n} c_i z_i^2 - \tfrac{1}{2}\iota|\tilde{\theta}_1|^2 - \tfrac{1}{2}\iota|\tilde{\theta}_2|^2 - \tfrac{1}{2}\iota\tilde{\psi}^2 - \tfrac{1}{2}\iota\tilde{\Theta}^2 \\ &+ \tfrac{1}{2}\iota|\theta_1|^2 + \tfrac{1}{2}\iota|\theta_2|^2 + \tfrac{1}{2}\iota\psi^2 + \tfrac{1}{2}\iota\Theta^2 + n\psi\kappa\varepsilon \end{aligned} \tag{16}$$

Let $\pi := [z^T, \tilde{\theta}_1, \tilde{\theta}_2, \tilde{\psi}, \tilde{\Theta}]$,and we get

$$\dot{V}(\pi) \leq -\bar{c}||\pi||^2 + \lambda \tag{17}$$

where $\bar{c} = \min\{c_0, \frac{1}{2}\iota\}$.

In the light of the Lyapunov stability theory, (17) implies that π is bounded. From (6) and assumption 2, we can see that x, u are bounded. (14) can be obtained by integrating (16).

4 Simulation Study

Consider the following unknown nonlinear time-delay system

$$\begin{cases} \dot{x}_1 = x_2 + \frac{x_1 x_1^3}{x_1^4+1} + 0.005\sin(t) \\ \dot{x}_2 = u - \sin(5y(t-\tau))e^{-y^2(t-\tau)} - 0.1\cos(t) \\ y = x_1. \end{cases} \tag{18}$$

where time delay $\tau = 5$. Reference signal is chosen as $y_r = \sin(0.5t)\sin(0.2t)$.

Based on the design method proposed in this paper, in our simulation, the initial condition are set to $x_1(\sigma) = -0.2, \ -\tau \leq \sigma \leq 0, \ x_2(0) = 0.5, \ \hat{\theta}_1(0) = \hat{\theta}_2(0) = \hat{\psi(0)} = \hat{\Theta}(0) = 0$. The adaptive gains and control parameters are chosen as $\Gamma_{\theta_1} = 0.8I$, $\Gamma_{\theta_2} = 1.6I$, $\gamma_\theta = 0.01$, $\gamma_\psi = 0.0003$, $c_1 = c_2 = 0.01$, $\iota = 0.0001$, $\varepsilon = 0.01$. The function approximators are RBF networks and the basis function is chosen as $\phi_i = \phi_i^0(z)/\sum_{j=1}^N \phi_j^0(z)$, where $\phi_j^0(z) = e^{-(z-\eta_i)^2} \ \bar{\varsigma}$, $\bar{\varsigma} = \frac{1}{100\ln(2)}$, $N = 11$. η_i is a center of the ith basis function, $\Omega_1 = [-1, 1]$. Simulation result are shown in figure 1.

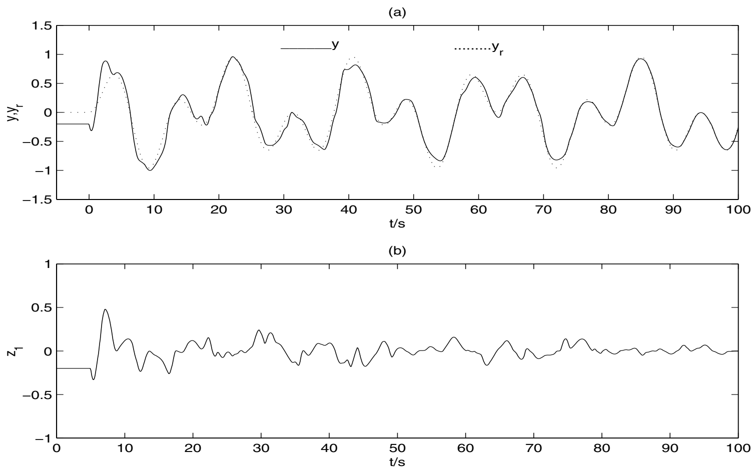


Fig. 1. The histories of the output $y(t)$, the reference signal $y_r(t)$ and the tracking error curve $z_1(t)$.

5 Conclusions

In this paper, an adaptive NN controller is proposed for a class of unknown nonlinear time-delay systems. NNs are used to approximate delay-free unknown terms and delay-dependent unknown terms. The SGUUB of the closed-loop systems is obtained. The feasibility is investigated by an illustrative simulation example.

References

1. Policarpou, M. M., Mears, M. J.: Stable Adaptive Tracking of Uncertain Systems Using Nonlinearly Parameterized On-line Approximators. Int. J. Control. **70** (1998) 363–384
2. Zhang, S. S., Ge, S. S., Hang, C. C. : Stable Adaptive Control for A Class of Nonlinear Systems Using A Modified Lyapunov Function. IEEE Trans. Automatic Control. **45** (2000) 125–132
3. Ge, S. S., Li, G. Y., Lee, T. H. : Adaptive NN Control for A Class of Strict Feedback Discrete Time Nonlinear Systems. Automatic. **39** (2003) 807–819
4. Jin, Y. C., Farrell, J. : Adaptive Observer Backstepping Control Using Neural Network. IEEE Trans. Neural Networks. **12** (2001) 1103–1112
5. Fu, Y. S., Tian, Z. H. : Output Feedback Stabilizaion for Time-Delay System. Acta Automatic Sinia. **28** (2002) 802–805
6. Ge, S. S., Li, G. Y., Lee, T. H. : Adaptive Neural Network Control of Nonlinear Systems with Unknown Time Delays. IEEE Trans. Automatic Control. **48** (2003) 2004–2010
7. Krstic, M., KanellaKopuos, I., Kokotovic, P. V. : Nonlinear and Adaptive Control Design. Springer-Verlag, Berlin Heidelberg New York (1995)

Multiple Models Adaptive Control Based on RBF Neural Network Dynamic Compensation

Junyong Zhai and Shumin Fei

Research Institute of Automation, Southeast University, Nanjing, Jiangshu 210096, China

Abstract. A novel multiple models adaptive control method is proposed to improve the dynamic performance of complex nonlinear systems under different operating modes. Multiple linearized models are established at each equilibrium point of the system. Each local linearized model is valid within a neighborhood of the point, and then an improved RBF algorithm is applied to compensate for modeling error. Simulation results are presented to demonstrate the validity of the proposed method.

1 Introduction

As it well known that conventional adaptive control algorithms are suitable to systems with slow change parameters. It takes long time after every drastic change in the systems to relearn model parameters and adjust controller parameters subsequently. One way to solve these problems is to use multiple models adaptive control (MMAC). The earliest MMAC appeared in the 70s last century [1],[2], where multiple Kalman Filter-based models were studied to improve the accuracy of the state estimate in estimation and control problems. This kind of multi-model control is always produced as the probability-weighted average of elemental controller outputs, and the stability of closed loop system is difficult to prove, only some convergence results have been obtained. However actual industrial systems are characterized by different operating modes such as new batch of materials, variations in equipment performance, effect of weather conditions, changes in production schedule, etc. The concept of multiple models, switching and tuning was proposed in [3] including the use of multiple identification models and choice of the best model and the corresponding controller at every instant. Based on the idea to describe the dynamics of the system using different models for different operating modes and to devise a suitable strategy for finding a model that is closest to the current plant dynamics in some sense, the model is in turn used to generate the control signal that achieves the desired control objective. In [4] fixed models, adaptive models and switch strategy are applied to improve the transient response of the continuous-time linear systems. The above result is generalized to discrete-time linear systems in [5]. In recent years there has also been a great deal of research activities in extending multiple models approach to the modeling and control of nonlinear systems. Nonlinear adaptive control using neural networks and multiple models is proposed in [6]. A multi-model predictive control scheme is proposed for nonlinear systems based on multi-linear-model representation in [7]. Some on exponential stability of delayed Neural Networks with a general class of activation functions can be found in [8].

In this paper we develop a novel MMAC method to improve the dynamic performance of nonlinear systems. Multiple linearized models are established at each equilibrium point of the system, and then an improved RBF algorithm is applied to compensate for modeling error. The multi-model control scheme depends on the multiple representation of a process using different models, which involves locating the model that best matched the process and generating the control signal that will drive the system to follow the desired trajectory according to the located model.

The paper is organized as follows. A statement of the problem is given in section 2. Improved learning algorithm of RBF Neural Network is proposed in section 3. In section 4 switch control strategy is studied. A simulation study is described in section 5, which shows the validity of the proposed method. Finally conclusion is drawn in section 6.

2 Problem Statement

Let a class of nonlinear systems be described by

$$y(t) = f[y(t-1),\cdots,y(t-n),u(t-1),\cdots,u(t-1-m)] + \zeta(t)\,. \tag{1}$$

where $f(\cdot)$ is a nonlinear function with one order of derivative in succession. $u(t)$, $y(t)$ and $\zeta(t)$ are the system input, output and disturbance at time t respectively. Assuming (1) has several different equilibrium points $(u_i, y_i), i\in\{1,2,\cdots,N\}$. It can be obtained N linearized models using Taylor series expansion at different equilibrium point. The ith linearized model can be represented as

$$A_i(q^{-1})y(t) = q^{-1}B_i(q^{-1})u(t) + d_i \quad i\in\{1,2,\cdots,N\}\,. \tag{2}$$

where q^{-1} is the shift operator, i.e., $q^{-i}\,y(t) = y(t-i)$. $A_i(q^{-1}) = 1+\mathrm{a}_{i,1}q^{-1}+\ldots+\mathrm{a}_{i,n}q^{-n}$, $B_i(q^{-1}) = b_{i,0} + b_{i,1}q^{-1} + \cdots + b_{i,m}q^{-m}$ are polynomials of degrees n and m in q^{-1} respectively.

$$d_i = (\textstyle\sum_{j=1}^{n} a_{i,j})y_i - (\textstyle\sum_{k=0}^{m} b_{i,k})u_i\ ,\ a_{i,j} = -\dfrac{\partial f}{\partial y(t-j)}\Big|_{(u_i,y_i)}\ ,\ b_{i,k} = \dfrac{\partial f}{\partial u(t-1-k)}\Big|_{(u_i,y_i)}\ ,$$

Since Taylor series expansion leaves out each high order items in [7], we introduce RBF Neural Network to compensate for modeling error and the ith linearized model can be represented as

$$A_i(q^{-1})y(t) = q^{-1}B_i(q^{-1})u(t) + d_i + f_{nn}(u,y) \quad i\in\{1,2,\cdots,N\}\,. \tag{3}$$

where $f_{nn}(u,y)$ is the output of the RBF neural network. In the following section an improved learning algorithm of RBF is proposed to approximate the modeling error.

3 Improved Learning Algorithm of RBF

We consider multiple inputs single output (MISO) systems without loss of generality. A structure of RBF comprises three layers. The hidden layer possesses an array of neurons, referred to as the computing nodes. Each hidden node in the network has two parameters called a center vector $C_i = [c_{i1}, c_{i2}, \cdots, c_{in}]^T$ and a width σ_i associated with it. The activation function of the hidden nodes is Gaussian function in general. The output of each hidden node depends on the radial distance between the input vector $X_n = [x_1, x_2, \cdots, x_n]^T$ and the center vector C_i. The response of the ith hidden node to the network input is expressed as

$$\varphi_i(X_n) = \exp\left(-\frac{\|X_n - C_i\|^2}{2\sigma_i^2} \right) . \tag{4}$$

where $\|\bullet\|$ denotes the Euclidean norm. The response of each hidden node is scaled by its connecting weights ω_i to the output node and then summed to produce the overall network output. The overall network output is expressed as

$$f(X_n) = b_0 + \sum_{i=1}^{h} \omega_i \varphi_i(X_n) . \tag{5}$$

where h is the number of hidden nodes in the network. b_0 is the bias term for the output node. The learning process of RBF involves allocation of new hidden node as well as adaptation of network parameters. The network begins with no hidden node. As input-output (X_n, y_n) data are received during training, some of them are used to generate new hidden nodes. The decision as to whether an input-output data should give rise to a new hidden node depends on the novelty in the data, which is decided on the following two conditions

$$d_n = \|X_n - C_{nr}\| \geq \Delta\varepsilon \ . \tag{6}$$

$$|e_n| = |y_n - f(X_n)| \geq \Delta e \ . \tag{7}$$

where C_{nr} is the center vector which is nearest to X_n. d_n is the Euclidean distance between X_n and C_{nr} $\Delta\varepsilon$ and Δe are thresholds to be selected appropriately. If the above two conditions are satisfied then the data is deemed to have novelty and a new hidden node is added. The parameters associated with the new hidden node are as follows

$$C_{h+1} := X_n, \omega_{h+1} := e_n, \sigma_{h+1} := \lambda \cdot d_n . \tag{8}$$

where λ is an overlap factor. When an observation (X_n, y_n) does not satisfy the novelty criteria, a hidden node is not added but the network parameters C_i, σ_i, ω_i are adapted to fit that observation. Platt [9] used the LMS algorithm for adapting C_i, ω_i not including σ_i. An enhancement to Resource Allocation Network (RAN) was proposed in [10], where used an EKF instead of the LMS algorithm for adjusting the network parameters C_i, σ_i, ω_i. However the drawbacks of RAN or RAN+EKF is that once a hidden node is created it can never be removed and consequently produce networks in which some hidden nodes end up contributing little to the whole network output. In [11] a pruning strategy algorithm was proposed. The pruning strategy removes those hidden nodes, which make insignificant contribution to the overall network output consecutively over a number of training observations.

Except for the above cases, with the process of learning there may be some hidden nodes with the approximate center and width, these nodes are said to be redundant nodes, i.e., the following two conditions are satisfied

$$\left\| C_i - C_j \right\| \le \Delta c \,. \tag{9}$$

$$\left| \sigma_i - \sigma_j \right| \le \Delta\sigma \quad i \ne j \in \{1, 2, \cdots, h\} \quad . \tag{10}$$

where Δc and $\Delta\sigma$ are thresholds to be selected appropriately. We can incorporate these hidden nodes, i.e.

$$C_i := (C_i + C_j)/2,\, \sigma_i := (\sigma_i + \sigma_j)/2,\, \omega_i := \omega_i + \omega_j \,. \tag{11}$$

4 Switch Control Strategy

Each model performance index is evaluated over the sliding data window L

$$J_i(t) = \sum\nolimits_{j=t-L+1}^{t} (y_i(j) - y(j))^2 \qquad i \in \{1, 2, \cdots, N\} \,. \tag{12}$$

where $y_i(j)$ and $y(j)$ are the output of the ith model and the actual plant at time j respectively. The next question is how to switch among linearized models. After all linearized models are initiated; the performance indices (12) are calculated at every instant and the system switches to a controller corresponding to the minimum value of the performance indices. The controller corresponding to the minimum value of the performance indices has been designed and put in place by the switching mechanism. This will not yield improved performance if the system exhibits rapid switching between controllers and indeed it is possible that degraded performance could occur. To counter this undesirable effect, a hysteresis term is introduced into the control scheme [12]. Let $J_i(t)$ be the minimum value of performance indices and denote the model used to design the current controller by $J_i(t)$. Then under the added hysteresis term, a switch will only occur if $J_i(t) > J_i(t) + \rho$, where $\rho > 0$ is the hysteresis constant to be selected.

5 Simulation Study

In this section simulations are given to show the validity of the proposed method. Let the nonlinear system can be represented as

$$y(t+2) = y(t+1)^2 + 2y(t+1) - 2y(t+1)y(t) + y(t)^2 + T^2 y(t)^2 - y(t) + T^2 u(t) + 0.05\sin(t)$$

where T is the sampling cycle, which value is one second. At the equilibrium point (u_i, y_i), the ith model can be described as

$$y(t+2) = 2(y(t+1) - y_i) + (2y_i T^2 - 1)(y(t) - y_i) + T^2 (u(t) - u_i) + y_i + f_{nn} \,.$$

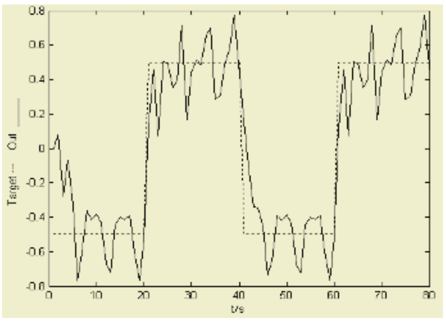


Fig. 1. The response of the system using conventional control

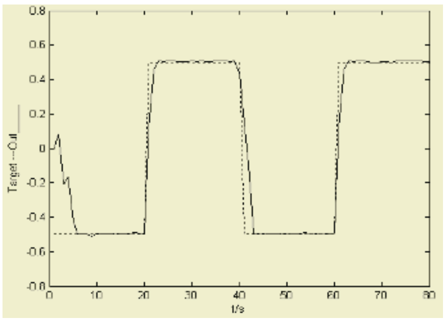


Fig. 2. The response of the system using the proposed method

In the simulation we choose six equilibrium points as follows: (0.5, –0.25), (1.25, –1.5625), (2, –4), (2.75, –7.5625), (3.5, –12.25) and (4, –16). The reference input signal is the square wave with $\tau = 40$ second and amplitude 0.5. Simulation parameters are $\varepsilon = 0.04$, $\Delta e = 0.05$, $\lambda = 0.96$, $L = 12$, $e_{\min} = 0.01$, $\Delta c = 0.01$, $\Delta\sigma = 0.01$, $\delta = 0.01$, $\rho = 0.08$. The response of the system using conventional adaptive control is solid line and the reference signal is dotted line in Fig. 1, whereas that of the system using the proposed method is solid line in Fig. 2 with the same reference signal. Apparently, the conventional adaptive control algorithm cannot deal with this drastic change well. However the method proposed in this paper improves the dynamical response performance.

6 Conclusions

In this paper a novel adaptive control algorithm based on multiple models is proposed to improve the dynamic performance. The main idea of the approach is to use multiple models, which correspond to different operating points. Multiple linearized models are established at respective equilibrium points of the plant. Since each local linear model is valid within a neighborhood of an operating point, and then an improved RBF neural network is applied to compensate modeling error. The simulation results are presented to demonstrate the validity of the proposed method.

Acknowledgement

This work was supported by Higher Education Doctor's Discipline Foundation under Grant No.20030286013.

References

1. Lainiotis, D.: Optimal Adaptive Estimation: Structure and Parameter Adaptation. IEEE Trans. Automatic Control, **16** (1971) 160-170
2. Athans, M., Castanon, D., Dunn, K., et al.: The Stochastic Control of the F-8C Aircraft Using a Multiple Model Adaptive Control (MMAC) Method--Part I: Equilibrium Flight. IEEE Trans. Automatic Control. **22** (1977) 768-780

3. Narendra, K.S., Balakrishnan, J.: Improving Transient Response of Adaptive Control Systems Using Multiple Models and Switching. IEEE Trans. Automatic Control, **39** (1994) 1861-1866
4. Narendra, K.S., Balakrishnan, J.: Adaptive Control Using Multiple Models and Switching. IEEE Trans. Automatic Control, **42** (1997) 171-187
5. Narendra, K.S., Cheng, X.: Adaptive Control of Discrete-Time Systems Using Multiple Models. IEEE Trans. Automatic Control. **45** (2000) 1669-1686
6. Chen, L., Narendra, K.S.: Nonlinear Adaptive Control Using Neural Networks and Multiple Models. Automatica, **37** (2001) 1245-1255
7. Xi, Y., Wang, F.: Nonlinear Multi-Model Predictive Control. Acta Automatica Sinica. **22** (1996) 456-461
8. Sun, C., Zhang, K., Fei, S., et al.: On Exponential Stability of Delayed Neural Networks with a General Class of Activation Functions. Physics Letters A, **298** (2002) 122-132
9. Platt, J.: A Resource-Allocating Network for Function Interpolation. Neural Computation, **3** (1991) 213-225
10. Kadirkamanathan, V., Niranjan, M.: A Function Estimation Approach to Sequential Learning with Neural Network. Neural Computation, **5** (1993) 954-975
11. Lu, Y., Sundararajan, N., Saratchandran, P.: A Sequential Learning Scheme for Function Approximation by Using Minimal Radial Basis Function Neural Networks. Neural Computation, **9** (1997) 461–478
12. Middleton, R.H., Goodwin, G.C., Hill, D.J., et al.: Design Issues in Adaptive Control. IEEE Trans. Automatic Control, **33** (1988) 50-58

Stability Analysis and Performance Evaluation of an Adaptive Neural Controller

Dingguo Chen[1] and Jiaben Yang[2]

[1] Siemens Power Transmission and Distribution Inc.,
7225 Northland Drive, Brooklyn Park, MN 55428, USA
[2] Department of Automation, Tsinghua University,
Beijing 100084, China

Abstract. In this paper, a neural network based adaptive controller is designed for a class of nonlinear systems. The offline neural network training and on-line neural network tuning are integrated to assure that not only the stability of the resulting closed-loop control system is guaranteed, but also the reasonable tracking performance is achieved. The adaptation of the parameters of neural networks is handled based on the robust adaptive control design methodology. The off-line training step incurs additional cost and maybe inconvenience compared to direct on-line neural network parameters tuning. However, the stability analysis and performance evaluation have a more solid basis; and the weight adaptation laws are different than those existing in the literature and bear more practical meaning and significance.

1 Introduction

A new trend has been witnessed in the recent years that neural networks have been introduced into nonlinear adaptive control design [1] to hopefully tackle some of the difficult problems that conventional design approaches can not handle. Direct use of offline trained neural networks for real time control may cause instability. Under certain assumptions, a few results [2, 3] claim that neural networks can be directly trained on-line for real time control purpose. One of the key assumptions is that initial values of the neural network parameters to be tuned on-line are within a prespecified range of their optimal values, which can not be verified. Further without the offline neural network training, it is difficult to have a good choice of the size of the neural network. Based on this observation, we propose to incorporate the knowledge gained from offline neural network training into on-line neural network tuning. As the optimal weights and biases of a neural network are unknown for a particular application, the offline training process provides sub-optimal weights and biases, which are nearly perfect from the practical application point of view. In addition, due to the availability of the weights and biases of the offline trained neural network, the Taylor series expansion of a neural network with ideal weights and biases around the estimates of these parameters, a key step in deriving the parameter adaptation laws and adaptive controller, allows certain terms, that contribute significantly to the total sum and are included in the residual inappropriately [2, 3], to be reorganized in an appropriate manner. As a result, first-order terms used in the approximation are not included in the residual. Consequently, this makes the on-line tuning of the weights and biases of a neural network more effective.

2 SMIB and Problem Formulation

The single-machine infinite-bus (SMIB) model [5] that has been studied takes the following form:

$$\begin{cases} \dot{\delta} = \omega_b(\omega - 1) \\ \dot{\omega} = \frac{1}{M}(P_m - P_l - D(\omega - 1) - \frac{V_t V_\infty}{X_d + (1-s)Xe} \sin\delta) \end{cases} \tag{1}$$

To make our control design applicable to a broader class of nonlinear uncertain systems, we consider a general system characterized by

$$\begin{aligned} \dot{x}_i &= x_{i+1}, i = 1, \cdots, n-1 \\ \dot{x}_n &= a(X) + b(X)u + c(X)^\tau P \end{aligned} \tag{2}$$

and

$$y = x_1 \tag{3}$$

where the state vector $X = \begin{bmatrix} x_1 \; x_2 \cdots x_n \end{bmatrix}^\tau \in \Omega_x \subset R^n$ with Ω_x being a compact subset of R^n, $a(X)$ and $b(X)$ are continuous functions defined on Ω_x, $c(X)$ a continuous vector field of dimension m, P a parameter vector of dimension m, and y the output. Note that $a(X)$ is the unknown nonlinearity involved in the system characterization, $b(X) \neq 0$ for $\forall X \in \Omega_x$, and P is a vector of unknown parameters. The control objective is to design a control law for u such that the state X follows a desired state trajectory as closely as possible regardless of the unknown nonlinearity and unknown parameters.

3 Neural Controller Design

3.1 Taylor Series Expansion of a Neural Network

As is known, a one-hidden layer neural network, when including sufficiently many neurons in the hidden layer, can approximate any continuous function defined on a compact domain at an arbitrary approximation accuracy. With this in mind, we employ one-hidden layer neural networks for control purpose. Suppose that a one-hidden neural network has an input layer of d_i inputs, an output layer of single output, and a hidden layer of d_h neurons. Let the activation function for the hidden layer be $\sigma_h(.)$, and the activation function for the output neuron is a linear function. A neuron n in the hidden layer is connected to all the inputs through d_i connections, each one associated with a weight $w_1^{n,j}$ where $j = 1, 2, \cdots, d_i$. The threshold or the bias for this neuron n is b_1^n. Similarly, the output neuron is connected to the hidden layer through d_h connections, each one associated with a weight w_2^j where $j = 1, 2, \cdots, d_h$. The threshold for the output neuron is b_2. Therefore, this neural network realizes a mapping from the input space Ω_i to R. Let the input vector be denoted by X. Then the realized function can be given by $NN(X) = W_2^\tau \Sigma(W_1^\tau X + B_1) + b_2$ where $W_2 = \begin{bmatrix} w_2^1 \; w_2^2 \cdots w_2^{d_h} \end{bmatrix}^\tau$, $W_1^\tau = \begin{bmatrix} w_1^{1,1} & w_1^{1,2} & \cdots w_1^{1,d_i} \\ w_1^{2,1} & w_1^{2,2} & \cdots w_1^{2,d_i} \\ \cdots & \cdots & \cdots\cdots \\ w_1^{d_h,1} & w_1^{d_h,2} & \cdots w_1^{d_h,d_i} \end{bmatrix}$, $B_1 = \begin{bmatrix} b_1^1 \; b_1^2 \cdots b_1^{d_h} \end{bmatrix}^\tau$.

The above equation can be rewritten in a more compact form as follows: $NN(X) = W_{2a}^\tau \Sigma_a(W_{1a}^\tau X_a)$ where W_{1a}, W_{2a} and Σ_a are properly formed.

Remark 1: $W_{2a} \in R^{d_h+1}$, $W_{1a} \in R^{(d_i+1)\times(d_h+1)}$.

Remark 2: For any function continuous function $f(X)$ defined on a compact set Ω_x, then there exists a one-hidden layer neural network $NN(X)$ such that for any positive $\epsilon > 0$, $|f(X) - NN(X)| = |f(X) - W_{2a}^\tau \Sigma_a(W_{1a}^\tau X_a)| < \epsilon$.

Remark 3: To make a distinction between different choices of neural network parameters, a neural network $NN(X)$ will be changed to $NN(X, \Theta)$ where Θ is a vector which is composed of all the parameters of both W_{2a} and W_{1a}. Hence, in the case of sub-optimal parameterization of the neural network, we have $NN(X, \Theta_s) = W_{2a,s}^\tau \Sigma_a(W_{1a,s}^\tau X_a)$; in the case of optimal parameterization of the neural network, we have $NN(X, \Theta_*) = W_{2a,*}^\tau \Sigma_a(W_{1a,*}^\tau X_a)$; and in the case of parameter estimation of the neural network, we have $NN(X, \hat{\Theta}) = \hat{W_{2a}}^\tau \Sigma_a(\hat{W_{1a}}^\tau X_a)$.

Assumption 1: Let the offline well-trained neural network be denoted by $NN(X, \Theta_s)$, and the ideal neural network by $NN(X, \Theta_*)$. The approximation of $NN(X, \Theta_s)$ to $NN(X, \Theta_*)$ is measured by two known positive numbers $\epsilon_{W_{1a}}$ and $\epsilon_{W_{1a}}$ such that $||W_{1a}^* - W_{1a,s}||_F \le \epsilon_{W_{1a}}$ and $||W_{2a}^* - W_{2a,s}|| \le \epsilon_{W_{2a}}$, where $||(.)||_F = tr\{(.)^\tau(.)\}$ with tr designating the trace of a matrix, representing the Frobenius norm of a matrix.

Notation 1: For the convenience of further development, we define: $\tilde{W}_{1a} = \hat{W}_{1a} - \hat{W}_{1a,*}$ and $\tilde{W}_{2a} = \hat{W}_{2a} - \hat{W}_{2a,*}$.

Theorem 1. *The above defined optimal neural network $NN(X, \Theta_*)$ linearized at its estimated parameter values $\hat{W}_{2a}$ and $\hat{W}_{1a}$ can be rewritten as:*

$$\begin{aligned} NN(X, \Theta_*) = {} & NN(X, \hat{\Theta}) - \hat{W}_{2a}^\tau \hat{\Sigma}_a' \tilde{W}_{1a}^\tau X_a \\ & - \tilde{W}_{2a}^\tau (\hat{\Sigma}_a - \hat{\Sigma}_a' (\hat{W}_{1a} - W_{1a,s})^\tau X_a) + r_x \end{aligned} \tag{4}$$

where $\hat{\Sigma}_a = \Sigma_a(\hat{W_{1a}}^\tau X_a)$, $\hat{\Sigma}_a' = diag\{D_{\hat{\Sigma}_a}\}$ with $D_{\hat{\Sigma}_a} = \begin{bmatrix} \sigma'((\hat{W}_{1a}^1)^\tau X_a) \; \sigma'((\hat{W}_{1a}^2)^\tau X_a) \cdots \sigma'((\hat{W}_{1a}^{d_i})^\tau X_a) \; 0 \end{bmatrix}$, and $r_x = W_{2a,*} o(\tilde{W}_{1a}^\tau X_a) - \tilde{W}_{2a}^\tau \hat{\Sigma}_a' ((\hat{W}_{1a,*} - W_{1a,s})^\tau X_a)$.

Furthermore, the residual r_x satisfies the following inequality:

$$|r_x| \le \gamma_0 + \gamma_1 ||W_{1a,*}||_F + \gamma_2 ||W_{2a,*}||_F + \gamma_3 ||W_{1a,*}||_F ||W_{2a,*}||_F \tag{5}$$

where γ_0, γ_1, γ_2, γ_3 are properly chosen positive constants.

Proof. Due to the limitation on the number of the paper pages, the proof details are not presented. The proof can be done by using Taylor's series expansion, applying appropriate norms, and inserting the weights/biases of the off-line trained neural network. □

3.2 Neural Control Design

The control objective is to design a neural network based controller that enables the system output $y(t)$ to follow the desired trajectory $y_d(t)$ asymptotically. Suppose that $y_d(t)$ is sufficiently smooth such that $y_d^{n-1}(t)$ exists. Define $X_d = \left[y_d \; \dot{y}_d \cdots y_d^{n-1} \right]^\tau$. Define $\tilde{X} = X - X_d = \left[\tilde{X}_1 \; \tilde{X}_2 \cdots \tilde{X}_n \right]^\tau$. Define a tracking error metric $e_X(t) = \Lambda_x^\tau \tilde{X}$ with $\Lambda_x = \left[\lambda_0 \; \lambda_1 \cdots \lambda_{n-1} \right]^\tau$. λ_{n-1} is set to 1 for simplicity. The transfer function of this error system is given by $H(s) = \frac{\tilde{X}_1(s)}{E_X(s)} = \frac{1}{s^{n-1}+\lambda_{n-2}s^{n-2}+\cdots+\lambda_1 s+\lambda_0}$. The coefficients λ_i's $(i = 0, 1, \cdots, n-1)$ are properly chosen so that the denominator of the transfer is Hurwitz. As such, the asymptotic output tracking can be accomplished by maintaining $e_X(t) = 0$ which defines a time-varying hyperplane in R^n on which $\tilde{X}$ vanishes exponentially.

3.3 Adaptive Controller Design Based on the Modified Gradient Algorithm

In this section, instead of including a switching term in the control design, we take a high-gain control approach. Consider the controller in the following form:

$$u = u_{nn} + u_r \tag{6}$$

where u_{nn} is the contribution to the total control effort by the neural network based control, which is given by $u_{nn} = NN(X, \hat{\Theta}) = \hat{W_{2a}}^\tau \Sigma_a(\hat{W_{1a}}^\tau X_a)$, and u_r is the additional control effort for achieving robustness.

Let $e_r(t) = \Lambda_0^\tau \tilde{X} - X_d^{(n)}$ where $\Lambda_0 = \left[0 \; \lambda_0 \; \lambda_1 \cdots \lambda_{n-2} \right]^\tau$, and

$$u_{nn} = \frac{1}{b(X)}(-NN(X, \hat{\Theta}) - e_r(t) - c(X)^\tau \hat{P}) \tag{7}$$

In order to derive the parameter adaptation laws, consider a positive definite function as follows

$$V = \frac{1}{2}(e_X^2(t) + \tilde{P}^\tau \Gamma_P^{-1} \tilde{P} + tr\{\tilde{W}_{1a}^\tau \Gamma_{W_{1a}}^{-1} \tilde{W}_{1a}\} + \tilde{W}_{2a}^\tau \Gamma_{W_{2a}}^{-1} \tilde{W}_{2a} \tag{8}$$

where $\Gamma_P = \Gamma_P^\tau > 0$, $\Gamma_{W_{1a}} = \Gamma_{W_{1a}}^\tau > 0$, $\Gamma_{W_{2a}} = \Gamma_{W_{2a}}^\tau > 0$.

Differentiating V with respect to t and taking into account the equality $-e_X \hat{W}_{2a}^\tau \hat{\Sigma}_a' \tilde{W}_{1a}^\tau X_a = tr\{-e_X \hat{W}_{2a}^\tau \hat{\Sigma}_a' \tilde{W}_{1a}^\tau X_a\} = tr\{-\tilde{W}_{1a}^\tau X_a e_X \hat{W}_{2a}^\tau \hat{\Sigma}_a'\}$, we obtain

$$\begin{aligned} \dot{V} = & -\tilde{P}^\tau (c(X) e_X - \Gamma_P^{-1} \dot{\hat{P}}) - \tilde{W}_{2a}^\tau (e_X (\hat{\Sigma}_a - \hat{\Sigma}_a' (\hat{W}_{1a} - W_{1a,s})^\tau X_a) \\ & -\Gamma_{W_{2a}}^{-1} \dot{\hat{W}}_{2a}) - tr\{\tilde{W}_{1a}^\tau (X_a e_X \hat{W}_{2a}^\tau \hat{\Sigma}_a' - \Gamma_{W_{1a}}^{-1} \dot{\hat{W}}_{1a})\} \\ & -(r_x + r_{nn,a}) e_X + b(X) e_X u_r \end{aligned} \tag{9}$$

In order to ensure the boundedness of the weight estimates and bias estimates, modifying terms are introduced. There are a number of modification schemes available, to cite a few, σ-modification [6], switching σ-modification, e-modification [4], and parameter projections, that may be used to achieve robustness. In this paper, we choose to apply e-modification based scheme.

Let

$$\begin{aligned}\dot{\hat{P}} &= \Gamma_P c(X)e_X - \Gamma_P(1+|e_X|)\sigma_P\hat{P}\\ \dot{\hat{W}}_{2a} &= \Gamma_{W_{2a}} e_X(\hat{\Sigma}_a - \hat{\Sigma}_a^{'}(\hat{W}_{1a} - W_{1a,s})^\tau X_a)) - \Gamma_{W_{2a}}(1+|e_X|)\sigma_{W_{2a}}\hat{W}_{2a}\\ \dot{\hat{W}}_{1a} &= \Gamma_{W_{1a}} X_a e_X \hat{W}_{2a}^\tau \hat{\Sigma}_a^{'} - \Gamma_{W_{1a}}(1+|e_X|)\sigma_{W_{1a}}\hat{W}_{1a} \end{aligned} \tag{10}$$

Then

$$\begin{aligned}\dot{V} = &-(r_x + r_{nn,a})e_X + b(X)e_X u_r - \sigma_P\tilde{P}^\tau\hat{P}(1+|e_X|)\\ &-\sigma_{W_{2a}}\tilde{W}_{2a}^\tau\hat{W}_{2a}(1+|e_X|) - \sigma_{W_{1a}} tr\{\tilde{W}_{1a}^\tau\hat{W}_{1a}\}(1+|e_X|)\end{aligned} \tag{11}$$

Let

$$u_r = \frac{1}{b(X)}e_X(-k_r - \frac{(\gamma_s+\gamma_w)^2}{4\epsilon_u}) \tag{12}$$

where $k_r > 0$, $\epsilon_u > 0$ and γ_w is a properly defined constant.

After some algebraic operations, we obtain

$$\dot{V} \le -k_r e_X^2 - \frac{1}{2}\sigma_P||\tilde{P}||^2 - \frac{1}{2}\sigma_{W_{2a}}||\tilde{W}_{2a}||^2 - \frac{1}{2}\sigma_{W_{1a}}||\tilde{W}_{1a}||_F^2 + \epsilon_u + \gamma_w \tag{13}$$

In what follows, we now formally state the above results in a theorem.

Theorem 2. *With the control expressed in (6), (7) and (12), and the parameter updating laws (10),*

(a) All signals in the closed-loop are bounded.
(b) The filtered error signal e_X converges to a predesignated arbitrarily small neighborhood of zero.
(c) The integrated tracking performance satisfies

$$\frac{1}{t}\int_0^t \tilde{X}_1^2(t)dt \le \frac{1}{tk_r}(C_1V(0) + k_rC_0) + \frac{\epsilon_u+\gamma_w}{k_r} \tag{14}$$

where $t > 0$, C_0 and C_1 are positive constants, $k_r > 0$, $\epsilon_u > 0$ and $\gamma_w > 0$ are all design parameters, and $V(0) = \frac{1}{2}(e_X^2(0) + \tilde{P(0)}^\tau \Gamma_P^{-1}\tilde{P(0)} + tr\{\tilde{W(0)}_{1a}^\tau \Gamma_{W_{1a}}^{-1}\tilde{W(0)}_{1a}\} + \tilde{W(0)}_{2a}^\tau\Gamma_{W_{2a}}^{-1}\tilde{W(0)}_{2a}$ where $X(0)$, $e_X(0)$, $\tilde{W}_{1a}$, $\tilde{W}_{2a}$ and $\tilde{P}$ represent the system initial conditions.

(d) The tracking error is bounded from above.

$$||\tilde{X}_1(t)|| \le ||e(t)|| \le C_2||e(0)|| + \frac{C_2}{C_3}\sqrt{\frac{\epsilon_u+\gamma_w}{k_r}} \tag{15}$$

where C_2 and C_3 are positive constants.

Proof. Define a Lyapunov-like function (8). Apply the control expressed in (6), (7) and (12), and the parameter updating laws (10), we arrive at the inequality (13).

Due to the limit imposed on the number of pages of the paper, only a sketch of the proof is provided as follows: Applying the Lyapunov theory and properly choosing ϵ_u and k_r lead to the filtered error convergence, which in turn leads to the boundedness of all signals in the closed-loop. By proper integration and application of appropriate norms, the rest of the theorem can be proved. □

4 Conclusions

In this paper, an adaptive control design has been developed fora class of nonlinear uncertain systems, a typical example of which is a popular SMIB system with unknown load. It has been proposed that the offline neural network training and on-line neural network tuning should be integrated. This removes the popular assumption that the initial values of these neural network parameters fall within prescribed range. Furthermore, this approach makes best use of the offline neural network training and makes the design more sensible and effective. The transient performance is shown to assure a level of satisfactory performance. The stability and convergence of the system with the developed control laws and parameter updating laws has been proves.

References

1. Lewis, F., Yesidirek, A., Liu, K.: Neural Net Robot Controller with Guaranteed Tracking Performance. IEEE Trans. Neural Networks, **6** (1995) 703-715
2. Ge, S., Hang, C., Zhang, T.: Neural-based Adaptive Control Design for General Nonlinear Systems and Its Application to process control. Proc. American Control Conference, Philadelphia (1998) 73-77
3. Gong, J., Yao, B.: Neural Neural-based Adaptive Robust Control of a Class of Nonlinear Systems in Normal Form. Proc. American Control Conference, Chicago (2000) 1419-1423
4. Narendra, K., Annaswamy, A.: Stable Adaptive System. Prentice-Hall, New Jersey (1989)
5. Chen, D.: Nonlinear Neural Control with Power Systems Applications. Ph.D. Dissertation, Oregon State University (1998)
6. Ioannou, P., Sun, J.: Robust Adaptive Control. Prentice-Hall, Englewood Cliffs, NJ (1996)

Adaptive Inverse Control System Based on Least Squares Support Vector Machines

Xiaojing Liu, Jianqiang Yi, and Dongbin Zhao

Laboratory of Complex Systems and Intelligence Science, Institute of Automation Chinese Academy of Sciences, Beijing 100080, China

Abstract. Adaptive inverse control (AIC) uses three adaptive filters: plant model, controller and disturbance canceller. A kind of hybrid AIC system based on Least Squares Support Vector Machines (LS-SVMs) is proposed in this paper. It has a PID controller to compensate the control signal error. A kind of adaptive disturbance canceller based on LS-SVM is also proposed. It can optimally eliminate plant disturbance. Simulation example is presented to demonstrate that the proposed method works very well.

1 Introduction

Adaptive inverse control (AIC), as reformulated by Widrow and Walach[1], is an automatic control-system design method that uses signal-processing methods and learns over time how to control a specific plant, whether it is linear or nonlinear[2],[3]. For nonlinear systems, nonlinear adaptive filtering methods employing dynamic neural networks are used throughout. Unfortunately, neural networks, as well known, have the local minimum and overfitting problems, which result in poor generalization ability.

The support vector machine (SVM) is a new kind of learning machine proposed by Vapnik in 1995[4], which derives from statistical learning theory and VC-dimension theory[5]. Because the training of SVM is transferred into a quadratic programming (QP) optimization problem, there does not exist local minimum. The problem of curse of dimensionality, as well as overfitting problem, is also overcome. Based on standard SVM, a lot of variations of SVM have been put forward such as least squares support vector machine (LS-SVM). LS-SVM has been applied to function estimation and regression problems[6].

In this paper, a kind of hybrid AIC system based on LS-SVM is proposed and discussed. In this structure the output error is applied to calculate the input error by a PID controller. It can compensate the control signal error in a good way. Furthermore, a kind of adaptive disturbance canceler based on LS-SVM is also proposed. This method can optimally eliminate plant disturbance.

2 Least Squares Support Vector Machines (LS-SVMs)

LS-SVM function regression algorithm is as follow. Consider a training set containing N data point $\{x_k, y_k\}$, $k = 1, \cdots, N$, with input $x_k \in R^n$ and output $y_k \in R$. The following regression model is used.

[1] This work was partly supported by NSFC Projects (Grant No. 60334020, 60440420130, and 60475030), and MOST Projects (Grant No. 2003CB517106 and 2004DFB02100), China.

$$y(x) = w \cdot \varphi(x) + b\,. \tag{1}$$

where $\varphi(x)$ maps the input data into a higher dimensional feature space, w is the coefficient, b is the bias. In LS-SVM for function estimation, the object function is defined as follows:

$$\min J(w,e) = \frac{1}{2} w^{\mathrm{T}} w + \frac{1}{2} C \sum_{k=1}^{N} e_k^2 \tag{2}$$

subject to the equality constraints

$$y_k = w^{\mathrm{T}} \varphi(x_k) + b + e_k\,,\ k = 1,\cdots N, \tag{3}$$

where $e = [e_1;\cdots e_k;\cdots e_N]$ and C is regularization parameter. The corresponding Lagrangian is given by

$$L(w,b,e;\alpha) = J(w,e) - \sum_{k=1}^{N} \alpha_k \{w^{\mathrm{T}} \varphi(x_k) + b + e_k - y_k\} \tag{4}$$

where $\alpha = [\alpha_1;\cdots \alpha_k;\cdots \alpha_N]$, $\alpha_k (k = 1,\cdots N)$ are Lagrange multipliers. Considering the optimization conditions

$$\frac{\partial L}{\partial w} = 0\,,\ \frac{\partial L}{\partial b} = 0\,,\ \frac{\partial L}{\partial e_k} = 0\,,\ \frac{\partial L}{\partial \alpha_k} = 0\,, \tag{5}$$

optimization problem can be rewritten as

$$\begin{bmatrix} 0 & 1_v^{\mathrm{T}} \\ 1_v & \Omega + \frac{1}{c} I \end{bmatrix} \begin{bmatrix} b \\ \alpha \end{bmatrix} = \begin{bmatrix} 0 \\ y \end{bmatrix}, \tag{6}$$

where $y = [y_1;\cdots;y_N]$, $1_v = [1;\cdots;1]$, $\alpha = [\alpha_1;\cdots;\alpha_N]$. From the application of the Mercer condition, one obtains

$$\Omega_{kl} = K(x_k, x_l) = \varphi(x_k)^{\mathrm{T}} \varphi(x_l) \tag{7}$$

The resulting LS-SVM model for function estimation becomes

$$y(x) = \sum_{i=1}^{N} \alpha_i K(x, x_i) + b \tag{8}$$

where α_i and b are the solutions to equation (6). The kernel function $K(x, x_i) = \exp\{-|x - x_i|^2 / 2\sigma^2\}$ is Gaussian kernel in this paper.

3 Adaptive Inverse Control System

3.1 System Identification Using LS-SVM

The first step in performing adaptive inverse control is to make an adaptive model of the plant. Plant modeling based on LS-SVM is illustrated in fig.1. Consider the discrete single-input single-output nonlinear model

$$y(k+1) = f[y(k), y(k-1),\cdots y(k-n); u(k), u(k-1),\cdots, u(k-m)] \tag{9}$$

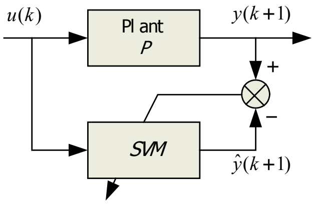


Fig. 1. Plant modeling based on LS-SVM

where $u(k)$ and $y(k)$ denote the input and output of system at the instant k. Given a series of input signals $u(k-m)$, $u(k-m+1)\cdots,u(k)$ and output signals $y(k-n)$, $y(k-n+1),\cdots,y(k)$, the corresponding output is $y(k+1)$. Then

$$X(i)=(y(i),y(i-1),\cdots,y(i-n),u(i),u(i-1),\cdots,u(i-m)),\ (i=1,2,\cdots,N) \tag{10}$$

$$y(i+1)=f(X(i)) \tag{11}$$

Constructing a data set $(X(i),y(i+1))$ and using LS-SVM regression algorithm, we get the estimator $\hat{y}(k+1)$ of the plant output $y(k+1)$.

$$\hat{y}(k+1)=\sum_{i=1}^{N}\alpha_i K(X(i),X(k))+b \tag{12}$$

3.2 Hybrid Adaptive Control Based on LS-SVM

Inverse modeling process to get the inverse model *SVMI*1 is similar to plant modeling process.

$$Y(i)=(y(i+1),y(i)\cdots y(i-n+1),u(i-1),u(i-2),\cdots u(i-m)) \tag{13}$$

$$u(i)=f(Y(i))\,. \tag{14}$$

So we can construct a data set $(Y(i),u(i))$. With LS-SVM regression algorithm, we get the estimator $\hat{u}(k)$ of plant input $u(k)$.

$$\hat{u}(k)=\sum_{i=1}^{N}\alpha_i^* K(Y(i),Y(k))+b^* \tag{15}$$

Through learning the samples, we can obtain α_i^*, b^* in (15).

In this paper, we propose a hybrid adaptive inverse control scheme shown in Fig.2. We use the error e_u between the output of two inverse models *SVMI*1 and *SVMIC* to optimize the overall dynamic response of the plant and its controller. The controller *SVMIC* adjusts its parameters according to *SVMI*1. That is to say, *SVMIC* is the copy of *SVMI*1.

When the trained SVM is used as a linear or nonlinear approximator, there usually exists approximate error. An error canceling method using a PID controller is presented in this paper. In this structure the PID controller is used to compensate the control signal error produced by the *SVMIC*. The parameters of the PID controller are chosen by experience in this paper.

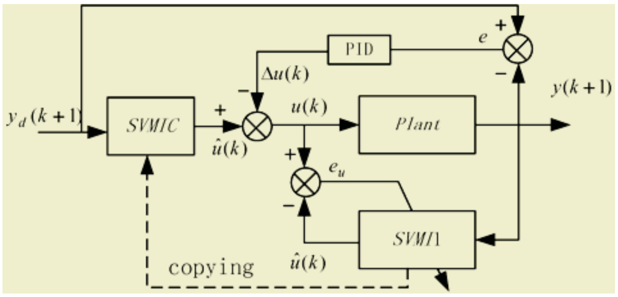


Fig. 2. Hybrid AIC system based on LS-SVM

3.3 Adaptive Inverse Disturbance Canceler Based on LS-SVM

First, we give another method of inverse modeling to get the inverse model *SVMI*2 as shown in fig.3. The idea is the same as that of plant modeling. The statistical character of synthetic noise $n(k)$ would be the same as that of the original plant disturbance.

$$Y(i) = (\hat{y}(i), \hat{y}(i-1)\cdots\hat{y}(i-n), n(i-1), n(i-2), \cdots n(i-m)) \tag{16}$$

$$n(i) = f(Y(i))\,. \tag{17}$$

So we can construct a data set $(Y(i), n(i))$. With LS-SVM regression algorithm, we get the estimator $\hat{n}(k)$ of disturbance $n(k)$.

$$\hat{n}(k) = \sum_{i=1}^{N} \alpha_i^{**} K(Y(i), Y(k)) + b^{**} \tag{18}$$

Through learning the samples, we can obtain α_i^{**}, b^{**} in (18).

Adaptive inverse disturbance system based on LS-SVM is diagrammed in fig.4. A copy of *SVM* is fed the same input as the plant *P*. The difference between the disturbed output of the plant and the disturbance-free output of *SVM* is a very close approximation to the plant output disturbance $n(k)$. The approximate $n(k)$ is then input to the filter *SVMI*2. The output of *SVMI*2 is subtracted from the plant input to effect cancellation of the plant disturbance.

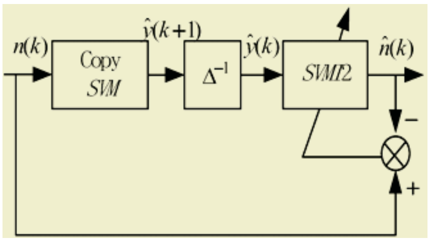


Fig. 3. Plant inverse modeling on LS-SVM

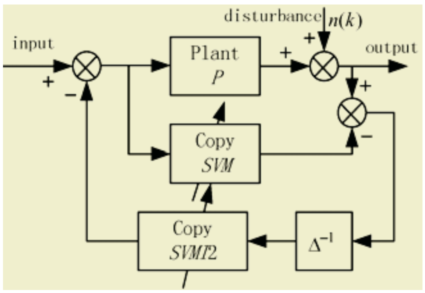


Fig. 4. Adaptive disturbance canceler based on LS-SVM

4 Simulation

The plant we wish to control is specified by the equations[7].

$$s(k) = \frac{s(k-1)}{1+s^2(k-1)} + \sin u(k-1), \quad y(k) = s(k) + n(k) \tag{19}$$

The input signal we selected is a sinusoidal signal whose amplitude is 1. The regularization parameter *C*=10 and kernel parameter σ=0.447. We select k_p=40, k_i=0 and k_d=0.28 for the PID control parameters. From fig.5 and fig.6, we can see that the hybrid AIC system with PID controller has a better control effect than that of the AIC system without PID controller.

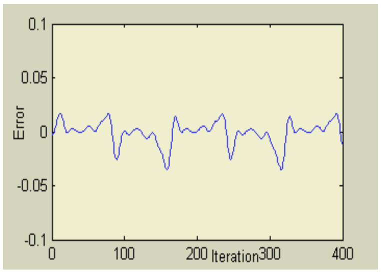


Fig. 5. Error between output and input of AIC system without PID controller

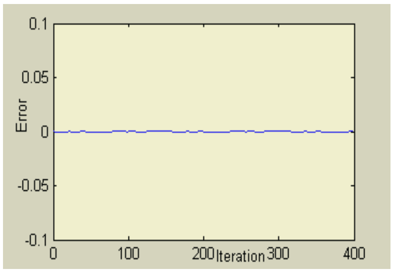


Fig. 6. Error between output and input of AIC system with PID controller

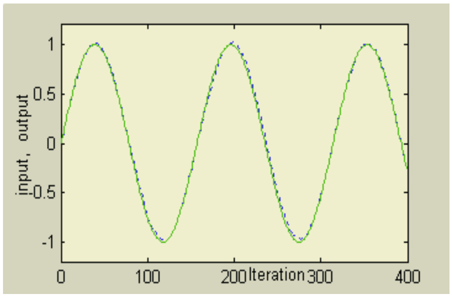


Fig. 7. AIC system without disturbance canceling

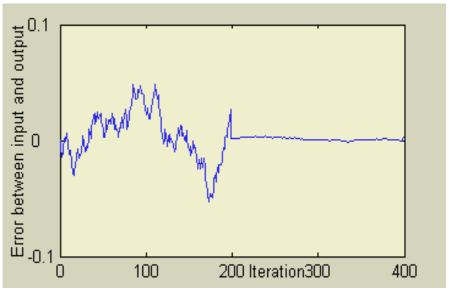


Fig. 8. AIC system with disturbance canceling (Disturbance canceling began at the 200th sample time)

For the Integrated AIC system, we select 400 sample points to construct data sets for plant model *SVM* and inverse model *SVMI*2. The regularization parameter *C*=10 and kernel parameter σ=0.447. The disturbance $n(k)$ is generated by filtering independently identically distributed Gaussian random numbers with zero mean and standard deviation 0.01 through a one-pole filter with the pole at z=0.99. The response of AIC system without disturbance canceling is shown in Fig.7. The result of a plant disturbance canceling is shown in Fig.8. At time sample 200, the canceling feedback is turned on. From Fig.8, we can see that the effect of plant disturbance cancellation is clearly evident.

5 Conclusions

A kind of AIC system based on LS-SVM is proposed. Three kinds of nonlinear filters based LS-SVM regression are used to construct a plant model, controller and disturbance canceller. Furthermore, a PID controller is used to compensate the control signal error produced by the controller. The simulation results demonstrate that the performances of the proposed control scheme are satisfactory.

References

1. Widrow, B., Walach, E.: Adaptive Inverse Control. Prentice Hall (1996)
2. Plett, G. L.: Adaptive Inverse Control of Plants with Disturbances. PhD Thesis, Stanford University, Stanford (1998)
3. Plett, G. L.: Adaptive Inverse Control of Linear and Nonlinear Systems Using Dynamic Neural Networks. IEEE Transactions on Neural Networks, **14** (2003) 360-369
4. Vapnik, V.: The Nature of Statistical Learning Theory. Springer-Verlag, New York (1995)
5. Vapnik, V.: Estimation of Dependences Based on Empirical Data. Springer-Verlag, Berlin (1982)
6. Suykens, J.A.K., Gestel, T.V., Brabanter, J.D., Moor, B.D., Vandewalle, J.: Least Squares Support Vector Machines. World Scientific, Singapore (2002)
7. Bilello, M.: Nonlinear Adaptive Inverse Control. PhD Thesis, Stanford University, Stanford, CA (1996)

H-Infinity Control for Switched Nonlinear Systems Based on RBF Neural Networks

Fei Long, Shumin Fei, and Shiyou Zheng

Department of Automatic Control, Southeast University, Nanjing 210096, China
flong1973@yahoo.com.cn

Abstract. Sub-controller and switching strategy based on RBF neural network are presented for a class of switched nonlinear systems in this paper. Sub-controller consists of equivalent controller and H-infinity controller. RBF neural network is used to approximate the unknown part of switched nonlinear systems, and the approximation errors of the RBF neural networks are introduced to the adaptive law in order to improve the performance of the whole systems. Sub-controller and switching strategy are designed to guarantee asymptotic stability of the output tracking error and to attenuate the effect of the external disturbance and approximation errors to a given level.

1 Introduction

During the last decades, applications of neural networks in system identification and control have been extensively studied. The study for nonlinear systems using universal function approximation has received much attention and many methods have been proposed (see [1], [2], [3], [4], [5], [6], [7], [8]). Typically, these methods use neural networks as approximation models for the unknown part of continuous systems. It has been shown that successful identification and control may be possible using neural networks for complex nonlinear dynamic systems whose mathematical models are not available from first principles. In [9] and [10], it was shown that for stable and efficient online control using the back propagation learning algorithm, the identification must be sufficiently accurate before the control action is initiated. In practical applications, it is desirable to have a systematic method of ensuring stability, robustness, and performance properties of the overall system. Recently, several good neural network control approaches have been proposed based on Lyapunov stability theory ([11], [12], [13], [14], [15]). One key advantage of these schemes is that the adaptive laws were derived based on Lyapunov synthesis and, therefore, guaranteed the stability of continuous systems without the requirement for offline training. However, most of these good results are restricted in continuous systems. Due to the difference between continuous systems and switched systems, stable controller designed in continuous system may become unstable in switched system via unstable switching strategy, thus we may run into troubles when we implement these networks controllers in switched system in which the data are typically available only at switching time instants. Therefore, the study for switched system based on neural network is necessary and significant. Unfortunately, all these good neural network controllers are designed for continuous/discrete nonlinear systems while there are a few attentions for switched control systems with disturbance presently. In [16], the state feedback control is studied for a class of switched nonlinear systems based on radial basis function (RBF) neural network. In this article, we investigate the adaptive H-infinity control problem for a class of switched nonlinear systems. Our results show that, provided that the upper bound of combined disturbance is small enough, the tracking error would be asymptotically stable and be attenuated the effect of the external disturbance and approximation errors to a given level.

2 Problem Statement

Consider the following switched nonlinear systems with exterior disturbance

$$\begin{cases} \dot{x} = \alpha_{s(t)}(x) + \beta_{s(t)}(x)u + d'_{s(t)} \\ y = \phi_{s(t)}(x) \end{cases} . \tag{1}$$

where $x \in \mathbb{R}^n$ is the state, $u \in \mathbb{R}$ is the control input, $y \in \mathbb{R}$ is the output and $s:[0,\infty) \to \{1,2,\cdots,N\} \overset{def}{=} \overline{\mathbb{N}}, N < \infty$, stands for the piecewise constant switching rule to be designed. $\alpha_{s(t)}(x)$, $\beta_{s(t)}(x)$ and $\phi_{s(t)}(x)$ are smoothly nonlinear function vectors or matrices with appropriated dimensions. $d'_{s(t)}$ denotes disturbance and uncertainty of system; may be it depends on state. Clearly, the system (1) can be generated by the following N subsystems via the action of switching strategy $s(t)$

$$\Sigma_k : \begin{cases} \dot{x} = \alpha_k(x) + \beta_k(x)u + d'_k \\ y = \phi_k(x) \end{cases}, k = 1,2,\cdots,N.$$

For any given $k \in \overline{\mathbb{N}}$, we suppose, for the system Σ_k, that

Assumption 1: The system Σ_k has relative degree n.

By calling up the nonlinear control systems theory [17] and the assumption 1, the system Σ_k can be rewritten by

$$y^{(n)} = f_k(x) + g_k(x)u + d_k . \tag{2}$$

where $(\cdot)^{(j)}$ denotes the *j-th* derivative of $(\cdot)$; $f_k(x) = L^n_{\alpha_k}\phi_k(x)$, $g_k(x) = L_{\beta_k}L^{n-1}_{\alpha_k}\phi_k(x)$,

$$d_k = \sum_{p=1}^{n} L^{n-p}_{\alpha_k + \beta_k u + d'_k} L_{d'_k} L^{p-1}_{\alpha_k}\phi_k(x) .$$

For the system (2), we assume that

Assumption 2: For $\forall x \in \mathbb{R}^n$ and $\forall k \in \overline{\mathbb{N}}$, $g_k(x)$ are bounded and $\sigma_{\min}(g_k(x)) \geq b > 0$. Where $\sigma_{\min}(g_k(x))$ denote the smallest singular value of the matrix $g_k(x)$.

Assumption 3: $f_k(x)$ is smooth, bounded function vector and d_k is bounded.

Setting $e = y - y_d$ (y_d is any given bounded reference output signal) denotes output tracking error, $\tilde{\theta} = \theta - \theta^*$ denotes the adaptive parameter error and ω_k stands for the sum of exterior disturbance d_k and the RBF neural network approximate error.

The objective in this paper is, for any given $\gamma > 0$, to design adaptive controller for every subsystem Σ_k and a switching strategy $s(t)$ for switched nonlinear system (1) such that the following two performances hold.

i). $\lim\limits_{t\to\infty} e(t) = 0$ when the combined disturbance ω_k has a upper bounder; and there exist a constant $M > 0$ such that $\|\theta\| \le M$.

ii). $\int_0^T e^T(t)Qe(t)dt \le \gamma^2 \int_0^T \varpi^T(t)\varpi(t)dt + S(e(0),\tilde{\theta}(0))$. where $Q^T = Q > 0$, S denotes appropriate positive-definite function, $\| \varpi \| = \max\{\| \varpi_1 \|, \| \varpi_2 \|, \cdots, \| \varpi_N \|\}$.

3 Main Results

Consider the switched nonlinear system (2), where $f_k(x)$ and $g_k(x)$ are unknown, smooth and bounded function; d_k denotes bounded exterior disturbance which is generated by some loads of system, external noise, etc. Now we utilize RBF network to approximate $f_k(x)$ and $g_k(x)$, i.e.

$$\hat{f}_k(x,\theta_f) = \theta_f^T \varphi_{fk}(x), \quad \hat{g}_k(x,\theta_g) = \theta_g^T \varphi_{gk}(x).$$

Where θ_f and θ_g are vector of adjustable weights; $\varphi_{fk}(x) = \begin{pmatrix}\varphi_{fk1}(x) & \cdots & \varphi_{fkp}(x)\end{pmatrix}$ and $\varphi_{gk}(x) = \begin{pmatrix}\varphi_{gk1}(x) & \cdots & \varphi_{gkq}(x)\end{pmatrix}$ denote vector of Guassian basis function; $\varphi_{fki}(x) = \exp(-\|x - c_{fki}\|^2 / \sigma_{fki}^2)$ and $\varphi_{gki}(x) = \exp(-\|x - c_{gki}\|^2 / \sigma_{gki}^2)$ stand for Guassian basis function, where (c_{fki}, σ_{fki}) and (c_{gki}, σ_{gki}) are center vector.

By calling up the neural network theory [18] and the literature [11], for $\forall x \in \Omega$ (Ω is a compact subset of $\mathbb{R}^n$) and $\varepsilon > 0$, there exist two Guassian basis function vectors $\varphi_{fk}(x)$ and $\varphi_{gk}(x)$, and two weight θ_f, θ_g such that $\| f_k(x) - \theta_f^T \varphi_{fk}(x) \| \le \varepsilon$, $\| g_k(x) - \theta_g^T \varphi_{gk}(x) \| \le \varepsilon$. Therefore $\theta_f^* = \arg\min\limits_{\theta_f \in \Omega_f} \left\{ \min\left\{ \sup\limits_{x\in\Omega} | f_k(x) - \theta_f^T \varphi_{fk}(x) |; k \in \overline{\mathbb{N}} \right\} \right\}$,

$$\theta_g^* = \arg\min_{\theta_g \in \Omega_g} \left\{ \min\left\{ \sup_{x\in\Omega} | g_k(x) - \theta_g^T \varphi_{gk}(x) |; k \in \overline{\mathbb{N}} \right\} \right\}, \quad \bar{d}_k = f_k(x) - \hat{f}_k(x,\theta_f^*) + (g_k(x) - \hat{g}_k(x,\theta_g^*))u$$

are well pose. Where Ω_f and Ω_g is known compact subset of $\mathbb{R}^p$ and $\mathbb{R}^q$.

Consider the controller $u = u_e + u_c$, where $u_e = \hat{g}_k^{-1}(x,\theta_g)\cdot(-\hat{f}_k(x,\theta_f) + \Lambda)$ denotes equivalent controller; $u_c = \hat{g}_k^{-1}(x,\theta_g)u_h$ (u_h is defined in the later) is compensatory controller that attenuate external disturbance and approximation errors of RBF neural networks with action of robust control; $\Lambda = y_d^{(n)} + \lambda_n(y_d^{(n-1)} - y^{(n-1)}) + \cdots + \lambda_1(y_d - y)$, where $(\lambda_1, \lambda_2, \cdots, \lambda_n)$ is Hurwitze vector, i.e., the following matrix is stable

$$A=\begin{pmatrix} 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 \\ -\lambda_1 & -\lambda_2 & \cdots & -\lambda_n \end{pmatrix}$$

The output tracking error dynamic equation of (2) is given by

$$\dot{\bar{e}} = A\bar{e} + B[u_h + \omega_k + (\hat{f}_k(x,\tilde{\theta}_f) + \hat{g}_k(x,\tilde{\theta}_g)u)]\,. \tag{3}$$

where $\bar{e}=[e,e^{(1)},\cdots,e^{(n-1)}]^T$; $B=[0,\cdots,0,1]^T\in\mathbb{R}^n$; $\tilde{\theta}_f=\theta_f^*-\theta_f$; $\tilde{\theta}_g=\theta_g^*-\theta_g$; $\omega_k=\bar{d}_k+d_k$ denotes the combined disturbance of systems.

According to the assumption 2 and the approximation property of RBF networks, the controller $u=u_e+u_c$ is bounded. Hence, the complex disturbance ω_k has upper bounder. Without loss of generality, we denote its upper bounder by $\bar{\omega}_k$. For the following switched system, i. e. the tracking error dynamic system of system (1),

$$\dot{\bar{e}} = A\bar{e} + B[u_h + \omega_{s(t)} + (\hat{f}_{s(t)}(x,\tilde{\theta}_f) + \hat{g}_{s(t)}(x,\tilde{\theta}_g)u)]\,. \tag{4}$$

we have the following result.

Theorem: Consider the system (4) and suppose that, for any given $\gamma>0$ and the matrix $Q^T=Q>0$, there exist a positive number $l\le 2\gamma^2$ and the matrix $P^T=P>0$ such that

$$PA+A^TP+Q+(\gamma^{-2}-2l^{-1})PBB^TP\le 0\,,\quad \bigcup_{k=1}^{N}\left\{\bar{e};\|\bar{\omega}_k\|^2<C_1C_2/\gamma^2\right\}\subseteq E\overset{def}{=}\left\{\bar{e}\,|\,\bar{e}^TP\bar{e}\le C_2\right\}.$$

Then there exist RBF neural network adaptive controller and the switching strategy such that the closed –loop system satisfies the performances i) and ii). Where the switching strategy and RBF neural network adaptive controller is given by

$$s(t)=\arg\min_k\left\{\|\bar{\omega}_k\|^2<C_1C_2/\gamma^2\right\},\quad u_h=-(1/l)B^TP\bar{e}\,,$$

$$\begin{cases}\dot{\theta}_g=\rho\bar{e}^TPB\varphi_{gk}(x)u,\theta_g\in\Omega_g\\ \rho^{-1}\tilde{\theta}_g^T(\dot{\theta}_g-\rho\bar{e}^TPB\varphi_{gk}(x)u)\ge 0,\theta_g\in\Omega_{\delta g}\setminus\Omega_g\end{cases},\quad \begin{cases}\dot{\theta}_f=\mu\bar{e}^TPB\varphi_{fk}(x),\theta_f\in\Omega_f\\ \mu^{-1}\tilde{\theta}_f^T(\dot{\theta}_f-\mu\bar{e}^TPB\varphi_{fk}(x))\ge 0,\theta_f\in\Omega_{\delta f}\setminus\Omega_f\end{cases}.$$

Where $\Omega_f\subset\Omega_{\delta f}$, $\Omega_g\subset\Omega_{\delta g}$ ($\Omega_{\delta f}\subset\mathbb{R}^p$ and $\Omega_{\delta g}\subset\mathbb{R}^q$ are known compact set), μ and ρ denote the learning rate, $C_1=\lambda_{\min}(Q)/\lambda_{\max}(P)$; $C_2>\max_{\bar{e}\in E_0}\bar{e}^TP\bar{e}$, the set E_0 is any given known compact subset of $\mathbb{R}^n$ and contains $\bar{e}(0)$.

4 An Example

Consider system (1) with $N=2$, and

$$\alpha_2(x)=\begin{pmatrix}x_1^2+x_2 & x_3 & -x_2\end{pmatrix}^T,\ \alpha_1(x)=\begin{pmatrix}x_1^2+x_2 & x_3 & x_2+x_3\end{pmatrix}^T,\ \beta_2(x)=\begin{pmatrix}0 & 0 & e^{x_2}\end{pmatrix}^T,$$

$$d_1'=\begin{pmatrix}0 & 0 & d_1\end{pmatrix}^T,d_2'=\begin{pmatrix}0 & 0 & d_2\end{pmatrix}^T,\ \phi_1(x)=\phi_2(x)=x_1,\ \beta_1(x)=\begin{pmatrix}0 & 0 & e^{x_3}\end{pmatrix}^T.$$

By calling up [17], both the systems Σ_1 and Σ_2 have relative degree 3; and

$$f_1(x) = 6x_1^4 + 8x_1^2x_2 + 2x_2^2 + 2x_1x_3 + x_2 + x_3, \; g_1(x) = e^{x_3},$$
$$f_2(x) = 6x_1^4 + 8x_1^2x_2 + 2x_2^2 + 2x_1x_3 - x_2, \; g_2(x) = e^{x_2}.$$

Setting the number of hidden layer $p = q = 20$, the width $\sigma_{fki} = \sigma_{gki} = 2$, $(\lambda_1, \lambda_2, \lambda_3) = (1,2,4)$, $Q = I_{3\times3}$. The center vector c_{fki} and c_{gki} are generated randomly in input-output domain. The learning rate is given by $\mu = 0.4, \rho = 0.01$. The disturbance attenuation level $\gamma = 0.25$ and $l = 2\gamma^2$. The plant output is required to track a reference signal that is the output of a low-pass filter with transfer function, driven by a unity amplitude square wave input with frequency 0.4 Hz and a time average of 0.5. The reference signal and its derivatives are shown in Figure 2. Figure 3 shows the corresponding switching signal with state feedback switching. The tracking error is showed in Figure 4. The simulation illustrates the fact: by using a robust component, it is possible to obtain H-infinity performance even with RBF neural networks that give large approximate errors.

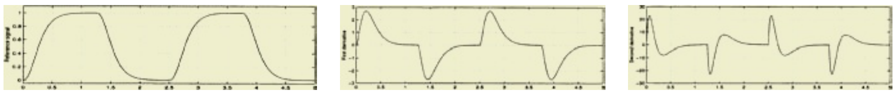
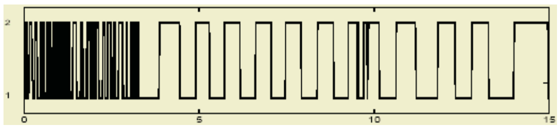


Fig. 1. The reference signal and its derivative



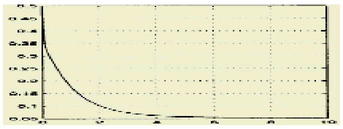


Fig. 2. The switching signal

Fig. 3. The tracking error

5 Conclusions

Using RBF neural networks, adaptive H-infinity control scheme and switching strategy have been studied for a class of switched nonlinear systems with external perturbation. Based on the results of [7] and [8], RBF neural network is used to adaptive compensate for the external perturbation of switched nonlinear system and the approximation errors of RBF neural networks are introduced to the adaptive law in order to improve the quality of the whole system. Each adaptive sub-controller and switching strategy are designed to guarantee asymptotic stability of the output tracking error and to attenuate the effect of the exterior perturbation and approximation errors of the RBF neural networks to a given level.

References

1. Liu, C., Chen, F.: Adaptive Control of Nonlinear Continuous Systems Using Neural Network-General Relative Degree and MIMO Case. Int. J. Control, **58** (1993) 317-335
2. Narendrk, K. S., Mukhopadhyay, S.: Adaptive Control of Nonlinear Multivariable System Using Neural Network. Neural Network, **7** (1994) 737-752
3. Liu, G. P., et al.: Variable Neural Networks for Adaptive Control of Nonlinear Systems. IEEE Trans Systems, man, Cybermetics-Part C, **29** (1999) 34-43
4. Patino, H. D., Liu, D.: Neural Network-based Model Reference Adaptive Control Systems. IEEE Trans Systems, man, Cybermetics-Part B, **30** (2001) 198-204
5. Sanner, R., Slotine, J. J.: Gaussian Networks for Direct Adaptive Control. IEEE Trans on Neural Networks, **3** (1992) 837-864
6. Seshagiri, S., Hassan, K. K.: Output Feedback Control of Nonlinear Systems Using RBF Neural Networks. IEEE Trans On Neural Network, **11** (2000) 69--79
7. Chen, M., et al.: Adaptive H_∞ Control For A Class Of Uncertain Nonlinear Systems Based On RBF Neural Networks. Control Theory & Applications, **20** (2003) 27-32
8. Ding, G., et al.: H_∞ Control Of Uncertain Nonlinear Systems Based On Neural Network, Control and Decision, **12** (1997) 571-575
9. Levin, A.U., Narendra, K.S.: Control of Nonlinear Dynamical Systems Using Neural Networks-Part II: Observability, Identification, and Control. IEEE Trans On Neural Networks, **7** (1996) 30-42
10. Narendra, K.S., Parthasarathy, K.: Identification and Control of Dynamic Systems Using Neural Networks. IEEE Trans On Neural Networks, **1** (1990) 4-27
11. Ge, S.S., et al.: A Direct Method for Robust Adaptive Nonlinear Control with Guaranteed Transient Performance. System Control Lett., **37** (1999) 275-284
12. Lewis, F.L., et al.: Multilayer Neural-net Robot Controller with Guaranteed Tracking Performance. IEEE Trans On Neural Networks, **7** (1996) 388-398
13. Polycarpou, M.M.: Stable Adaptive Neural Control Scheme for Nonlinear Systems. IEEE Trans. Automat. Contr., **41** (1996) 447-450
14. Ge, S.S., et al.: Adaptive Neural Network Control of Robotic Manipulators. World Scientific, Singapore (1998)
15. Ge, S.S., et al.: Stable Adaptive Neural Network Control. Norwell, MA: Kluwer (2001)
16. Long, F., Fei, S.: State Feedback Control for a Class of Switched Nonlinear Systems Based on RBF Neural Networks. Proc. 23rd Chinese Control Congress. Wuxi, China (2004) 1611-1614
17. Isidori, A.: Nonlinear Control Systems (2nd edition). Springer-Verlag, New York (1995)
18. Haykin, S.: Neural Networks: A Comprehensive Foundation (2nd edition). Prentice Hall, New York (1994)

Neural Networks Robust Adaptive Control for a Class of MIMO Uncertain Nonlinear Systems

Tingliang Hu, Jihong Zhu, Chunhua Hu, and Zengqi Sun

State Key Lab of Intelligent Technology and Systems,
Department of Computer Science and Technology,
Tsinghua University,
Beijing 100084, China
htl02@mails.tsinghua.edu.cn

Abstract. This paper presents a robust adaptive control scheme for a class of multi-input multi-output (MIMO) uncertain nonlinear systems. Multiple multi-layer neural networks are used to compensate for the model uncertainty. The on-line updating rules of the neural networks parameters are obtained by Lyapunov stability theory. All signals in the closed-loop system are bounded. The output tracking error converges to a small neighborhood of zero, while the stability of the closed-loop system is guaranteed. Finally the effectiveness of the control scheme is verified by a simulation of two-link manipulator.

1 Introduction

During the last decade, significant progress has been made in the design of neural network controllers which are able to guarantee the closed-loop stability property for systems with complex and possibly unknown nonlinearities. Due to the fact of universal approximators, we may straightforwardly substitute unknown system nonlinearities by a neural network which is of known structure but contains a number of unknown parameters (synaptic weights), plus a modeling error term, thus, transforming the original problem into a nonlinear robust adaptive control one. Therefore it is not necessary to spend much effort on system modeling which might be very difficult in some cases. Since the early 1990s, Lyapunov's stability theory has been utilized in the neural networks control for the attempt to introduce guaranteed closed-loop system performance [1-6]. However, a priori knowledge of the controlled plant can not be incorporate into the neural networks controller except for the off-line training, because the synaptic weights of the neural networks controller are not of obvious physical meaning.

In this paper a new neural networks robust adaptive control scheme for MIMO systems with uncertainty is proposed. Multiple multi-layer neural networks are used in the controller to compensate the model uncertainty. Using Lyapunov stability theory, the on-line updating rules of the neural networks parameters are obtained, while the stability and the boundedness of all the signals in the closed-loop system are guaranteed and the output tracking error converges to a small neighborhood of zero.

The paper is organized as follows: In Section II, the problem statement is presented. In Section III, the structure of the controller is firstly given and the error system dynamics is established. Then the on-line updating rules of the neural networks parameters are obtained. In Section IV, we present a simulation of two-link manipulator to verify the effectiveness of the proposed neural controller. The conclusions are drawn in Section V.

2 Problem Statement

Consider the following MIMO system that is described by differential equation:

$$\boldsymbol{y}^{(n)} = \boldsymbol{f}(\boldsymbol{x})+\Delta\boldsymbol{f}(\boldsymbol{x})+\left(\boldsymbol{g}(\boldsymbol{x})+\Delta\boldsymbol{g}(\boldsymbol{x})\right)\boldsymbol{u}+\boldsymbol{d}\cdot \tag{1}$$

where $\boldsymbol{y}=[y_1,\cdots,y_{\rm m}]^T\in \mathrm{R}^m$ is output vector, $\boldsymbol{y}^{(n)}=[y_1^{(n_1)},\cdots,y_m^{(n_m)}]^T\in \mathrm{R}^m$, $y_i^{(n_i)}=\mathrm{d}^{n_i}y_i/\mathrm{d}t^{n_i}$; $\boldsymbol{x}=[y_1,\cdots,y_1^{(n_1-1)},\cdots\cdots,y_m,\cdots,y_m^{(n_m-1)}]\in \mathrm{R}^n$ with $n=n_1+\cdots+n_m$ is the state vector; $\boldsymbol{f}(\boldsymbol{x})\in \mathrm{R}^m$ and $\boldsymbol{g}(\boldsymbol{x})\in \mathrm{R}^{m\times m}$ are known functions and $\Delta\boldsymbol{f}(\boldsymbol{x})\in \mathrm{R}^m$ and $\Delta\boldsymbol{g}(\boldsymbol{x})\in \mathrm{R}^{m\times m}$ are unknown continuous functions, i.e. the uncertainties. $\boldsymbol{u}\in \mathrm{R}^m$ is the input vector; $\boldsymbol{d}=[d_1,\cdots,d_m]$ is slow changing disturbance vector with $\|\boldsymbol{d}\|<d_0,\dot{\boldsymbol{d}}\approx 0$.

Define the reference signal vector $\boldsymbol{Y}_d$ and the tracking error vector $\boldsymbol{E}$ as

$$\boldsymbol{Y}_d=[y_{1\mathrm{d}},\cdots,y_{1d}^{(n_1-1)},\cdots\cdots,y_{md},\cdots,y_{md}^{(n_m-1)}]\in \mathrm{R}^n. \tag{2}$$

$$\boldsymbol{E}=[e_1,\cdots,e_1^{(n_1-1)},\cdots\cdots,e_m,\cdots,e_m^{(n_m-1)}]\in \mathrm{R}^n\cdot \tag{3}$$

where $e_i=y_i-y_{id}\in R,\ i=1,\cdots,m$. Then, the control objective is to make the system outputs follows the given bounded reference signals under the stability constraint and all signals involved in the system must be bounded. In this paper, $\|\cdot\|$ denotes Euclidean norm of vector, $\|\cdot\|_F$ stands for the Frobenius norm of matrix, L_∞ is the bounded continuous function space. For the system (1) the following assumptions are given:
a) System (1) is controllable, i.e. $\boldsymbol{g}(\boldsymbol{x})\subset \mathrm{R}^{m\times m}$ is invertible for $\boldsymbol{x}\in\Omega$ where Ω is a compact set; b) The reference signal vector $\boldsymbol{Y}_d$ is known bounded functions of time; c) The states of the system are measurable.

3 Adaptive Controller

3.1 Controller Structure and the Error Dynamics

The structure of the controller is as follows:

$$\boldsymbol{u}=\boldsymbol{G}(\boldsymbol{x})\left(\boldsymbol{v}+\boldsymbol{F}(\boldsymbol{x})+\boldsymbol{\delta}\right)\cdot \tag{4}$$

where $\boldsymbol{G}(\boldsymbol{x})$ and $\boldsymbol{F}(\boldsymbol{x})$ are function matrix and vector of the state vector $\boldsymbol{x}$; $\boldsymbol{\delta}$ is used to cancel the approximated errors between the nonlinear functions and the neural network approximator with limited neutrons, $\boldsymbol{v}$ is the new control input. Let

$$\begin{aligned}&\boldsymbol{v}=\boldsymbol{y}_d^{(n)}-\boldsymbol{\lambda}^T\boldsymbol{E},\quad \boldsymbol{\lambda}^T=\mathrm{diag}(\lambda_1^T,\lambda_2^T,\cdots,\lambda_m^T)\in R^{m\times n}\\&\lambda_i=[\lambda_{i0},\lambda_{i1},\cdots,\lambda_{i,(n_i-1)}]^T\in R^{n_i},\ i=1,\cdots,m\end{aligned}\quad\cdot \tag{5}$$

choosing $\lambda_i=[\lambda_{i0},\lambda_{i1},\cdots,\lambda_{i,(n_i-1)}]^T\in R^{n_i}$, such that (6) is Hurwitz

$$s_i^{(n_i)}+\lambda_{i,(n_i-1)}s_i^{(n_i-1)}+\cdots+\lambda_{i0}s_i=0,\ i=1,\cdots,m. \tag{6}$$

Applying the controller (4) into the system (1), we obtain

$$\boldsymbol{y}^{(n)} = \boldsymbol{f}(\boldsymbol{x})+\Delta\boldsymbol{f}(\boldsymbol{x})+\left(\boldsymbol{g}(\boldsymbol{x})+\Delta\boldsymbol{g}(\boldsymbol{x})\right)\boldsymbol{G}(\boldsymbol{x})\left(\boldsymbol{v}+\boldsymbol{F}(\boldsymbol{x})+\boldsymbol{\delta}\right)+\boldsymbol{d}\,. \tag{7}$$

To guarantee the asymptotic convergence of the output error $\boldsymbol{E}$, the following relationships should be satisfied

$$\begin{array}{l}\left(\boldsymbol{g}(\boldsymbol{x})+\Delta\boldsymbol{g}(\boldsymbol{x})\right)\boldsymbol{G}^{*}(\boldsymbol{x})=1; \quad \boldsymbol{d}=-\left(\boldsymbol{g}(\boldsymbol{x})+\Delta\boldsymbol{g}(\boldsymbol{x})\right)\boldsymbol{G}^{*}(\boldsymbol{x})\boldsymbol{\delta}^{*} \\ \boldsymbol{f}(\boldsymbol{x})+\Delta\boldsymbol{f}(\boldsymbol{x})+\left(\boldsymbol{g}(\boldsymbol{x})+\Delta\boldsymbol{g}(\boldsymbol{x})\right)\boldsymbol{G}^{*}(\boldsymbol{x})\boldsymbol{F}^{*}(\boldsymbol{x})=0, \qquad \forall\boldsymbol{x}\in\Omega\end{array}. \tag{8}$$

where $\boldsymbol{G}^{*}(\boldsymbol{x}), \boldsymbol{F}^{*}(\boldsymbol{x}), \boldsymbol{\delta}^{*}$ denote the ideal unknown nonlinear function matrix and vectors. Define following error matrices and error vectors:

$$\begin{array}{l}\tilde{\boldsymbol{G}}(\boldsymbol{x})=\boldsymbol{G}^{*}(\boldsymbol{x})-\boldsymbol{G}(\boldsymbol{x}),\ \tilde{\boldsymbol{\psi}}(\boldsymbol{x})=\boldsymbol{G}^{*-1}(\boldsymbol{x})-\boldsymbol{G}^{-1}(\boldsymbol{x}) \\ \tilde{\boldsymbol{F}}(\boldsymbol{x})=\boldsymbol{F}^{*}(\boldsymbol{x})-\boldsymbol{F}(\boldsymbol{x}),\tilde{\boldsymbol{\delta}}=\boldsymbol{\delta}^{*}-\boldsymbol{\delta}\end{array}. \tag{9}$$

Substituting (8), (9), (5) into (7), we obtain the error system dynamic and transform it to the state space form

$$\dot{\boldsymbol{E}}=\boldsymbol{A}\boldsymbol{E}+\boldsymbol{B}\boldsymbol{\Delta}\,. \tag{10}$$

where

$$\boldsymbol{A}=\begin{bmatrix}\boldsymbol{A}_1 & 0 & \cdots & 0\\ 0 & \boldsymbol{A}_2 & \cdots & 0\\ \cdots & \cdots & \cdots & \cdots\\ 0 & 0 & 0 & \boldsymbol{A}_m\end{bmatrix} \qquad \boldsymbol{B}=\begin{bmatrix}\boldsymbol{B}_1\\ \boldsymbol{B}_2\\ \cdots\\ \boldsymbol{B}_m\end{bmatrix}$$

$$\boldsymbol{A}_i=\begin{bmatrix}0 & 1 & 0 & \cdots & 0\\ 0 & 0 & 1 & \cdots & 0\\ \vdots & \vdots & \vdots & \vdots & \vdots\\ 0 & 0 & 0 & \cdots & 1\\ -\lambda_{i0} & -\lambda_{i1} & -\lambda_{i2} & \cdots & -\lambda_{i,(n_i-1)}\end{bmatrix} \qquad \boldsymbol{B}_i=-\begin{bmatrix}0 & \cdots & 0 & \cdots & 0\\ 0 & \cdots & 0 & \cdots & 0\\ \cdots & \cdots & \cdots & \cdots & \cdots\\ 0 & \cdots & 1_i & \cdots & 0\end{bmatrix}_{n_i\times m}$$

$$i=1,\cdots,m,\ \Delta=-\tilde{\psi}(\boldsymbol{x})u+\tilde{F}(\boldsymbol{x})+\tilde{\delta}$$

Let

$$\boldsymbol{G}(\boldsymbol{x})=\left(\boldsymbol{g}(\boldsymbol{x})+\Delta\boldsymbol{g}(\boldsymbol{x})\right)^{-1},\ \ \boldsymbol{F}(\boldsymbol{x})=\boldsymbol{f}(\boldsymbol{x})+\Delta\boldsymbol{f}(\boldsymbol{x})\,. \tag{11}$$

According to (9), we have

$$\tilde{\boldsymbol{F}}(\boldsymbol{x})=\Delta\boldsymbol{f}^{*}(\boldsymbol{x})-\Delta\boldsymbol{f}(\boldsymbol{x}),\ \tilde{\boldsymbol{\psi}}(\boldsymbol{x})=\boldsymbol{G}(\boldsymbol{x})^{*-1}-\boldsymbol{G}(\boldsymbol{x})^{-1}=\Delta\boldsymbol{g}^{*}(\boldsymbol{x})-\Delta\boldsymbol{g}(\boldsymbol{x})\,. \tag{12}$$

where $\Delta\boldsymbol{f}^{*}(\boldsymbol{x})$ and $\Delta\boldsymbol{g}^{*}(\boldsymbol{x})$ represent the ideal unknown function matrix and vector.

Although there are $m(m+1)$ uncertain nonlinear functions in the controller, $(m+1)$ single-hidden-layer neural networks, each one has m outputs, can be used to approximate $\Delta\boldsymbol{g}^{*}(\boldsymbol{x})$ and $\Delta\boldsymbol{f}^{*}(\boldsymbol{x})$. The structure of the neural network is shown in Fig.1. Assuming that the outputs of the $m+1$ neural networks with ideal weights are equal to $\Delta\boldsymbol{g}^{*}(\boldsymbol{x})$ and $\Delta\boldsymbol{f}^{*}(\boldsymbol{x})$, the small bounded approximated error can be merged into $\boldsymbol{\delta}$. Then

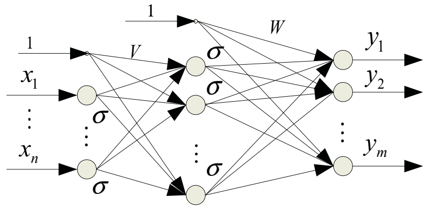


Fig. 1. Single hidden layer neural networks

$$\Delta \boldsymbol{g}^*(\boldsymbol{x}) = [\boldsymbol{W}_1^{*T}\boldsymbol{\sigma}(\boldsymbol{V}_1^{*T}\bar{\boldsymbol{x}}),\cdots,\boldsymbol{W}_m^{*T}\boldsymbol{\sigma}(\boldsymbol{V}_m^{*T}\bar{\boldsymbol{x}})],\ \ \Delta \boldsymbol{f}^*(\boldsymbol{x}) = \boldsymbol{W}^{*T}\boldsymbol{\sigma}(\boldsymbol{V}^{*T}\bar{\boldsymbol{x}})\cdot \tag{13}$$

where $\boldsymbol{\sigma}(z)=\left[1,\sigma(z_1),\sigma(z_2),\cdots,,\sigma(z_N)\right]$ denotes the output vector of the hidden layer, $\bar{\boldsymbol{x}}=\left[1,\sigma(x_1),\sigma(x_2),\cdots,\sigma(x_n)\right]$ is the output vector of the input layer, $\sigma(\cdot)$ is the Sigmoid function. $\boldsymbol{W}^*,\boldsymbol{V}^*,\ \boldsymbol{W}_1^*,\ \boldsymbol{V}_1^*,\cdots,\boldsymbol{W}_m^*,\ \boldsymbol{V}_m^*$ are the ideal bounded weights of the neural networks and $M_W,M_V,M_{W_1},M_{V_1},\cdots,M_{W_m},M_{V_m}$ are their bounds respectively.

The output of the neural networks can be represented as:

$$\Delta \boldsymbol{g}(\boldsymbol{x}) = [\boldsymbol{W}_1^{T}\boldsymbol{\sigma}(\boldsymbol{V}_1^{T}\bar{\boldsymbol{x}}),\cdots,\boldsymbol{W}_m^{T}\boldsymbol{\sigma}(\boldsymbol{V}_m^{T}\bar{\boldsymbol{x}})],\qquad \Delta \boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{W}^{T}\boldsymbol{\sigma}(\boldsymbol{V}^{T}\bar{\boldsymbol{x}}). \tag{14}$$

where $\boldsymbol{W},\boldsymbol{V},\ \boldsymbol{W}_1,\ \boldsymbol{V}_1,\cdots,\boldsymbol{W}_m,\ \boldsymbol{V}_m$ denote the estimated value of the ideal weight matrices $\boldsymbol{W}^*,\boldsymbol{V}^*,\ \boldsymbol{W}_1^*,\ \boldsymbol{V}_1^*,\cdots,\boldsymbol{W}_m^*,\ \boldsymbol{V}_m^*$ respectively. Define the following error matrices

$$\tilde{\boldsymbol{W}} = \boldsymbol{W}^* - \boldsymbol{W},\tilde{\boldsymbol{V}} = \boldsymbol{V}^* - \boldsymbol{V},\tilde{\boldsymbol{W}}_i = \boldsymbol{W}_i^* - \boldsymbol{W}_i,\tilde{\boldsymbol{V}}_i = \boldsymbol{V}_i^* - \boldsymbol{V}_i, i=1,\cdots,m. \tag{15}$$

Using the Taylor series expansion of $\boldsymbol{\sigma}(\boldsymbol{V}^*\boldsymbol{x})$ about $\boldsymbol{\sigma}(\boldsymbol{V}\boldsymbol{x})$, the differences between the true value and the estimated output of the neural networks can be represent as:

$$\begin{aligned}&\boldsymbol{W}^{*T}\boldsymbol{\sigma}(\boldsymbol{V}^{*T}\bar{\boldsymbol{x}}) - \boldsymbol{W}^{T}\boldsymbol{\sigma}(\boldsymbol{V}^{T}\bar{\boldsymbol{x}}) = \tilde{\boldsymbol{W}}^{T}\boldsymbol{\sigma}(\boldsymbol{V}^{T}\bar{\boldsymbol{x}}) + \boldsymbol{W}^{*T}\left(\boldsymbol{\sigma}(\boldsymbol{V}^{*T}\bar{\boldsymbol{x}}) - \boldsymbol{\sigma}(\boldsymbol{V}^{T}\bar{\boldsymbol{x}})\right).\\ &= \tilde{\boldsymbol{W}}^{T}\left(\boldsymbol{\sigma}(\boldsymbol{V}^{T}\bar{\boldsymbol{x}}) - \dot{\boldsymbol{\sigma}}\boldsymbol{V}^{T}\bar{\boldsymbol{x}}\right) + \boldsymbol{W}^{T}\dot{\boldsymbol{\sigma}}\,\tilde{\boldsymbol{V}}^{T}\bar{\boldsymbol{x}} + \tilde{\boldsymbol{W}}^{T}\dot{\boldsymbol{\sigma}}\,\boldsymbol{V}^{*T}\bar{\boldsymbol{x}} + \boldsymbol{W}^{*T}O(\tilde{\boldsymbol{V}}^{T}\bar{\boldsymbol{x}})^2\end{aligned} \tag{16}$$

where $\boldsymbol{\sigma}(\boldsymbol{V}^{T}\bar{\boldsymbol{x}}) - \boldsymbol{\sigma}(\boldsymbol{V}^{*T}\bar{\boldsymbol{x}}) = \dot{\boldsymbol{\sigma}}\,\tilde{\boldsymbol{V}}^{T}\bar{\boldsymbol{x}} + O(\tilde{\boldsymbol{V}}^{T}\bar{\boldsymbol{x}})^2$, $\dot{\boldsymbol{\sigma}} = \mathrm{d}\boldsymbol{\sigma}(z)/\mathrm{d}z\big|_{z=\boldsymbol{V}^T\bar{\boldsymbol{x}}}$. Since $\boldsymbol{\sigma},\dot{\boldsymbol{\sigma}}$ and $\boldsymbol{W}^*,\boldsymbol{V}^*,\ \boldsymbol{W}_1^*,\ \boldsymbol{V}_1^*,\cdots,\boldsymbol{W}_m^*,\ \boldsymbol{V}_m^*$ are bounded, we can obtain the following inequalities:

$$\left\|\tilde{\boldsymbol{W}}^{T}\dot{\boldsymbol{\sigma}}\,\boldsymbol{V}^{*T}\bar{\boldsymbol{x}} + \boldsymbol{W}^{*T}\boldsymbol{O}(\tilde{\boldsymbol{V}}^{T}\bar{\boldsymbol{x}})^2\right\| \le c_1 + c_2\left\|\tilde{\boldsymbol{W}}\right\|_F + c_3\left\|\tilde{\boldsymbol{V}}\right\|_F\cdot \tag{17}$$

$$\left\|(\tilde{\boldsymbol{W}}_i^{T}\dot{\boldsymbol{\sigma}}\,\boldsymbol{V}_i^{*T}\bar{\boldsymbol{x}} + \boldsymbol{W}_i^{*T}\boldsymbol{O}(\tilde{\boldsymbol{V}}_i^{T}\bar{\boldsymbol{x}})^2)u_i\right\| \le \left(d_{1i} + d_{2i}\left\|\tilde{\boldsymbol{W}}_i\right\|_F + d_{3i}\left\|\tilde{\boldsymbol{V}}_i\right\|_F\right)\left\|u_i\right\|,\ \ i=1,\cdots,m\cdot \tag{18}$$

where $c_1,c_2,c_3,d_{1i},d_{2i},d_{3i},\ i=1,\cdots,m$ are positive constants.

3.2 Stability Analysis and Adaptive Rules

Based on the error dynamic equation (10) and Taylor series expansion of ideal output about the estimated value of the neural networks, the system stability is demonstrated and the adaptive rules of the neural networks parameters are obtained. Theorem of neural networks robust adaptive control is first given as follows:

Theorem: For system (1) satisfying assumption (a)~(c), if the controller structure is (4), where $\boldsymbol{G}(\boldsymbol{x}) = [\boldsymbol{g}(\boldsymbol{x}) + \boldsymbol{W}_1^T\sigma(\boldsymbol{V}_1^T\bar{\boldsymbol{x}}),\cdots,\boldsymbol{W}_m^T\sigma(\boldsymbol{V}_m^T\bar{\boldsymbol{x}})]^{-1}$ and $F(\boldsymbol{x}) = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{W}^T\sigma(\boldsymbol{V}^T\bar{\boldsymbol{x}})$, and the parameter adaptive rules are

$$\begin{aligned}&\dot{\boldsymbol{W}} = \gamma_W\left[\left(\boldsymbol{\sigma}(\boldsymbol{V}^T\bar{\boldsymbol{x}}) - \dot{\boldsymbol{\sigma}}\boldsymbol{V}^T\bar{\boldsymbol{x}}\right)\boldsymbol{E}^T\boldsymbol{P}\boldsymbol{B} - \kappa_W\left\|\boldsymbol{E}\right\|\cdot\boldsymbol{W}\right]\\ &\dot{\boldsymbol{V}} = \gamma_V\left(\bar{\boldsymbol{x}}\boldsymbol{E}^T\boldsymbol{P}\boldsymbol{B}\boldsymbol{W}^T\dot{\boldsymbol{\sigma}} - \kappa_V\left\|\boldsymbol{E}\right\|\cdot\boldsymbol{V}\right)\\ &\dot{\boldsymbol{W}}_i = \gamma_{Wi}\left[-\left(\boldsymbol{\sigma}(\boldsymbol{V}_i^T\bar{\boldsymbol{x}}) - \dot{\boldsymbol{\sigma}}\boldsymbol{V}_i^T\bar{\boldsymbol{x}}\right)\boldsymbol{E}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{B} - \kappa_{W}\left\|\boldsymbol{E}\right\|\cdot\boldsymbol{W}_i\right]u_i\ \cdot\\ &\dot{\boldsymbol{V}}_i = \gamma_{Vi}\left(-\bar{\boldsymbol{x}}\boldsymbol{E}^T\boldsymbol{P}\boldsymbol{B}\boldsymbol{W}_i^T\dot{\boldsymbol{\sigma}} - \kappa_{Vi}\left\|\boldsymbol{E}\right\|\cdot\boldsymbol{V}_i\right)u_i \qquad i=1,\cdots,m\\ &\dot{\boldsymbol{\delta}} = \gamma_\delta\boldsymbol{B}^T\boldsymbol{P}\boldsymbol{E}\end{aligned} \tag{19}$$

where $\gamma_w,\gamma_v,\gamma_{Wi},\gamma_{Vi},\gamma_\delta,\kappa_w,\kappa_v,\kappa_{Wi},\kappa_{Vi}>0, i=1,\cdots,m$, then the close-loop system is stable , all the signals in the close-loop system is bounded and the output tracking error asymptotically converge to a neighborhood of zero.

Proof: Choosing the Lyapunov function as

$$\begin{aligned} V=&\frac{1}{2}\boldsymbol{E}^T\boldsymbol{PE}+\mathrm{tr}(\tilde{\boldsymbol{W}}^T\tilde{\boldsymbol{W}})/2\gamma_1+\mathrm{tr}(\tilde{\boldsymbol{V}}^T\tilde{\boldsymbol{V}})/2\gamma_2 \\ &+\sum_{i=1}^m\left(\mathrm{tr}(\tilde{\boldsymbol{W}}_i^T\tilde{\boldsymbol{W}}_i)/2\gamma_{Wi}+\mathrm{tr}(\tilde{\boldsymbol{V}}_i^T\tilde{\boldsymbol{V}}_i)/2\gamma_{Vi}\right)+\tilde{\boldsymbol{\delta}}^T\tilde{\boldsymbol{\delta}}/2\gamma_\delta \end{aligned} \quad . \tag{20}$$

where $\gamma_1,\gamma_2,\gamma_{Wi},\gamma_{Vi},\gamma_\delta>0, i=1,\cdots,m$, P is a symmetric positive define matrix which satisfies the Lyapunov equation: $\boldsymbol{A}^T\boldsymbol{P}+\boldsymbol{PA}=-\boldsymbol{Q},\quad \boldsymbol{Q}=\boldsymbol{Q}^T>0$

In terms of (15), we have

$$\dot{\boldsymbol{W}}=-\dot{\tilde{\boldsymbol{W}}},\ \dot{\boldsymbol{V}}=-\dot{\tilde{\boldsymbol{V}}},\ \dot{\boldsymbol{W}}_i=-\dot{\tilde{\boldsymbol{W}}}_i,\ \dot{\boldsymbol{V}}_i=-\dot{\tilde{\boldsymbol{V}}}_i, i=1,\cdots,m . \tag{21}$$

In addition, the following inequality is hold

$$\mathrm{tr}(\tilde{\boldsymbol{Z}}^T\boldsymbol{Z})=\mathrm{tr}\left(\tilde{\boldsymbol{Z}}^T(\boldsymbol{Z}^*-\tilde{\boldsymbol{Z}})\right)\le\left\|\boldsymbol{Z}^*\right\|_F\left\|\tilde{\boldsymbol{Z}}\right\|_F-\left\|\tilde{\boldsymbol{Z}}\right\|_F^2\le Z_M\left\|\tilde{\boldsymbol{Z}}\right\|_F-\left\|\tilde{\boldsymbol{Z}}\right\|_F^2 . \tag{22}$$

where $\boldsymbol{Z}$ represents one of $\boldsymbol{W},\boldsymbol{V},\ \boldsymbol{W}_1,\boldsymbol{V}_1,\cdots,\boldsymbol{W}_m,\boldsymbol{V}_m$.

Differentiate (20) with respect to time and substitute(10),(9),(12),(13),(14),(15),(16),(17),(18),(19),(21) and (22) into it, we have

$$\begin{aligned} \dot{V}\le&-\frac{1}{2}\boldsymbol{E}^T\boldsymbol{QE}+\kappa_W\left\|\boldsymbol{E}\right\|\left(M_W\left\|\tilde{\boldsymbol{W}}\right\|_F-\left\|\tilde{\boldsymbol{W}}\right\|_F^2\right)+\kappa_V\left\|\boldsymbol{E}\right\|\left(M_V\left\|\tilde{\boldsymbol{V}}\right\|_F-\left\|\tilde{\boldsymbol{V}}\right\|_F^2\right) \\ &+\sum_{i=1}^m\left[\kappa_{Wi}\left\|\boldsymbol{E}\right\|\left(M_{Wi}\left\|\tilde{\boldsymbol{W}}_i\right\|_F-\left\|\tilde{\boldsymbol{W}}_i\right\|_F^2\right)\left\|u_i\right\|+\kappa_{Vi}\left\|\boldsymbol{E}\right\|\left(M_{Vi}\left\|\tilde{\boldsymbol{V}}_i\right\|_F-\left\|\tilde{\boldsymbol{V}}_i\right\|_F^2\right)\left\|u_i\right\|\right] \\ &+\left\|\boldsymbol{E}\right\|\cdot\left\|\boldsymbol{PB}\right\|\left(c_1+c_2\left\|\tilde{\boldsymbol{W}}\right\|_F+c_3\left\|\tilde{\boldsymbol{V}}\right\|_F\right) \\ &+\sum_{i=1}^m\left\|\boldsymbol{E}\right\|\cdot\left\|\boldsymbol{PB}\right\|\left(d_{1i}+d_{2i}\left\|\tilde{\boldsymbol{W}}_i\right\|_F+d_{3i}\left\|\tilde{\boldsymbol{V}}_i\right\|_F\right)\left\|u_i\right\| \end{aligned}$$

For convenience, the products of constant $c_1,c_2,c_3,d_{11},d_{21},d_{31},\cdots,d_{1m},d_{2m},d_{3m}$ and $\left\|\boldsymbol{PB}\right\|$ are still denoted by $c_1,c_2,c_3,d_{11},d_{21},d_{31},\cdots,d_{1m},d_{2m},d_{3m}$ respectively, then

$$\dot{V}\le-\left\|\boldsymbol{E}\right\|\cdot\left[\begin{array}{l}\frac{1}{2}\lambda_{Qmin}\left\|\boldsymbol{E}\right\|+\kappa_W\left(\left\|\tilde{\boldsymbol{W}}\right\|_F-\frac{1}{2}(\frac{c_2}{\kappa_W}+M_W)\right)^2+\kappa_V\left(\left\|\tilde{\boldsymbol{V}}\right\|_F-\frac{1}{2}(\frac{c_3}{\kappa_V}+M_V)\right)^2 \\ +\sum_{i=1}^m\left[\kappa_{Wi}\left\|u_i\right\|\left(\left\|\tilde{\boldsymbol{W}}_i\right\|_F-\frac{1}{2}(\frac{d_{2i}}{\kappa_{Wi}}+M_{Wi})\right)^2+\kappa_{Vi}\left\|u_i\right\|\left(\left\|\tilde{\boldsymbol{V}}_i\right\|_F-\frac{1}{2}(\frac{d_{3i}}{\kappa_{Vi}}+M_{Vi})\right)^2\right]-\theta\end{array}\right] \tag{23}$$

.

where

$$\begin{aligned} \theta=&c_1+\frac{1}{4}\kappa_W(\frac{c_2}{\kappa_W}+M_W)^2+\frac{1}{4}\kappa_V(\frac{c_3}{\kappa_V}+M_V)^2 \\ &+\sum_{i=1}^m\left(d_{1i}+\frac{1}{4}\kappa_{Wi}(\frac{d_{2i}}{\kappa_{Wi}}+M_{Wi})^2+\frac{1}{4}\kappa_{Vi}(\frac{d_{3i}}{\kappa_{Vi}}+M_{Vi})^2\right)\left\|u_i\right\| \end{aligned} \quad .$$

Let

$$\frac{2\theta}{\lambda_{Qmin}} = a_1, \sqrt{\frac{\theta}{\kappa}} + \frac{1}{2}\kappa_W\left(\frac{c_2}{\kappa_W} + M_W\right) = a_2, \sqrt{\frac{\theta}{\kappa}} + \frac{1}{2}\kappa_V\left(\frac{c_3}{\kappa_V} + M_V\right) = a_3 \tag{24}$$
$$\sqrt{\frac{\theta}{\kappa_{Wi}\|u_i\|}} + \frac{1}{2}\kappa_{Wi}\left(\frac{d_{2i}}{\kappa_{Wi}} + M_{Wi}\right) = \eta_{Wi}, \sqrt{\frac{\theta}{\kappa_{Vi}\|u_i\|}} + \frac{1}{2}\kappa_{Vi}\left(\frac{d_{3i}}{\kappa_{Vi}} + M_{Vi}\right) = \eta_{Vi} \cdot$$
$$i = 1, \cdots, m$$

So long as one of the following inequalities is hold

$$\|\boldsymbol{E}\| \geq a_1,\ \|\tilde{\boldsymbol{W}}\|_F \geq a_2,\ \|\tilde{\boldsymbol{V}}\|_F \geq a_3,\ \|\tilde{\boldsymbol{W}}_i\|_F \geq \eta_{Wi},\ \|\tilde{\boldsymbol{V}}_i\|_F \geq \eta_{Vi},\ i = 1, \cdots, m \cdot \tag{25}$$

then $\dot{V} < 0$. If $\|u\| \in L_\infty, \theta \in L_\infty$ is hold, then the closed-loop system is stable.

Define error vector and error neighborhood of zero as follows:

$$z = \left(\|\boldsymbol{E}\|, \|\tilde{\boldsymbol{W}}\|_F, \|\tilde{\boldsymbol{V}}\|_F, \|\tilde{\boldsymbol{W}}_1\|_F, \|\tilde{\boldsymbol{V}}_1\|_F, \cdots, \|\tilde{\boldsymbol{W}}_m\|_F, \|\tilde{\boldsymbol{V}}_m\|_F\right)^T$$
$$\omega = z \in \left\{\|\boldsymbol{E}\| < a_1 \times \|\tilde{\boldsymbol{W}}\|_F < a_2 \times \|\tilde{\boldsymbol{V}}\|_F < a_3 \times \|\tilde{\boldsymbol{W}}_1\|_F < \eta_{W1} \times \|\tilde{\boldsymbol{V}}_1\|_F < \eta_{V1} \times \cdots\cdots \times \|\tilde{\boldsymbol{W}}_m\|_F < \eta_{Wm} \times \|\tilde{\boldsymbol{V}}_m\|_F < \eta_{Vm}\right\}$$

$\bar{\omega}$ is the complementary of ω. If $z \in \bar{\omega}$, then $\|\boldsymbol{E}\| \neq 0$, there exist a constant $\alpha > 0$ such that $\dot{V} = -\alpha\|\boldsymbol{E}\|$, and $\int_0^t \dot{V} dt = V(t) - V(0) < e^{-\alpha\|E\| t}$, hence $V(t) < V(0)$, Due to $V(0) \in L_\infty$, $V(t) \in L_\infty$ is hold. From (20), we have $\boldsymbol{E}, \tilde{\boldsymbol{W}}, \tilde{\boldsymbol{V}}, \tilde{\boldsymbol{W}}_1, \tilde{\boldsymbol{V}}_1, \cdots, \tilde{\boldsymbol{W}}_m, \tilde{\boldsymbol{V}}_m, \tilde{\boldsymbol{\delta}} \in \boldsymbol{L}_\infty$. Finally we obtain $\boldsymbol{W}, \boldsymbol{V}, \boldsymbol{W}_1, \boldsymbol{V}_1, \cdots, \boldsymbol{W}_m, \boldsymbol{V}_m, \boldsymbol{\delta}, \boldsymbol{x} \in L_\infty$ from (3), (9) and (15). Assume that β is the least value of V on the boundary of ω. In terms of the above integral inequality we have $t \leq \ln(V(0) - \beta)/\alpha\|E\|$. Due to $\|\boldsymbol{E}\| \neq 0$, z converges to ω asymptotically in limited time. According to Eq. (24) ω can be reduced by increasing design parameters. If all the inequalities in (25) are not hold, the set comprised by z is a subset of ω, therefore it diminish as ω decreases.

4 Simulation Example

A two rigid-link robot manipulator is utilized in this study to verify the effectiveness of the proposed control scheme. The dynamic model [7] is as follows:

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{V}_m(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{G}(\boldsymbol{q}) + \boldsymbol{F}(\dot{\boldsymbol{q}}) + \boldsymbol{T}_L(t) = \tau(t)$$

$$\boldsymbol{M}(\boldsymbol{q}) = \begin{bmatrix} l_2^2 m_2 + l_1^2(m_1 + m_2) + 2l_1 l_2 m_2 \cos(q_2) & l_2^2 m_2 + l_1 l_2 m_2 \cos(q_2) \\ l_2^2 m_2 + l_1 l_2 m_2 \cos(q_2) & l_2^2 m_2 \end{bmatrix}$$

$$\boldsymbol{V}_m(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} = \begin{bmatrix} -2l_1 l_2 m_2 \sin(q_2)(\dot{q}_1 \dot{q}_2 + 0.5\dot{q}_2^2) \\ l_1 l_2 m_2 \sin(q_2)\dot{q}_1^2 \end{bmatrix}$$

$$\boldsymbol{G}(\boldsymbol{q}) = \begin{bmatrix} l_1(m_1 + m_2) g\cos(q_1) + l_2 m_2 \cos(q_1 + q_2) \\ l_2 m_2 g\cos(q_1 + q_2) \end{bmatrix}$$

$$\boldsymbol{F}(\dot{\boldsymbol{q}}) = \begin{bmatrix} 12\dot{q}_1 + 0.5\, sgn(\dot{q}_1) & 12\dot{q}_2 + 0.5\, sgn(\dot{q}_2) \end{bmatrix}^T$$

$$\boldsymbol{T}_L(t) = \begin{bmatrix} 5\sin(5t) & 5\sin(5t) \end{bmatrix}^T$$

where q_1 and q_2 are the angle of joints 1 and 2; $m_1 = 0.8$ Kg and $m_2 = 2.3$ Kg are the mass of links 1 and 2; $l_1 = 1.0$ m and $l_2 = 1.0$ m are the length of links 1 and 2; $g = 9.8$ m/s^2 is the gravity acceleration

The objective is to control the angles of joints 1 and 2 to track the desired command $q_{1d} = \sin t$ and $q_{2d} = \cos t$. First, the tracking response are given under the nominal condition ($\boldsymbol{F}(\dot{\boldsymbol{q}})$=0 and $\boldsymbol{T}_L(t)$=0). Secondly, the mass of the link 2 increases 1Kg and the friction forces $\boldsymbol{F}(\dot{\boldsymbol{q}})$ and external forces $\boldsymbol{T}_L(t)$ are considered at 10s. Finally, the manipulator should be settled at $q_{1d} = 0.5\ rad,\ q_{2d} = -0.5\ rad$ from 20s.

The reference model is given as follows

$$\dot{y}_m = A_m y_m + B_m r$$

$$A_m = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -10 & -6.3 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -10 & -6.3 \end{bmatrix} \qquad B_m = \begin{bmatrix} 0 & 0 \\ 10 & 0 \\ 0 & 0 \\ 0 & 10 \end{bmatrix}$$

Adaptive gains: $\gamma_W = \gamma_V = 0.1,\ \gamma_{W_1} = \gamma_{V_1} = \gamma_{W_2} = \gamma_{V_2} = 0.0001,\quad \gamma_\delta = 0,\ \kappa = 1, \kappa_g = 10$

Control parameter: $\lambda = \mathrm{diag}\left([-100 \quad -20] \quad [-500 \quad -45]\right)^T \qquad Q = \mathrm{I}$

Initial values of the weights and the number of hidden layer neurons:

$$W = V = 0,\ W_{11} = 1, W_{12} = 0,\quad V_1 = 0,\ W_{21} = 0,\ W_{22} = 2,\ \ V_2 = 0,\ N = 20$$

where W_{ij} is the weights of the j-th output of the i-th neural network.

Initial value of system states: $x_1 = 0,\ x_2 = 0, x_3 = 0,\ x_4 = 0$.

Considering the dynamics of the actuator, one-order low pass filters are added into plant dynamics and the filter time constant is 0.05s. Fig 2 is the response of link1 where solid line represents the actual response and dash line is the desired command in a), b) is the output tracking error and c) shows the control input torque. Fig 3 is the response of link2 where the denotations are same as in Fig 2.

From the results of simulation, we noticed that the closed system is stable and the output tracking error converge to a small neighborhood of zero by increasing the control parameters. The choice of the reference model is important, if its dynamics is fast than actual dynamics of the controlled plant, the control input may frequent action with large amplitude. The tracking performance is not only dependent on the control parameters but also closed related to the initial value of the neural networks of the input matrix.

5 Conclusions

In this paper, a neural networks robust adaptive control scheme for a class of MIMO nonlinear system with unknown nonlinearity is developed. Multiple single-layer neural networks are used to approximate the uncertainty of the model. The Lyapunov function-based design of adaptive laws guarantees the global stability of the closed-loop system. The size of the residual set for the tracking error depends solely on design parameters and the initial value of the neural networks, which can be chosen to meet desired upper bounds for the tracking error. A design procedure is presented which shows how to choose the various parameters in the control law in order to guarantee that the steady-state tracking error is within prespecified bounds. The on-line adaptive laws of the weights of neural networks are independent on the controlled plant parameters, hence the controller is robust to external disturbance and parameter uncertainty.

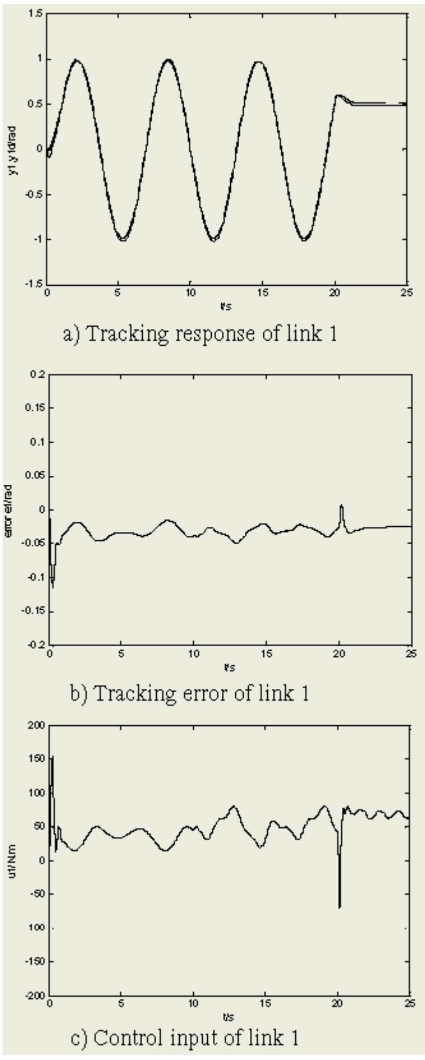


Fig. 2. Response and control of link 1

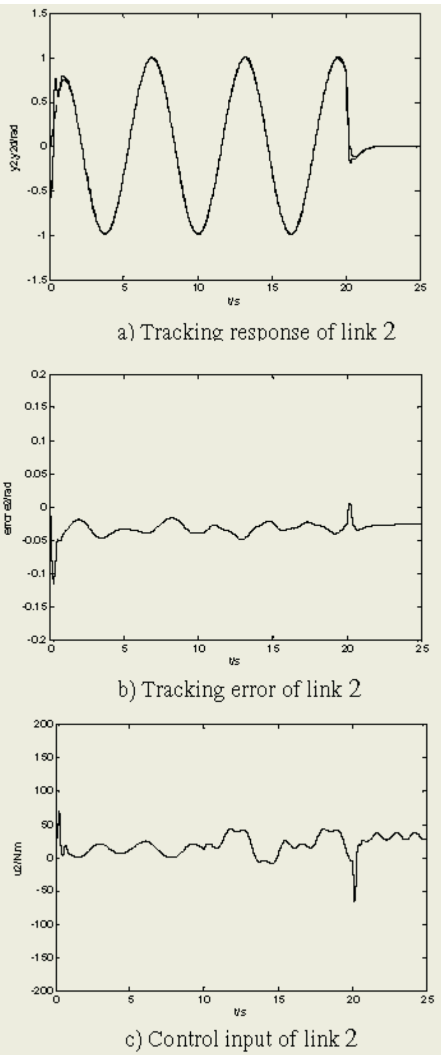


Fig. 3. Response and control of link 2

References

1. Sanner, R.M., Slotine, J.J.E.: Gaussian Networks for Direct Adaptive Control. IEEE Trans. Neural Networks, **3** (1992) 837–863
2. Chen, F.C., Khalil, H.K.: Adaptive Control of a Class of Nonlinear Discrete Time Systems Using Neural Networks. IEEE Trans. Automat. Contr, **40** (1995) 791–801
3. Sadegh, N.: A Perceptron Network for Functional Identification Using Radial Gaussian Networks. IEEE Trans. Neural Networks, **4** (1993) 982–988
4. Ge, S.S., Wang, C.: Adaptive Neural Control of Uncertain MIMO Nonlinear Systems. IEEE Trans. Neural Networks, **15** (2004) 674-692
5. Rovithakis,G.A., Christodulou, M.A.,: Neural Adaptive Regulation of Unknown Nonlinear Dynamical Systems. IEEE Trans. Syst., Man, Cybern. B, **27** (1997) 810–822
6. Lewis, F.L., Liu,K., Yesildirek, A.: Neural Net Robot Controller with Guaranteed Tracking Performance. IEEE Trans. Neural Networks, **6** (1995) 703–715
7. Taylor, D.: Composite Control of Direct-Drive Robots. Proceedings of the IEEE Conference on Decision and Control (1989) 1670-1675

Adaptive Critic for Controller Malfunction Accommodation

Gary G. Yen

Oklahoma State University, School of Electrical and Computer Engineering,
Stillwater, OK, 74078, USA
gyen@okstate.edu

Abstract. Built on the existing framework, we propose a novel supervisory system capable of detecting controller malfunctions before the stability of the plant is compromised. Furthermore, due to its ability to discern between controller malfunctions and faults within the plant, the proposed supervisor acts in a specific fashion in the event of a controller malfunction to provide new avenues with a greater probability of convergence using information from a Dynamic Model Bank. The classification and distinction of controller malfunctions from the faults in the plant itself is achieved through an advanced decision logic based on three independent quality indexes. Proof-of-the-concept simulations over a nonlinear plant demonstrate the validity of the approach.

1 Introduction

Fault Tolerant Control (FTC) is an area of research that emerges to increase *availability* by specifically designing control algorithms capable of maintaining stability and performance despite the occurrence of faults. *Availability*, defined as the probability that a system will operate satisfactorily and effectively at any point in time [1], has become one of the key design criteria of a system. It is now considered in the process industry to be the single factor with the highest impact on profitability. However, as the demand for availability increases, so does the number of tasks each system is designed to accomplish with stringent specifications, leading inevitably to an increase in the overall complexity. At a higher level of complexity, such as autonomous vehicles for off-planet exploration, it is not feasible to have all possible fault scenarios determined during design time. The growing complexity of physical plants and control missions inevitably leads to increasing occurrence, diversity and severity of faults. Therefore, although some fault scenarios are known at design time and can even be predicted if enough information is available, it is not reasonable in real-world applications to assume complete knowledge of the dynamics of the plant over all possible fault scenarios. In addition, while an approximate linear model can often be derived for a plant operating close to its nominal point, nonlinearities introduced or augmented by a fault after its occurrence can become of paramount importance to achieve a successful new control solution. Therefore, a complete FTC architecture must be equipped with adaptive capabilities for the online generation of new nonlinear control solutions in response to unknown fault scenarios.

Neural Network (NN) has been well regarded as an effective tool in function approximation due to its universal nonlinear mapping capability. The questions that arise when designing a NN adaptive controller are how to perform online learning in an effective manner for the most diversified scenarios and how to guarantee stability. Under the present state-of-the-art control designs however, adaptation flexibility and stability are still conflicting specifications and therefore a suitable compromise must be reached. Adaptive critic designs have shown to implement useful approximations of dynamic programming, a method for determining optimal control policies in the context of nonlinear plants. As a candidate solution for the FTC problem, adaptive critic designs are capable of achieving superior performance by combining three different NNs: an identification NN that produces information of the dynamics of faults as they occur, a critic NN to estimate the impact of the underlying control strategy over time; and an action NN to generate real-time control actions to accommodate faults as they emerge. In this paper, Globalized Dual Heuristic Programming (GDHP), appointed as the most complete and powerful adaptive critic design [2], is implemented. In [3], GDHP is used in the architecture shown in Fig. 1 to autonomously generate control solutions for unexpected faults while explicitly producing input-output maps through the use of neural networks. Located inside the supervisor, the DMB stores, for the nominal plant and each known fault scenario (added during design time or learned online), an input-output identification map and a controller that has been successfully designed for each specific scenario. The main contribution of FDD to the overall FTC architecture is to differentiate abrupt from incipient faults. Abrupt faults are characterized by sudden changes in the dynamics of the plant while incipient faults cause slow and continuing variations on plant parameters. Independent of its particular implementation, the adaptive nonlinear controller then becomes a vulnerable point of the FTC architecture as it may diverge or converge to local minima through the course of its online adaptation. If left unchecked, such situation may lead to loss of availability even when recovery of plant stability was still reachable.

In order to account for such deficiency, we propose in this paper to add to the FDD scheme the capability of detecting malfunctions within the nonlinear adaptive controller itself. In the case of implementations involving the online training of NNs in particular, there are two scenarios that must be avoided. The first relates to the training process converging to a local minima that prevents the NN weights to reach the global optimal, while the second refers to weight divergence due to the application of the training algorithm in close loop with unknown dynamics of a faulty plant. To properly identify these two control malfunctions and discern them from the plant faults, a third quality index is introduced. By measuring the degree of activity within the NNs that compose the GDHP architecture, the weight quality index is capable of adding to the information gathered by the other two quality indexes to achieve such goals. Once a control malfunction is detected and identified, it is then possible to use the information from the DMB to provide the training algorithm with a new set of initial conditions that represents the closest possible known dynamics and effectively prevent divergence and greatly increase the chance of recovery of stability and performance. Due to space limitation, the details of GDHP controller will not be elaborated here. The remainder of the paper is organized as follows. In Section 2, the proposed supervisor scheme is described, and its interactions with the controller are elaborated in detail. Section 3 presents a numerical example of a complex nonlinear system through which the improved tolerance of the GDHP against both faults and controller malfunctions can be seen. Section 4 concludes the paper with pertinent observations.

2 Supervisor

The supervisor system used to perform FDD, detect controller malfunctions and coordinate the DMB is introduced in this section. One of the main contributions of this paper involves the controller malfunction detection and capability to suggest solutions on-the-fly. To better understand the functionality of the supervisor, it can be divided into three layers. The first layer collects and analyzes data from the plant and the controller block. The second one is responsible for the decision making, and the final layer devises ways to implement the resolutions.

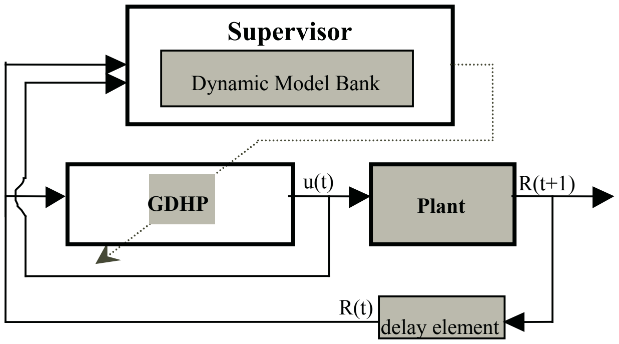


Fig. 1. General diagram of the proposed scheme

The first layer receives the sampled output of the plant $R(t)$ and a delayed input $u(t-1)$, computes three quality indices and indicates the known scenario that better approximates the current dynamics. The indexes generated by the first layer are the control quality index $q_c(t)$, the identification quality index $q_i(t)$ and the weight quality index $q_w(t)$. The control quality index measures the reconfigurable controller ability to track the desired trajectory by performing a decaying integration of the primary utility function,

$$q_c(t) = \int_0^t e^{-\xi_c(t-\tau)} U(\tau) d\tau, \tag{1}$$

where $0 < \zeta_c < 1$ is a time decay factor. In this structure, greater time decay factors will lead to $q_c(t)$ being more affected by the quality of control actions related to instances further into the past. Concerning the occurrence of abrupt faults, adopting a ζ_c closer to the unity will lead to a more conservative supervisor that will observe the reconfigurable controller longer before adding a new solution to the DMB and will also wait for a longer period of poor performance before detecting a fault. For the calculation of the identification quality index, the delayed input is then fed into the DMB. The DMB contains information on the plant under nominal condition and under all the known fault scenarios, organized in the form of copies of the IdNN, AcNN and CrNN used to control the plant under each situation. To guarantee their specialization, no additional training is performed on networks once inserted into the DMB. Each one of the IdNNs in the DMB is then used to generate an identification error.

For each of those, a decaying integration is used, and the results are compared. As shown in (2), the smallest identification error history defines the identification quality index, and the corresponding model m is appointed as the switching candidate.

$$q_i(t) = \min_{m \in M} \left(\int_0^t e^{-\zeta_i (t-\tau)} \left| \hat{R}^m(\tau) - R(\tau) \right| d\tau \right), \tag{2}$$

where $0 < \zeta_i < 1$ is a time decay factor, and $\hat{R}^m(t)$ is the vector of outputs predicted by the IdNN *m*, which in turn is an element of the set of *M* models in the DMB. The identification decay factor ζ_i can also be used to fine-tune the behavior of the supervisor. Greater identification decay factors will also lead to a more conservative supervisor in the sense that it will require more data points to conclude that the observed plant dynamics match the ones described by one of the known faults.

Finally, Equation (3) shows how the weight quality index is calculated in order to measure the amount of activity within the three continuously adapting neural networks (i.e. IdNN, AcNN and CrNN#1).

$$q_w(t) = \int_{t_s}^{t} e^{-\zeta_w (t-\tau)} \left[\left\| w^i(\tau) - w^i(\tau - \Delta t) \right\| + \left\| w^a(\tau) - w^a(\tau - \Delta t) \right\| + \left\| w^c(\tau) - w^c(\tau - \Delta t) \right\| \right] d\tau, \tag{3}$$

where ζ_w is a time decay factor in the range [0,1], Δt is the sampling period and t_s is the time of occurrence of the latest switching operation. The presence of t_s is a key feature to understand the interrelations between the detection and diagnosis of faults inside the plant and the detection of faults within the controller in what is called controller malfunctions. In essence, t_s resets the memory of the filter used in the calculation of $q_w(t)$ to prevent the switching operations carried out by the supervisor (in response to faults within the plant or controller) to be perceived as major changes in the weight structure, which are associated with controller divergence during neural network training.

In the second layer, FTC design parameters are converted into thresholds to distinguish high (Hq_c, Hq_i) and low (Lq_c, Lq_i) values for the control and identification quality indexes. The threshold for $q_c(t)$ define what is to be considered as an acceptable performance, while the one for $q_i(t)$ stipulate the degree of similarity of the input-output behavior that should be used to consider two models equivalent. For the weight quality index, two thresholds are used to distinguish between high activity Hq_w, standard activity Sq_w and low activity Lq_w. Standard activity is adjusted as a broad range within which the activity of the networks remain during successful training. The controller malfunction detection takes precedence over the FDD, ensuring that only a functional GDHP controller is used for fault recovery or compensation and that malfunction within the IdNN do not interfere with the FDD process. By using the information contained in the controller and the weight quality indexes, it is possible to verify the condition of the GDHP nonlinear adaptive controller over two different malfunctions. The first one relates to the online GDHP training converging to a local minima and therefore being incapable of reaching the optimal tracking error. Such malfunction is characterized by a high control quality index matched by a low weight quality index. The second malfunction relates to controller divergence and is marked by both a high control quality index and a high weight quality index. Fig. 2 displays a flow chart that illustrates the controller malfunction detection procedure engaged at every iteration. Note that in a healthy training situation, a high control quality index, and therefore elevated tracking error over time, would generate changes in the weight structure of the neural networks that compose the GDHP controller and therefore indicate Sq_w. In order to increase noise rejection and reduce the probability of false alarms, an observation time is used during which such conditions must remain unaltered so that a controller malfunction can be positively detected.

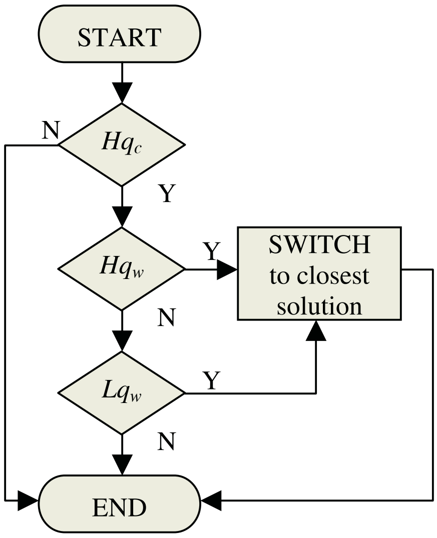


Fig. 2. Flowchart for controller malfunction detection

Assuming that no control malfunctions are detected, FDD is then performed through the decision process. Note that only the control and identification quality indexes are used for FDD purpose, generating a decision logic based on four different states. It's important to call to attention that, in this formulation, FDD and the decisions of when to switch and add a model to the DMB take place in the 12 transitions between states. State 1 (Lq_c, Lq_i) is characterized by the reconfigurable controller performing satisfactorily over a known environment (i.e. with a matching model in the DMB). While in State 1, an abrupt fault may cause the performance to be degraded enough for the controller quality index $q_c(t)$ to be classified as high. If in this case $q_i(t)$ also exceeds its high threshold (implying that no model in the DMB matches the current dynamics), the abrupt fault is classified as unknown and State 2 is reached. In State 2 the supervisor does not possess specific knowledge to provide to the GDHP controller, which is then responsible to generate a control solution to this new fault scenario. On the other hand, if $q_i(t)$ remains low, the abrupt fault is declared as known, and the decision logic moves to State 4. Switching takes place in this transition by loading a previously computed (online or off-line) solution from the

DMB to the GDHP controller. If more than one model provides a good quality index identifying more than one possible known fault dynamics, Equation (2) will directly make the model whose dynamics better match the observed to cross the threshold sooner and therefore the supervisor will shift to the one who is first identified. In case of more than one model crossing the threshold during a single sampling period, the model with smaller quality index reading is selected. The decision process then remains in State 4 until either the performance is recovered or another fault takes place before that is achieved guaranteeing that chattering behavior due to continuous shifting does not take place.

The plant working under satisfactory performance levels while in an unknown scenario characterizes State 3. There are two distinct events that can lead the plant to State 3, the first being the occurrence of an incipient fault and the second the development of a control solution for a previously unknown fault. Incipient faults, often connected to component aging, may be gradually adapted by the reconfigurable controller and eventually indicate a high $q_i(t)$, even though $q_c(t)$ remains low during all the process (transition from State 1 to 3). In this case, there is no purpose in learning a new environment/controller pair since the parameters are continuously changing. As a matter of fact, if allowed to learn all the transient models, the DMB might rapidly grow to an intractable size with models that do not represent distinct fault scenarios.

When the GDHP controller is adapting to a new environment (State 2), $q_c(t)$ is expected to decrease to the point where it crosses its lower threshold (transition to State 3), and a new set of parameters is added to the DMB. In this case the plant will only remain in State 3 for a few iterations since $q_i(t)$ will decrease rapidly due to the new model in the bank. There are however two other possible outcomes for the system in State 2. The first one addresses to the possibility of an abrupt known fault to happen before the fault currently active is completely dealt with. In this case $q_i(t)$ reaches a low value prior to $q_c(t)$ and switching to the known environment takes place. The second scenario addresses the situation in which the performance remains below the desired level. Even if an ideal reconfigurable controller is available, this situation can still occur as the desired performance level might become unreachable due to actuator or physical limitations in a particular fault scenario. In such case, the decision logic remains in State 2, as the quality of the control cannot be improved by supervisor intervention.

The third layer manipulates the DMB by making new entries and by switching to the reconfigurable controller indicated by the first layer, when requests arrive from the second. Switching is implemented by loading a complete set of parameters of the three neural networks (i.e. IdNN, AcNN and CrNN) to the GDHP algorithm currently being used. The fact that the controller is switched to one devised to a similar plant and the natural generalization capabilities of neural networks add to improved stability when the parameters are loaded as new initial conditions to the adaptive process. The DMB also stored copies of all the gradients required when updating the networks using backpropagation through time. Uploading the gradients also works to increase switching smoothness since more information about the new dynamics of the plant is supplied.

3 Numerical Example

The simulated plant possesses two inputs, three states, and two outputs which are expected to track two independent trajectories given below

$$R_1^t = \mu_1 \sin\left(t\pi/250\right) + 0.4\sin\left(t\pi/125\right),\quad R_2^t = 0.1 + \mu_2 \sin\left((t+150)\pi/250\right) + 0.6\sin\left((t+190)\pi/125\right) \quad (4)$$

where μ_1 and μ_2 assume values in the interval [0, 0.5] randomly selected once at every 500 iterations. For the sake of the presented simulation, tracking is considered satisfactory if the mean error over a cycle of 500 iterations is less than 0.001.

Through the course of the simulation, we introduce two abrupt nonlinear faults: Abrupt Fault 1 (AF1) that changes the system's dynamics to (5), and Abrupt Fault 2 (AF2) that modifies it to (6). For both faults, the changes in the dynamics take place instantly at their time of occurrence.

$$x_1(t+1) = 1.5u_1(t) + x_1(t) - x_3^2(t) - \sin\left(\mu_1(t)\pi/4\right),\ x_2(t+1) = 0.2x_2(t) + x_2(t)u_1(t) + 2\left(x_3(t)/1+x_3^2(t)\right) \quad (5a)$$

$$x_3(t+1) = u_2(t) + 0.6u_2(t)\sin(0.5x_1(t)) + 0.4x_3(t) \quad (5b)$$

$$x_1(t+1) = u_1(t) + x_3^2(t),\ x_2(t+1) = x_3(t) - 0.8x_2(t)u_1(t),\ x_3(t+1) = 1.5u_2(t) - 0.5x_3(t) \quad (6)$$

To simulate the occurrence of an incipient fault (IF), the dynamics of the nominal system are gradually modified over the course of 10^4 iterations as described by (7), where t_i is used to adjust the time of occurrence of the IF so that by the end of its interval of occurrence the nonlinear terms take full effect. The use of the tansig function creates a smooth gradual introduction of the incipient dynamics with steep changes in the middle range. Note that not only new nonlinear dynamics are introduced, but also the states of the system become coupled demanding more complex controller action.

$$x_1(t+1) = u_1(t) + (0.5 + 0.5\tan sig(t - t_i/1500))\frac{x_1(t)}{1+x_1^2(t)}u_2(t), \quad (7a)$$

$$x_2(t+1) = x_3(t) + (0.5 + 0.5\tan sig(t - t_i/1500))\frac{\sin(4x_3(t))}{2}x_3(t),\ x_3(t+1) = u_2(t) \quad (7b)$$

The simulation starts with the plant under nominal dynamics for the first 15,000 iterations. The GDHP controller is initialized with a set of weights previously designed for the nominal dynamics. The DMB inside the supervisor is initialized with a copy of the same nominal weights and, therefore, it is initialized with the knowledge of how to identify and control the plant under nominal dynamics. Hence, both $q_i(t)$ and $q_c(t)$ start with low values corresponding to state 1 in the FDD decision logic. In order to demonstrate the supervisor responses to all key circumstances it is designed for, faults were introduced into the system according to the schedule in Table 1. Although displaying the entire 65,000 iterations of the simulation would be optimal for the visualization of the complete challenge set out for the proposed architecture, its length prevents it from being printed in full and therefore only key intervals are illustrated with the relevant graphs. At iteration 15,000 the plant dynamics abruptly change to those of AF1. Since the GHDP controller is continuously adapted online, as soon as the dynamics are modified the weights of the three neural networks start being adjusted in order to search for a control solution for this scenario. During learning, the tracking error grows, leading to Hq_c. Concomitantly, the change in the dynamics leads also to an increase in the identification error represented by $q_i(t)$ until Hq_i is reached. In this manner, state 2 of the FDD decision logic is reached and an abrupt unknown fault is detected and correctly identified. Note that, although possible in a discreet system, both quality indexes need not reach Hq_i and Hq_c at the same iteration. For instance, in the discussed simulation Hq_c was reached before Hq_i, leading the FDD decision logic to transitorily assume state 4 at iteration 15,002 before reaching state 2 at iteration 15,005. The transition through state 4 leads to a momentary misdiagnosis of the fault as an abrupt *known* fault and issues a switch command, however the fault diagnosis is corrected three iterations later and the effects of the switching are almost imperceptive since the GHDP programming has not yet had sufficient time to reconfigure from controlling the plant under nominal dynamics. Throughout the learning process, while $q_c(t)$ is high, $q_w(t)$ remains between the low and high thresholds indication healthy levels of reconfiguration within the neural networks and therefore no controller malfunction is detected. Indeed, at 18,475 with Lq_c the GDHP controller manages to develop a satisfactory solution to the tracking problem, the FDD decision logic goes to state 3 declaring "control success" and adding the current set of weights to the DMB. Since now the supervisor possesses knowledge of the plant under AF1 dynamics, $q_i(t)$ drops below its threshold and the FDD decision logic returns to state 1 (i.e. known dynamics).

Table 1. Simulation schedule for plant dynamics

Plant dynamics	Interval of occurrence (iterations)
nominal	1 to 15000
AF1	15000 to 25000
nominal	25000 to 35000
IF	35000 to 45000

The plant returns to the nominal dynamics in an abrupt fashion at iteration 25,000, depicting a hypothetical situation in which the cause of a fault is removed from the system ceasing its effect on the dynamics. This transition differ from the one at iteration 15,000 discussed previously in the fact that the DMB has knowledge of the new dynamics and therefore can switch to the known solution as soon as it is identified. To highlight the reduction in reconfiguration time and cumulative tracking error brought by the supervisor's intervention, Fig. 3(a) displays the trajectories of the output of a parallel simulation run without the supervisor. With only the GDHP controller striving alone to reconfigure itself, tracking performance is recovered at 1,659 iterations after the alteration in the dynamics. Fig. 3(b), on the other hand, shows the outcome of the main simulation in which the FDD decision logic identifies the change in the dynamics as being an abrupt known "fault" (in this case the new dynamics are of the nominal plant) at iteration 25,247 and switches the weights of all three neural networks with the ones designed for the nominal dynamics stored in the DMB since its initialization. As a consequence, the GHDP controller is able to reach the desired tracking performance at 734 iterations after the change in dynamics. In comparison, switching as determined by the supervisor led to a reduction of 55.76% in the reconfiguration time and a reduction of 74.01% in the cumulative tracking error in the first 1,500 iterations after the alteration of the dynamics as illustrated in Fig. 3(c).

In the interval between iterations 35,000 and 45,000 IF gradually modifies the plant dynamics and is correctly identified as an incipient fault by the supervisor. The correct fault diagnosis prevents the supervisor from switching to any solutions already stored in the DMB, an action that in this case would disrupt the successful continuous adaptation of the GDHP controller to counter the growing effects of IF. The successful maintenance of the tracking performance well below the acceptable level throughout this period is illustrated in Fig. 3.

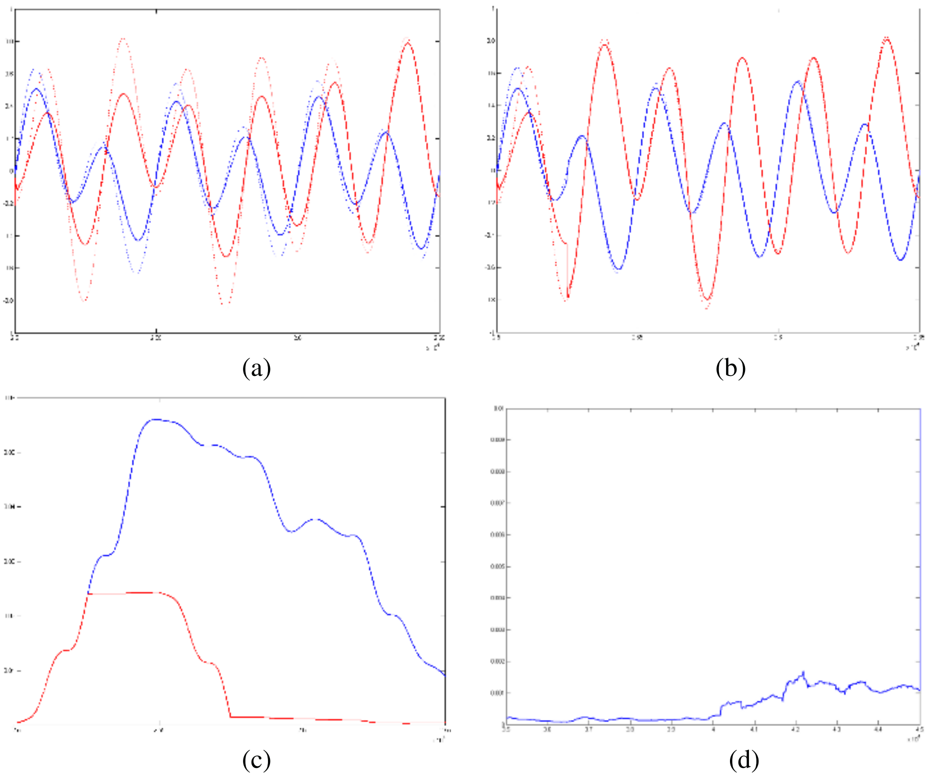


Fig. 3. Comparison of outputs $R_1(t)$ (blue) and $R_2(t)$ (red) and respective desired trajectories (dotted) of simulations without (a) and with (b) supervisory intervention. Switching impact illustrated by (c), the average tracking error for the simulation without (blue) and with (red) supervisory intervention. (d) Average tracking error during IF application.

4 Conclusion

The presented work has demonstrated that the implementation of a synergistically combination of an GDHP controller and a supervisor based on three distinct quality indexes generates an efficient and reliable Fault Tolerant Control architecture. As demonstrated in the nonlinear plant simulations, the introduction of the weight quality index has made possible to distinguish between faults in the plant and controller malfunctions caused by online training divergence or local minima convergence. Further more, the Dynamic Model Bank was successfully used to generate new initial conditions to the neural network training that improve their efficacy as the supervisor autonomously acquires more nonlinear models of the plant under healthy and diverse faulty scenarios. Although the results so far have been greatly encouraging, qualitative analysis of the complete combined online stability and real-world complications is essential and will be carried out in future research.

References

1. Åström, K., Albertos, P., Blanke, M., Isidori, A., Schaufelberger, W., and Sanz, R. (Eds.), Control of Complex Systems, Springer, London, UK, 2001.
2. Prokhorov, D., Santiago, R., and Wunsch, D.: Adaptive Critic Designs: A Case Study for Neurocontrol. Neural Networks, 8 (1995) 1367-1372
3. Yen, G., and DeLima, P.: Improving the Performance of Globalized Dual Heuristic Programming for Fault Tolerant Control though And Online Learning Supervisor. to appear in Trans. Automation Science and Engineering, 2 (2005)

Output Based Fault Tolerant Control of Nonlinear Systems Using RBF Neural Networks

Min Wang and Donghua Zhou

[1]Department of Automation, Tsinghua University, Beijing 100084, China
wang-m02@mails.tsinghua.edu.cn

Abstract. In this paper, an output based fault tolerant controller using radius basis function (RBF) neural networks is proposed which eliminates the assumption that all the states are measured given in Polycarpou's method. Inputs of the neural network are estimated states instead of measured states. Outputs of the neural network compensate the effect of a fault. The closed-loop stability of the scheme is established. An engine model is simulated in the end to verify the efficiency of the scheme.

1 Introduction

In recent years, the research of fault-tolerant control of nonlinear systems has become an active area. Much technique has been used to achieve fault tolerance such as adaptive control [1], nonlinear H_∞ control [2], artificial intelligence [3],[4],[5] and so on. Among them, the learning approach using neural networks has achieved relatively more results [3],[4]. The learning ability of neural networks was used to estimate faults, in which inputs of the neural networks are measured states and outputs of the neural networks compensate the effect of faults. The closed-loop stability can be established using Lyapunov theory. However, this scheme assumes that all the states of the system are available that is not always true in practice as indicated by Polycarpou himself [3]. This fact motivates us to handle the occasions that not all the states can be measured. On the other hand, the research of adaptive output feedback control has achieved some results [6],[7],[8] that may be helpful to solve our problem, i.e., output based fault tolerant control of nonlinear systems.

In this paper, an adaptive control method proposed by Seshagiri [8] is modified to solve our problem. The closed-loop stability can also be established.

2 Preliminaries and Problem Formulations

Consider the following system as a single input form of the system studied in [3].

$$\dot{z} = \zeta_0(z) + G_0(z)[u + \eta_0(z,t) + \beta(t-T)f_0(z)], \quad z(0) = z_0 . \tag{1}$$

with state vector $z \in R^n$ and input $u \in R$.

$\zeta_0(z)$, $G_0(z)$	known sufficiently smooth continuous functions;
$\eta_0(z,t): R^n \times R^+ \to R$	modeling uncertainty;
$\beta(t-T)$	step function representing an abrupt fault;
$f_0(z): R^n \to R$	continuous function representing an unknown fault.

Under the assumption that all the states are available, a fault tolerant controller was designed in [3]. The closed-loop stability was established. In order to perform output based controller design, we augment (1) with an output equation

$$y = h(z), y \in R . \tag{2}$$

$h(z)$ is a sufficiently smooth function.

Assumption 1. The *relative degree*[1] of system (1)-(2) is n over the domain of interest.

Under assumption 1, the system can be transformed into the following form through a nonlinear diffeomorphism $z = T(x)$ [9].

$$\begin{aligned} \dot{x}_i &= x_{i+1} \qquad 1 \le i \le n-1 \\ \dot{x}_n &= \zeta(x) + G(x)[u + \eta(x,t) + \beta(t-T) f(x)] \\ y &= x_1 \end{aligned} \tag{3}$$

We can see that the form of (3) is similar to the system considered in Seshagiri's work [8]. In [8], output based adaptive tracking problem of a class of nonlinear system is considered. RBF neural networks are used to estimate unknown functions that represent the plant dynamics. Thus, we can expect that a modified Seshagiri's method can solve our problem, i.e., output based fault tolerant control of (3).

In this paper, we design an output based fault tolerant controller so that y tracks a given reference signals y_r in the presence of fault. Let

$$\mathcal{Y}_r(t) = [y_r(t),\ y_r^{(1)}(t), \cdots,\ y_r^{(n-1)}(t)]^T . \tag{4}$$

Let $e = x - \mathcal{Y}_r$. When $t > T$, we rewrite (3) as

$$\begin{aligned} \dot{e} &= A_m e + b\{Ke + \zeta(e+\mathcal{Y}_r) + G(e+\mathcal{Y}_r)[u + \eta(e+\mathcal{Y}_r,t) + f(e+\mathcal{Y}_r)] - y_r^{(n)}\} \\ y &= Ce \end{aligned} \tag{5}$$

where

$$A = \begin{bmatrix} 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}, b = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}, K \triangleq [k_1, k_2, \cdots, k_n], C = \begin{bmatrix} 1 & \cdots & 0 & 0 \end{bmatrix} \tag{6}$$

$A_m = A - bK$ is Hurwitz. Let $P = P^T > 0$ be a solution of the Lyapunov equation $PA_m + A_m^T P = -I$.

[1] The definition of relative degree is given in the appendix.

A high gain observer that is the same as the one used in [8] is utilized to estimate the tracking error

$$\begin{aligned} \dot{\hat{e}}_i &= \hat{e}_{i+1} + \frac{\alpha_i}{\varepsilon^i}(e_1 - \hat{e}_1), \quad 1 \le i \le n-1 \\ \dot{\hat{e}}_n &= \frac{\alpha_n}{\varepsilon^n}(e_1 - \hat{e}_1) \end{aligned} \quad . \tag{7}$$

Constants α_i are chosen such that the polynomial $s^n + \alpha_1 s^{n-1} + \cdots + \alpha_{n-1}s + \alpha_n$ is Hurwitz. $\varepsilon > 0$ is a design parameter.

Let $\xi_i = (e_i - \hat{e}_i)/\varepsilon^{n-i}, \quad 1 \le i \le n, \xi = [\xi_1, \cdots, \xi_n]^T$, we obtain

$$\begin{aligned} \varepsilon\dot{\xi} &= (A - HC) + \varepsilon\{Ke + \zeta(e + \mathcal{Y}_r) \\ &\quad + G(e + \mathcal{Y}_r)[u + \eta(e + \mathcal{Y}_r, t) + f(e + \mathcal{Y}_r)] - y_r^{(n)}\}, \quad t > T \end{aligned} . \tag{8}$$

where $H = \begin{bmatrix} \alpha_1 & \cdots & \alpha_{n-1} & \alpha_n \end{bmatrix}^T$. Let $V_\xi = \xi^T \bar{P} \xi,$ where $\bar{P} = \bar{P}^T > 0$ is a solution of the Lyapunov equation $\bar{P}(A - HC) + (A - HC)^T \bar{P} = -I$.

Lemma 1. [8] The fault f in (3) can be reconstructed by a RBF neural network operating on the estimated states over a compact set χ .

From the above lemma, we have that

$$f(e + \mathcal{Y}_r) = \theta^{*T}\Omega(\hat{e} + \mathcal{Y}_r) + d \, . \tag{9}$$

where $\Omega(\cdot)$ are radius basis functions and d is approximation error. Let $\hat{\theta}$ be an estimate of the unknown optimal weights θ^* . Inputs of the network are estimated states $\hat{e} + \mathcal{Y}_r$. Output of the network is an estimate of the fault $\hat{f} = \hat{\theta}^T \Omega(\hat{e} + \mathcal{Y}_r)$. Let $\tilde{\theta} = \hat{\theta} - \theta^*$.

Lemma 2. [3] Over the compact set χ **,** there exists an unknown bound $\bar{\theta}^*$ such that

$$\|\eta(e + \mathcal{Y}_r, t) + d\| \le \bar{\theta}^* \, . \tag{10}$$

Let $\bar{\theta}$ be an estimate of the unknown bound $\bar{\theta}^*$. The learning rule will be given later.

Assumption 2. When a fault occurs, $e(T)$ belongs to a known compact set E_0 **.**

Define

$$c_2 = \max_{\theta^* \in \Phi_f, \hat{\theta} \in \bar{\Phi}_f} \frac{1}{2}\tilde{\theta}^T \Gamma^{-1} \tilde{\theta}, c_3 = \max_{\bar{\theta}^* \in \Phi_\delta, \bar{\theta} \in \bar{\Phi}_\delta} \frac{1}{2\gamma}(\bar{\theta} - \bar{\theta}^*)^2 . \tag{\textbf{11}}$$

where matrix $\Gamma > 0$ and constant $\gamma > 0$ are adaptation gains. Let λ be the minimal eigenvalue of Γ . Φ_f is a known compact set to which optimal weights belong, and $\bar{\Phi}_f$ is a known compact set to which the estimate of optimal weights are confined using projection algorithms [8]. The meanings of Φ_δ and $\bar{\Phi}_\delta$ are likewise.

Assumption 3. Over the domain of interest, there exists $\left|\hat{G}(\cdot)\right| \ge k_0 > 0$.

3 Main Results

Let $c_1 = \max_{e \in E_0} e^T Pe$. $E \triangleq \{e^T Pe \le c_4\}$. Choose the constant $c_4 > c_1 + c_2 + c_3$. Design a fault tolerant controller as follows.

$$u = -\hat{\theta}^T \Omega(\hat{e} + \mathcal{Y}_r) - \bar{\theta}\omega(\hat{e} + \mathcal{Y}_r) + G^{-1}(\hat{e} + \mathcal{Y}_r)(-K\hat{e} + y_r^{(n)} - \zeta(\hat{e} + \mathcal{Y}_r)) . \tag{12}$$

$$\dot{\hat{\theta}} = \Gamma\Omega(\hat{e} + \mathcal{Y}_r)G(\hat{e} + \mathcal{Y}_r)p(\hat{e}) . \tag{13}$$

$$\dot{\bar{\theta}} = \gamma\omega(\hat{e} + \mathcal{Y}_r)G(\hat{e} + \mathcal{Y}_r)p(\hat{e}) . \tag{14}$$

where $p(\hat{e}) \triangleq 2\hat{e}^T Pb$ and $\omega(\hat{e} + \mathcal{Y}_r) \triangleq \tanh(p(\hat{e})G(\hat{e} + \mathcal{Y}_r)/\mu)$. $\mu > 0$ is a design parameter. In order to reduce overshoot [8], the controller is made saturated outside the domain of interest.

Theorem 1. Let assumptions 1-3 hold, then there exist constants $\varepsilon > 0$ and $\mu > 0$ such that all signals of the closed-loop system (1) under the control law (12)-(14) are bounded in the presence of a fault.

Proof: Because the proof is similar to Seshagiri's [8], we do it briefly. Let

$$V = e^T Pe + \frac{1}{2}\tilde{\theta}^T \Gamma^{-1}\tilde{\theta} + \frac{1}{2\gamma}(\bar{\theta} - \bar{\theta}^*)^2 , \quad R = \{V \le c_4\} \cap \bar{\Phi}_f \cap \bar{\Phi}_\delta \times \{V_\xi \le c_5\varepsilon^2\} .$$

We aim to show that there exists a constant c_5 such that R is *positively invariant*[2].

First, using the boundedness of continuous functions on compact sets and the saturated control, we can show that there exists a constant c_5 such that $\dot{V}_\xi \le 0$ on the boundary $\{V \le c_4\} \cap \bar{\Phi}_f \cap \bar{\Phi}_\delta \times \{V_\xi = c_5\varepsilon^2\}$.

Secondly, within R , we can show that the derivative of V satisfies

$$\dot{V} \le -kV + k(c_2 + c_3) + k_\varepsilon\varepsilon + k_\mu\mu\bar{\theta}^* . \tag{15}$$

k_ε and k_μ are constants. If $\varepsilon \le \varepsilon^* = \dfrac{k(c_4 - c_3 - c_2)}{2k_\varepsilon}$ and $\mu \le \mu^* = \dfrac{k(c_4 - c_3 - c_2)}{2k_\mu\bar{\theta}^*}$, then $\dot{V} \le 0$ on the boundary $\{V = c_4\} \cap \bar{\Phi}_f \cap \bar{\Phi}_\delta \times \{V_\xi \le c_5\varepsilon^2\}$.

Therefore, the set R is positively invariant. Under assumption 2, we show that all signals of the closed-loop system are bounded. □

Remark 1. Using a procedure as that in [8], we can show that the trajectory of the closed-loop system enters the set R in very short time after a fault occurs. Hence, all the states are bounded and (15) is satisfied almost for all $t > T$. The approximate tracking error e after a fault occurs is of the order $O(\varepsilon + \mu + 1/\lambda + 1/\gamma)$.

[2] The definition of positive invariant set is given in the appendix.

4 Simulation Results

Consider a jet engine as that in [3].

$$\begin{aligned}\dot{z}_1 &= -z_2 - \frac{3}{2}z_1^2 - \frac{1}{2}z_1^3 \\ \dot{z}_2 &= -[u + \eta_0(t) + \beta(t-T)f_0(z)], z_1(0) = 0.1, z_2(0) = 0.5 \\ y &= z_1\end{aligned} \quad . \tag{16}$$

The unknown disturbance was set to $\eta_0(t) = 0.05\sin(11t) + 0.1\sin(37t)$, and the unknown fault function was chosen as $f_0(z) = 0.7\cos(z_2)$ which are similar to the example in [3]. The fault occurs at *t*=20*s*. A reference signal is given as follows.

$$y_r = \begin{cases} 1 & t \le 10s \\ 2 & t > 10s \end{cases}$$

The above system can be rewritten as

$$\begin{aligned}\dot{x}_1 &= x_2 \\ \dot{x}_2 &= -(3x_1 + \frac{3}{2}x_1^2)x_2 + u + \eta + \beta(t-T)f \\ y &= x_1\end{aligned} \tag{17}$$

through a nonlinear diffeomorphism $x_1 = z_1, x_2 = -z_2 - \frac{3}{2}z_1^2 - \frac{1}{2}z_1^3$.

Parameters of the high gain observer are $\varepsilon = 0.01$ and $H = [2\ \ 1]^T$. The initial condition is $\hat{e}(0) = 0$. For on-line approximation, we use 121 Gaussian basis functions that are uniformly centered in the region $[-1,1] \times [-1,1]$ with variance $\sigma^2 = 1$. Initial weights are zeros. The adaptive gain is $\Gamma = diag(2,2)$.

The no fault case can be considered as a special case of the faulty case if we let the fault equal zero. Hence, a same controller (12)-(14) is used whether a fault occurs or not. The parameters are set as follows. $K = [1\ \ 2]$, $\overline{\theta}(0) = 0$, P=$\begin{bmatrix} 1.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix}$, $\mu = 0.01$ and $\gamma = 1$. The controller saturation value is 10.

Fig.1 shows the trajectories of $y(t)$ and the controller input $u(t)$. From Fig.1, we see that the closed-loop system can track the reference signal even if a fault occurs.

5 Conclusions

The proposed method in this paper partly solved the problem of output feedback based fault tolerant control of a class of nonlinear systems, in the sense that only SISO systems are considered. Output based fault tolerant control of MIMO systems deserves further research.

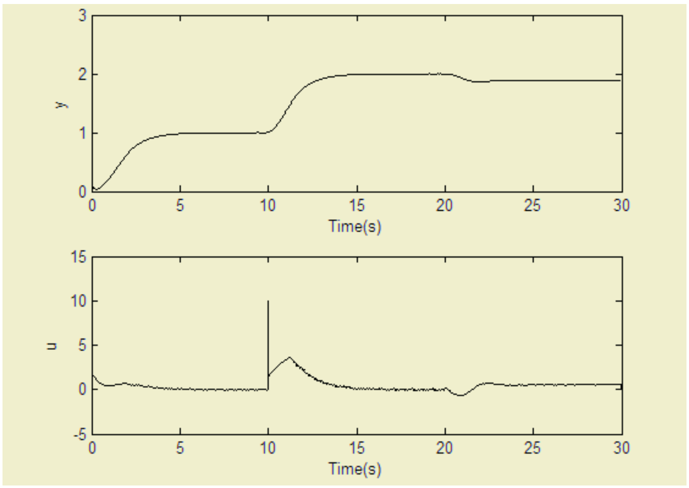


Fig. 1. The trajectories of the output $y(t)$ and the control input $u(t)$.

Acknowledgements

This work was supported by the NSFC and the national 973 program (Grant No. 2002CB312200) of China.

References

1. Diao, Y.X., Passino, M. K.: Intelligent Fault-Tolerant Control Using Adaptive and Learning Methods. Control Eng. Practice, **10** (2002) 801-817
2. Yang, G.H., Wang, J.L., Soh, C.B.: Reliable Nonlinear Control System Design by Using Strictly Redundant Control Elements. Int. J. Control, **69** (1998) 315-328
3. Polycarpou, M.M.: Fault Accommodation of a Class of Multivariable Nonlinear Dynamical Systems Using a Learning Approach. IEEE Trans. Autom. Control, **46** (2001) 736-742
4. Zhang, X.D., Parisini, T., Polycarpou, M.M.: Adaptive Fault-Tolerant Control of Nonlinear Uncertain Systems: An Information-Based Diagnostic Approach. IEEE Trans. on Autom. Control, **49** (2004) 1259-1274
5. Wang, H., Wang, Y.: Neural-Network-Based Fault-Tolerant Control of Unknown Nonlinear Systems. Control Theory and Applications, **146** (1999) 389-398
6. Marino, R., Tomei, P.: Observer-Based Adaptive Stabilization for a Class of Nonlinear Systems. Automatica, **28** (1992) 787-793
7. Kabore, P., Wang, H.: Design of Fault Diagnosis Filters and Fault-Tolerant Control for a Class of Nonlinear Systems. IEEE Trans. on Autom. Control, **46** (2001) 1805-1810
8. Seshagiri, S., Khalil, H.K.: Output Feedback Control of Nonlinear Systems Using RBF Neural Networks. IEEE Trans. Neural Netw, **11** (2000) 69-79
9. Isidori, A.: Nonlinear Control Systems: An Introduction. Lecture Notes in Control and Information Sciences, Springer-Verlag, Berlin Heidelberg New York, **72** (1985)
10. Khalil, H.K.: Nonlinear Systems. 3rd ed. Prentice Hall, Upper Saddle River New Jersey (2002)

Appendix: Definitions

Definition 1. (Relative degree) [10] Consider the following SISO system

$$\begin{aligned} \dot{x} &= f(x) + g(x)u \\ y &= h(x) \end{aligned} . \tag{18}$$

where $f(\cdot)$, $g(\cdot)$ and $h(\cdot)$ are sufficiently smooth continuous functions in a domain $D \subset R^n$. The nonlinear system (18) is said to have *relative degree* $\rho, 1 \le \rho \le n$ in a region $D_0 \subset D$ if

$$L_g L_f^{i-1} h(x) = 0, i = 1, 2, \cdots, \rho - 1; \quad L_g L_f^{\rho-1} h(x) \ne 0 . \tag{19}$$

for all $x \in D_0$.

The derivative in (19) is Lie derivative that is defined as follows.

$$L_f^0 h(x) = h(x), L_f^i h(x) = \frac{\partial L_f^{i-1} h}{\partial x} f(x), L_g L_f^i h(x) = \frac{\partial L_f^i h}{\partial x} g(x) .$$

Definition 2. (Positive invariant) [10] Consider the system $\dot{x} = f(x)$, a set M is said to be a positively invariant set if $x(0) \in M \Rightarrow x(t) \in M, \forall t > 0$.

Fault Tolerant Control of Nonlinear Processes with Adaptive Diagonal Recurrent Neural Network Model

Ding-Li Yu, Thoonkhin Chang, and Jin Wang

Control Systems Research Group, School of Engineering
Liverpool John Moores University, Liverpool L3 3AF, UK
d.yu@livjm.ac.uk

Abstract. Fault tolerant control (FTC) using an adaptive recurrent neural network model is developed in this paper. The model adaptation is achieved with the extended Kalman filter (EKF). A novel recursive algorithm is proposed to calculate the Jacobian matrix in the model adaptation so that the algorithm is simple and converges fast. A model inversion control with the developed adaptive model is applied to nonlinear processes and fault tolerant control is achieved. The developed control scheme is evaluated by a simulated continuous stirred tank reactor (CSTR) and effectiveness is demonstrated.

1 Introduction

Recurrent neural networks (RNN) have been used for modelling of nonlinear dynamic systems as this type of neural network (NN) includes dynamics in their structure. There are two kinds of RNNs, fully RNN and partially RNN. It has been reported that the training of fully RNN is difficult and very slow [1], while the partially RNN is easy to train and has smaller number of weights. Ku and Lee [2] developed a diagonal RNN (DRNN). The DRNN has considerably fewer weights than the fully RNN and the network is simplified considerably. The DRNN was trained by a generalized dynamic back-propagation algorithm which was still time consuming.

In this research we choose a DRNN to develop an adaptive model for dynamic systems, as this kind of network model is more accurate in model generalisation, especially for multi-step-ahead prediction, which is a key feature for model-based control. As the feedbacks with different time lags are included in the DRNN structure, a recursive calculation is developed in this paper, which greatly reduces the computing burden. The developed adaptive DRNN model is applied to a simulated CSTR process and is evaluated for set-point tracking control when the process is subject to different actuator or component faults.

2 DRNN Structure

The DRNN consists of one input layer, one hidden layer and one output layer. The configuration of the network is shown in Fig.1.

In Fig.1 $x\in\Re^n$, $h\in\Re^q$ and $\hat{y}\in\Re^p$ are the network input, hidden layer and output vectors respectively. $W^h\in\Re^{q\times(n+1)}$, and $W^y\in\Re^{p\times(q+1)}$ are feedforward weight matrices, while $W^d\in\Re^{v\times q}$ is the feedback weight matrix from hidden layer output to hidden layer input with different time delays. The network output is calculated by the following equations.

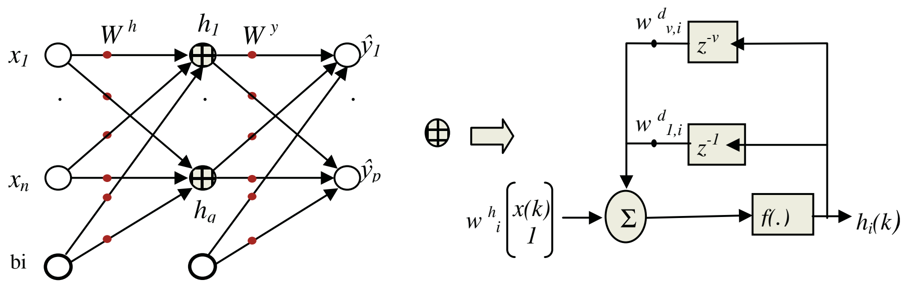


Fig. 1. DRNN structure

$$\hat{y}(k) = W^y \begin{bmatrix} h(k) \\ 1 \end{bmatrix} \tag{1}$$

$$h(k) = \frac{1}{1+e^{-z(k)}} \tag{2}$$

$$z(k) = W^h \begin{bmatrix} x(k) \\ 1 \end{bmatrix} + \begin{bmatrix} diag(w_1^d) & \cdots & diag(w_v^d) \end{bmatrix} \begin{bmatrix} h(k-1) \\ \vdots \\ h(k-v) \end{bmatrix} \tag{3}$$

In this paper, a recursive training algorithm is developed based on the EKF. The algorithm can be used to update the network model in on-line mode with the process input/output measurement.

3 Adaptive Training Algorithm

The nonlinear dynamic system to be modelled is represented by the NARX model of the following form,

$$y(k) = g[u(k-d-1),\cdots,u(k-d-n_u),y(k-1),\cdots,y(k-n_y)] + e(k) \tag{4}$$

where n_u and n_y are the input order and output order respectively, d represents the process transmission delay, $e(k)$ is a zero mean white noise vector, and $g(.)$ denotes a vector-valued nonlinear function. Consider the DRNN structure given in Fig.1, a model of system (4) can be realised with a DRNN,

$$\hat{y}(k) = f[x(k), W^h, W^d, W^y] \tag{5}$$

where

$$x(k) = \begin{bmatrix} u^T(k-d) & y^T(k-1) \end{bmatrix}^T \tag{6}$$

is the input vector of the network, $x(k) \in \Re^n$ with $n=m+p$ and v in (3) is

$$v = \max\{n_u \quad n_y\} - 1 \tag{7}$$

the time delay order within the hidden layer. Also, $u(k) \in \Re^m$ and $y(k) \in \Re^p$ are the sampled process input and output at sample instant k, $\hat{y}(k) \in \Re^p$ is the DRNN model output vector, nonlinear function $f(.)$ is the combination of the relationships presented by (1)-(3).

For on-line updating the weights of the DRNN model, the extended Kalman filter (EKF) algorithm is used to derive a recursive updating algorithm. The weight matrices to be updated are W^h, W^d and W^y and are formulated as a parameter vector $\theta \in \Re^{[q(n+1)+qv+p(q+1)]\times 1}$,

$$\theta = \begin{bmatrix} \theta^h \\ \theta^y \end{bmatrix} \tag{8}$$

with

$$\theta^h = \begin{bmatrix} \begin{bmatrix} w_1^h & w_{1,1}^d & \cdots & w_{v,1}^d \end{bmatrix}^T \\ \vdots \\ \begin{bmatrix} w_q^h & w_{1,q}^d & \cdots & w_{v,q}^d \end{bmatrix}^T \end{bmatrix}, \qquad \theta^y = \begin{bmatrix} (w_1^y)^T \\ \vdots \\ (w_p^y)^T \end{bmatrix} \tag{9}$$

where w_i^h is the ith row vector in W^h and w_i^y is the ith row vector in W^y. Then, the model parameter estimation with the EKF are formulated as the following equations,

$$\theta(k+1) = \theta(k) \tag{10}$$

$$y(k) = \hat{y}(k) + e_m(k) \tag{11}$$

where

$$\hat{y}(k) = f[x(k), \theta(k)] = W^y[(\frac{1}{1+e^{-W^h[x^T(k)\ 1]^T}})^T\ 1]^T \tag{12}$$

A standard EKF algorithm is applied to (10) to (12),

$$\hat{\theta}(k) = \hat{\theta}(k-1) + K_w(k)[y(k) - \hat{y}|_{k|k-1}] \tag{13}$$

$$E_w(k) = [I - K_w(k)C_w(k)]E_w(k-1) \tag{14}$$

$$K_w(k) = E_w(k-1)C_w(k)^T[R|_{k|k-1} + C_w(k)E_w(k-1)C_w(k)^T]^{-1} \tag{15}$$

where

$$\hat{y}|_{k|k-1} = \hat{g}[\hat{\theta}(k-1), x(k)] \tag{16}$$

$$C_w(k) = \frac{\partial \hat{y}(k)}{\partial \theta}\Big|_{\theta=\hat{\theta}(k-1)} \tag{17}$$

and where $R|_{k|k-1} \in \Re^{p\times p}$ is an unknown priori error covariance matrix that can be calculated recursively. $C_w(k) \in \Re^{p\times[q(n+1)+p(q+1)]}$ in (17) is the Jacobian matrix of model output $\hat{y}(k)$ with respect to θ, and is expressed as

$$C_w(k) = \frac{\partial \hat{y}(k)}{\partial \theta}\Big|_{\theta=\hat{\theta}(k-1)} = \begin{bmatrix} \frac{\partial \hat{y}(k)}{\partial \theta^h}\Big|_{\theta=\hat{\theta}(k-1)} & \frac{\partial \hat{y}(k)}{\partial \theta^y}\Big|_{\theta=\hat{\theta}(k-1)} \end{bmatrix} \tag{18}$$

Calculations of the two derivatives in (18) can be derived according to the DRNN structure and equation (10)-(12), and a recursive algorithm is obtained. For details see [3].

4 Model Predictive Control of a CSTR

4.1 Model Predictive Control Scheme

The model predictive control based on the developed adaptive DRNN model is illustrated by Fig. 2.

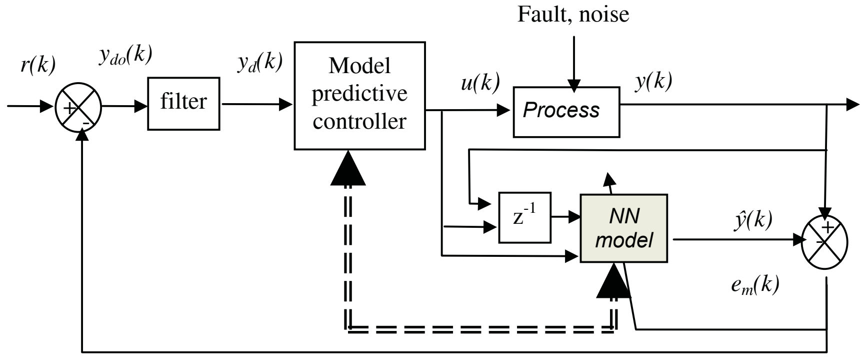


Fig. 2. The schematic of the MPC

The objective function used in the optimisation is

$$J = \sum_{i=1}^{p} (M_i - \hat{Y}_i)^T W_y^{(i)} (M_i - \hat{Y}_i) + \sum_{j=1}^{m} \xi(j) \Delta U_j^T W_u^{(j)} \Delta U_j \tag{19}$$

where

$$M_i^T = [m_i(t+N_1(i)), \cdots, m_i(t+N_2(i))]$$

$$\hat{Y}_i^T = [\hat{y}_i(t+N_1(i)), \cdots, \hat{y}_i(t+N_2(i))]$$

$$\Delta U_j^T = [u_j(t) - u_j(t-1), \cdots, u_j(t+N_u(j)) - u_j(t+N_u(j)-1)]$$

N_1, N_2 are vectors specifying the prediction horizon, with the *i*th element specifying the parameter for the corresponding output, N_u is the control horizon vector, with the *j*th element specifying the parameter for the corresponding input. The second summation term in the cost function (19) is designed to smooth the control signal. $W^{(i)}_y$ and $W^{(j)}_u$ are the weighting matrices for the ith output tracking error and jth input increment, respectively. ξ is the control weighting vector, with the *j*th element applied to the *j*th input. M is the modified set-point to introduce a feedback in order to compensate the system steady-state error and against disturbance effects. The filtered model prediction error in Fig.2 is also used to compensate the model outputs in the MPC scheme to correct the future predictions. This is to modify the model outputs in (5) by adding the corresponding filtered error to the right-hand side of the equations.

4.2 The CSTR Process

The CSTR process used in this research is a typical nonlinear dynamic process used in chemical and biochemical industry. A second order endothermic chemical reaction $2A \rightarrow B$ takes place in the tank. The schematic diagram of the process is shown in Fig. 3.

The process operations can be described as follows. The reactant A with constant concentration c_i and temperature $T_i(t)$ flows into the tank with the flow rate $q_i(t)$. An

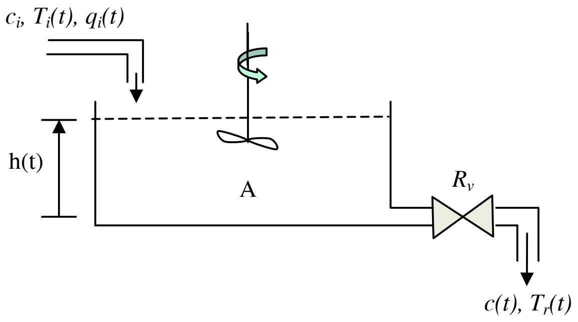


Fig. 3. CSTR process

endothermic chemical reaction takes place in the tank, which is based on the temperature, and the reaction absorbs heat energy. The two inputs and two outputs are chosen as follows, $u = [q_i \quad T_i]^T$, $y = [c \quad T_r]^T$. The following equations can be derived to describe the process dynamics.

$$\frac{dc(t)}{dt} = \frac{1}{Ah}\left\{c_i q_i(t) - c(t)\frac{h}{R_v} - 2Ahk_o c^2(t)\exp\left[-\frac{E_A}{RT_r(t)}\right]\right\} \tag{20}$$

$$\frac{dT_r(t)}{dt} = \frac{1}{\rho_r \sigma_r Ah}\{\rho_r \sigma_r [q_i(t)T_i(t) - \frac{h}{R_v}T_r(t)] -$$
$$-\Delta HAhk_o c^2(t)\exp[-\frac{E_A}{RT_r(t)}] + U_1[T_h - T_r(t)] - U_2[T_r(t) - T_x]\} \tag{21}$$

The operating ranges of the input variables applied by the actuators are

$$q_i(t) \subset [2, 5]\ (l/\sec),\ T_i(t) \subset [273, 480]\ (Kelvin) \tag{22}$$

The CSTR process is simulated as an unknown non-linear dynamic process in the research to evaluate the performance of the proposed adaptive DRNN model based MPC.

4.3 Simulation Results

In the simulation, *v=2*, *d=0* and 10 hidden layer nodes are found to be most appropriate for the DRNN modelling structure to represent the CSTR. In the DRNN recursive training simulation, the initial value of *θ(0)* vector is assigned to a small random value in the range of [-0.1, 0.1]. The initial value of the parameters of EKF, $E_w(0)$ and $R|_{k|k-1}$ are assigned to be an identity matrix and a zero matrix respectively. All the initial values of the Jacobian matrix in the learning calculation are assigned to be zero. The control parameters chosen are $N_1 = [1 \quad 1]^T$, $N_2 = N_1 + [6 \quad 6]^T$, $N_u = [1 \quad 1]^T$, $\xi = [0.1 \quad 0.1]^T$. The fault simulated to occurred to the process is an abrupt 40% change of q_i actuator fault,

$$q_i^*(k) = q_i(k) + \Delta q_i(k) = q_i(k) + 0.4q_i(k),\ \ \mathrm{k} > 250 \tag{23}$$

The tracking performance is displayed in Fig.4. It can be clearly seen that the tracking performance is recovered and stability is maintained after fault occurred at k = 250. This is due to that adaptive model learned the post-fault dynamics. Different faults have been tested, for which the detail is given in [3].

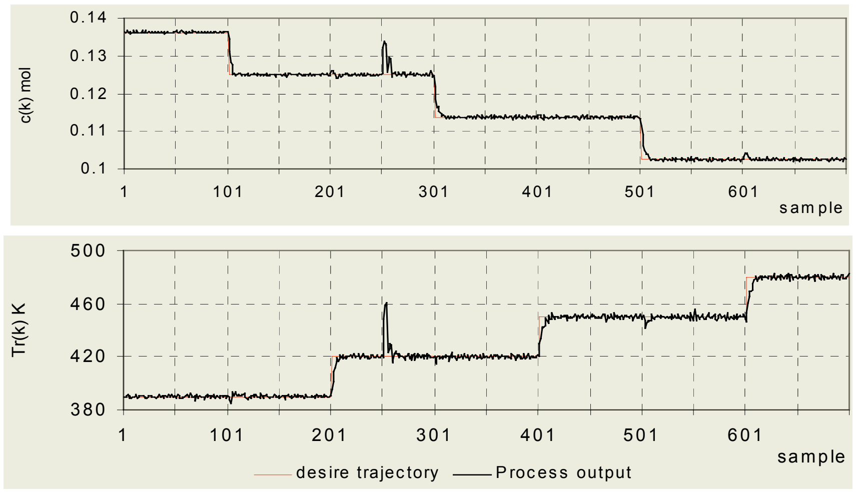


Fig. 4. CSTR tracking response with abrupt Δq_i fault

5 Conclusions

The direct recurrent neural network is made adaptive with an EKF-based on-line learning algorithm and is used in MPC scheme for fault tolerant control. Simulation results shown that the developed method is effective to learn the post-fault dynamics so that to compensate the degraded performance caused by faults.

References

1. Almeida, L.B: Backpropagation in Non-Feedforward Networks in Neural Computing Architectures. Edited by Aleksander, Cambridge, MA: MIT Press (1989) 75-91
2. Ku, C.C. , Lee, K.Y: Diagonal Recurrent Neural Networks for Dynamic System Control. IEEE Trans. on Neural Networks, **6** (1995) 144-156
3. Chang, T.K.: Fault Tolerant Control for Nonlinear Processes Using Adaptive Neural Networks. PhD Thesis, Scholl of Engineering, Liverpool John Moores University, U.K. (2002)

Dealing with Fault Dynamics in Nonlinear Systems via Double Neural Network Units*

Yong D. Song[1], Xiao H. Liao[1], Cortney Bolden[1], and Zhi Yang[2]

[1] Department of Electrical and Computer Engineering
North Carolina A&T State University
Greensboro, NC 27411, USA
{songyd,xliao}@ncat.edu

[2] Automation Institute Chongqing University, Chongqing 400044, China

Abstract. Most fault detection and accommodation methods have traditionally been derived based on linear system modeling techniques, which restrict the type of practical failure situation that can be modeled. In this paper we explore a methodology for fault accommodation in nonlinear dynamic systems. A new control scheme is derived by incorporating two neural network (NN) units to effectively attenuate and compensate uncertain dynamics due to unpredictable faults. It is shown that the method is independent of the nature of the fault. Numerical simulations are included to demonstrate the effectiveness of the proposed method.

1 Introduction

Due to growing demand on reliability of complex systems (such as automotive, manufacturing, robots and flight vehicles, etc), fault detection and accommodation (FDA) has become an important research area, attracting considerable attention over the past few years. Several methods have been developed with different applications. For example Yang and Saif introduced an adaptive observer for fault detection and isolation [1]. In [2], Wu, Grimble and Wei presented a QFT-based approach to this problem for its application in flight control. Visinsky, Cavallaro and Walker proposed a layered fault tolerance framework for remote robots in [3]. The learning approach is one of the methodologies for detecting, identifying and accommodating fault in nonlinear systems. The idea in this approach is to approximate any faulty behavior of the system by using an online approximation method i.e. Neural Networks, in which online approximator was presented to estimate an unknown nonlinear fault function, which can be used to accommodate the undesired fault in the system dynamics. Trunov and Polycarpou in [4] used a learning methodology in fault diagnosis in nonlinear multivariable systems. Vemuri and Polycarpou used neural network as a learning methodology for fault diagnosis in robotic systems in [5]. In the work by Borairi and Wang [6] and the work by Wang [7], neural network based method for fault detection and accommodation in nonlinear systems has also been used.

* This work was supported in part by NSF under the HBCU-RISE program (Award number HRD-0450203).

In this work we consider a class of nonlinear systems with faults in both system dynamics and actuation dynamics. A new fault-tolerant control scheme is derived by incorporating two neural network units to estimate and compensate faulty dynamics. Both theoretic analysis and numerical simulations show that the proposed method is effective in dealing with significant (ramp and jump) faulty dynamics.

2 Problem Formulation

The nonlinear dynamic system under consideration is described as follows,

$$\dot{x} = f_0(x) + g_0(x,u) + G(x,u)\beta_x(t - T_x) + \xi(x,u)\,, \tag{1}$$

where $x \in R^n$ denotes the state and $u \in R^n$ represents the virtual input, $f_0 \in R^n$ and $g_0 \in R^n$ are known dynamic functions of the system, $G \in R^n$ models the unknown dynamics due to the fault occurring at the instant $t \ge T_x$, $\xi \in R^n$ denotes model uncertainties of the system, the function $\beta_x(t - T_x)$ is defined as:

$$\beta_x(t - T_x) = \begin{cases} 0 & t \le T_x \\ 1 - e^{-\alpha_x(t - T_x)} & t \ge T_x \end{cases}. \tag{2}$$

Note that $\beta_x(t - T_x)$ becomes a step function as $\alpha_x \to \infty$, therefore this function is able to describe both ramp and jump faults. In this study, we also consider the actuator dynamics with possible actuation faulty, as governed by

$$\dot{u} = \psi(u) + bv + \gamma(u,v)\beta_u(t - T_u)\,, \tag{3}$$

where $\psi \in R^n$ and $b \in R^{n\times n}$ denote the known portion of the actuation dynamics with $\gamma\beta_u(t - T_u)$ being the fault actuation dynamics, where $\beta_u(t - T_u)$ is a function similar to $\beta_x(t - T_x)$ as defined before.

The objective is to develop a control scheme v such that the actual system states and the desired system states evolve with time as closely as possible in the presence of unexpected faults. Here x^* denotes the desired trajectory, i.e.:

$$\|x - x^*\| \le \varepsilon < \infty\,. \tag{4}$$

3 Fault-Tolerant Control Design

Let the error between actual trajectory x and desired x^* be $e = x - x^*$. The error dynamics then can be expressed as

$$\dot{e} = f_0 + g_0 - \dot{x}^* + f(x,u)\,, \tag{5}$$

where

$$f(x,u) = G\beta + \xi\,.$$

In practice very limited information about the faulty dynamics would be available. For this reason, we use Neural Networks (NN) to online recover its effects. By constructing a neural network properly, we can approximate any nonlinear function with bounded approximation error in that

$$f(x,u) = f_{NN}^I + \varepsilon_1 , \tag{6}$$

with

$$f_{NN}^I = a_1^T \varphi_1(x) = \sum_{i=1}^{n_1} a_{1i}^T \varphi_{1i}(x) , \tag{7}$$

where $a_{1i} \in R^{n_1 \times n}$ represent the ideal NN weights, $\varphi_{1i} \in R^{n_1}$ is the basic function (n_1 is the number of the neurons), and ε models the reconstruction error which is bounded by an unknown constant, i.e. $\|\varepsilon_1\| < \rho_1 < \infty$. Note that f_{NN}^I , in equation (7) represents an ideal (optimum) neural network – a network with optimum weights. A practical and implementable network should have estimate weights, i.e.,

$$\hat{f}_{NN}^I = \hat{a}_1^T \varphi_1(x) . \tag{8}$$

Now we can rewrite the error dynamic equation (5) as follows:

$$\dot{e} = f_0 + g_0 + a_1^T \varphi_1(x) + \varepsilon_1 - \dot{x}^* . \tag{9}$$

By introducing the weight estimation error $\tilde{a} = a - \hat{a}$ and two new variables z_N and v_p , the error dynamic equation (9) can be expressed as

$$\dot{e} = -k_0 e + \tilde{a}_1^T \varphi_1(x) + \varepsilon_1 + z_N + v_p , \tag{10}$$

where

$$z_N = k_0 e + f_0 + g_0 + \hat{a}_1^T \varphi_1(x) - \dot{x}^* - v_p . \tag{11}$$

It follows that

$$\dot{z}_N = A_0 + B_0 v + F - \dot{v}_p , \tag{12}$$

where

$$M_0 = \hat{a}_1^T \frac{\partial \varphi_1}{\partial x} + \frac{\partial g_0}{\partial x} + \frac{\partial f_0}{\partial x} + k_0 I, \qquad A_0 = \dot{\hat{a}}_1^T \varphi_1 + M_0(f_0 + g_0) - k_0 \dot{x}^* - \ddot{x}^*$$
$$B_0 = \frac{\partial g_0}{\partial u} b, \qquad F = M_0(G\beta_x + \xi) + \frac{\partial g_0}{\partial u}(\psi + \gamma\beta_u) .$$

To compensate the lumped uncertain function F , we introduce the second neural network f_{NN}^{II} such that

$$F = f_{NN}^{II} + \varepsilon_2 = a_2^T \varphi_2(x) + \varepsilon_2 , \tag{13}$$

with bounded reconstruction error, i.e., $\|\varepsilon_2\| < \rho_2 < \infty$. Now we are in the position to propose the control scheme

$$v = B_0^{-1}(-A_0 - k_N z_N - \hat{a}_2^T \varphi_2 + v_c + \dot{v}_p - e) , \tag{14}$$

where v_c is a compensating signal to be specified later. The above control scheme leads to the following close-loop dynamics

$$\dot{z}_N = -k_N z_N + \tilde{a}_2^T \varphi_2 + v_c - e + \varepsilon_2 . \tag{15}$$

To show the stability of the closed-loop system we consider following Lyapunov function candidate:

$$V=\frac{1}{2}[e^T e+z_N^T z_N+tr(\tilde{a}_1^T\tilde{a}_1)+tr(\tilde{a}_2^T\tilde{a}_2)+(\rho_1-\hat{\rho}_1)^2+(\rho_2-\hat{\rho}_2)^2]. \tag{16}$$

it can be shown that

$$\dot{V}=-k_0e^T e-k_N z_N^T z_N+tr[\tilde{a}_1^T(\varphi_1 e^T-\dot{\hat{a}}_1)]+tr[\tilde{a}_2^T(\varphi_2 z_N^T-\dot{\hat{a}}_2)]$$
$$+e^T(v_p+\varepsilon_1)+z_N^T(v_c+\varepsilon_2)-(\rho_1-\hat{\rho}_1)\dot{\hat{\rho}}_1-(\rho_2-\hat{\rho}_2)\dot{\hat{\rho}}_2$$

By introducing the algorithms

$$\begin{aligned} &\dot{\hat{a}}_1=\varphi_1 e^T, && \dot{\hat{a}}_2=\varphi_2 z_N^T \\ &v_p=-\frac{\hat{\rho}_1 e}{\|e\|+\mu_1}, && v_c=-\frac{\hat{\rho}_2 z_N}{\|z_N\|+\mu_2}, \\ &\dot{\hat{\rho}}_1=\frac{\|e\|^2}{\|e\|+\mu_1}, && \dot{\hat{\rho}}_2=\frac{\|z_N\|^2}{\|z_N\|+\mu_2} \end{aligned} \tag{17}$$

where $\mu_{1,2}>0$ are chosen such that $\int_0^t \mu_{1,2}d\tau\le\eta_{1,2}<\infty$, and $\sigma=\mu_1\rho_1+\mu_2\rho_2$, we can show that

$$\dot{V}\le-k_0e^T e-k_N z_N^T z_N+\sigma. \tag{18}$$

Note that $\int_0^t \sigma d\tau=\int_0^t(\mu_1\rho_1+\mu_2\rho_2)d\tau\le\eta_1\rho_1+\eta_2\rho_2<\infty$. Therefore, from (18) it is readily deduced that $e\in L_\infty\cap L_2$, $z_N\in L_\infty\cap L_2$, $\tilde{a}_1\in L_\infty$, $\tilde{a}_2\in L_\infty$, $\hat{\rho}_1\in L_\infty$, $\hat{\rho}_2\in L_\infty$ thus, $v_c\in L_\infty$, $v_p\in L_\infty$. Moreover, from equation (10) and (15) we can show that $\dot{e}$ and $\dot{z}_N$ are bounded and therefore e and z_N are uniformly continuous. Hence, $\|e\|\to 0$ as $t\to\infty$ by Barbalat lemma.

4 Simulation Examples

To evaluate the effectiveness of the proposed approaches, we have conducted two simulation studies. In the first simulation example, the dynamics of the system is given by:

$$\begin{cases}\dot{x} &=\sin x+u+10\cos x\times\beta(t-10)+\sin(x+u)\\ \dot{u} &=-u+v+\sin u\times\beta(t-5)\end{cases}$$

and the desired state is given by: x*=sin t. We consider the fault model $\beta(t-10)$ as depicted in equation (2), the real state (blue line) and the desired state (green line) are plotted in Figure 1. It is seen that the control scheme is able to accommodate the failure effectively.

In the second simulation example, the following dynamics of the system is tested:

$$\begin{cases}\dot{x} &=\sin x+u+5\cos^2 x\times\beta(t-20)+\sin(x+u)\\ \dot{u} &=-u+v+3e^{-u^2}\times\beta(t-10)\end{cases}$$

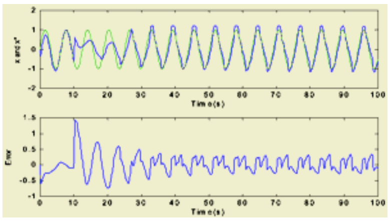


Fig. 1. Tracking Trajectory

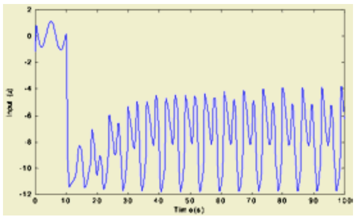


Fig. 2. Control Input

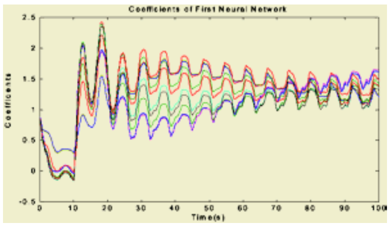


Fig. 3. Estimated Weights of First NN

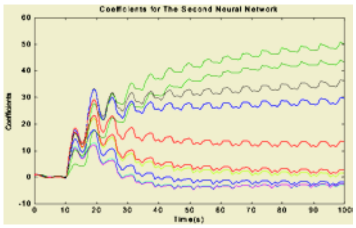


Fig. 4. Estimated Weights of Second NN

The fault model $\beta(t-20)$ as described as equation (2) is simulated. The desired trajectory is given as $x* = e^{-t}$. The actual system state (blue line) and the desired state (green line) are depicted in Figure 5. It can be seen that the error signal suddenly increases after the occurrence of the fault. Additionally, it can be observed that the actual system state follows to the desired trajectory in very short period of time even an unexpected fault occurs.

5 Conclusion

In this paper, neuro-adaptive approach is introduced for fault estimation and accommodation in nonlinear dynamics systems. The proposed approach is independent of the nature of the fault. Two NN units are incorporated to effectively attenuate the fault dynamics occurring in both systems and actuators. As verified via computer simulation, the proposed control method is able to deal with both ramp and jump faults effectively. Application of the method to practical systems (i.e., UAVs, UGVs) is underway.

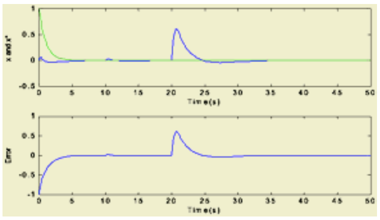


Fig. 5. Tracking Trajectory

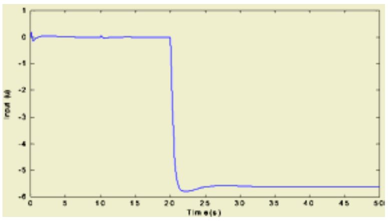


Fig. 6. Control Input

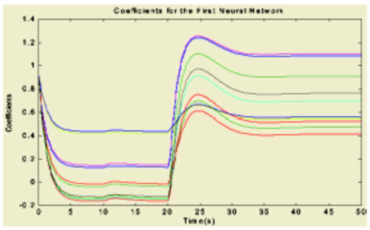


Fig. 7. Estimated Weights of First NN

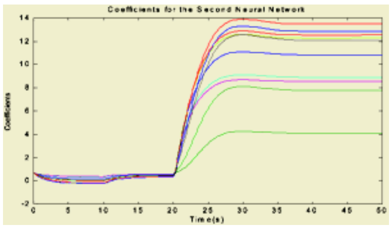


Fig. 8. Estimated Weights of Second NN

References

1. Yang, H., Saif, M.: Fault Detection and Isolation for a Class of Nonlinear Systems Using an Adaptive Observer. Proc. Amer. Contr. Conf., (1997) 463-467
2. Wu, S., Grimble, M. J., Wei, W.: QFT-Based Robust/Fault-Tolerant Flight Control Design for a Remote Pilot-less Vehicle. IEEE Trans. Control System Technology, **8** (2000) 1010-1016
3. Visinsky, M. L., Cavallaro, J. R., Walker, I. D.: A Dynamic Fault Tolerance Framework for Remote Robots. IEEE Trans. Robotics and Automation, **11** (1995) 477-490
4. Trunov, A. B., Polycarpou, M. M.: Automated Fault Diagnosis in Nonlinear Multivariable Systems Using a Learning Methodology. IEEE Trans. Neural Networks, 11 (2000) 91-101
5. Vemuri, A. T., Polycarpou M. M.: Neural Network-based Robust Fault Diagnosis in Robotic Systems. IEEE Trans. Neural Networks, 8 (1997) 1410-1420
6. Borairi, M., Wang, H.: Actuator and Sensor Fault Diagnosis of Nonlinear Dynamic Systems via Genetic Neural Networks and Adaptive Parameter Estimation Technique. IEEE Intl. Conf. on Contr. App (1998)
7. Wang, H.: Fault Detection and Diagnosis for Unknown Nonlinear Systems: a Generalized Framework via Neural Network. IEEE Intl. Conf. On Intelligent Processing Sys., (1997) 1506-1510

Neural Adaptive Singularity-Free Control by Backstepping for Uncertain Nonlinear Systems

Zhandong Yu and Qingchao Wang

Astronautics School, Harbin Institute of Technology,
PO Box 355, Harbin, Heilongjiang 150001, China
zhandong_yu@ds.hit.edu.cn

Abstract. An adaptive neural control approach based on backstepping is presented. Unmodelled dynamics or external disturbances are considered in the nonlinear models. Approximating nonlinear dynamic is one of the performances of multi-layer feedforward neural networks. By the Lyapunov's stability theory, the NN weights are turned on-line with no prior training needed. An important feature of the presented approach is that by modifying a novel quasi-weighted Lyapunov function, the possible control singularity in the design of NN adaptive controller is avoided effectively. Finally, the approach is applied in CSTR system. The simulation result is given to demonstrate the feasibility and effectiveness of the proposed method.

1 Introduction

The subject of the adaptive control of nonlinear systems with uncertainties has been attracting the attention of researchers for years. One of the long-standing obstacles in adaptive design of nonlinear systems, the so-called matching condition problem, was completely overcome with the adaptive backstepping design [1]. A lot of research about backstepping control approach for systems with linearly parameterized uncertainties has been reported in various publications [2], [3]. However, the most physical models cannot be linearly parameterized and the condition that nonlinear vector fields are known friendly is rigorous. Some researchers extended the adaptive backstepping technique to more systems by combining with the neural network approximation [4], [5], [6].

Theoretically, as long as a sufficient number of neurons are employed, neural network can approximate any continuous function to an arbitrary accuracy on any compact set [7]. Neural network technique has been found to be particularly useful for controlling nonlinear systems with uncertainties. The major neural adaptive approach is based on the Lyapunov's stability theory, which gives an adaptive control law that guarantees the stability of systems [8], [9]. It is worth noting that control singularity or problem losing controllability is usually met in adaptive feedback controller. In order to solve it, Ge et al (1999) and Zhang et al (2000) designed the integral Lyapunov function.

In this paper, we develop an adaptive backstepping controller combined with NN technique for the nonlinear systems not transformable into the linear parametric forms. By utilizing the approximation property of feedforward neural network, all nonlinear functions are allowed to be almost totally unknown. In order to avoid the singularity, a quasi-weighted Lyapunov function for backstepping procedure is modified. It can be proved that the presented control law guarantees global uniform ultimate boundedness in the presence of unmodelled dynamic or unknown external disturbance.

2 Preliminaries

2.1 Problem Formulation and Assumption

The uncertain nonlinear systems are described by the following equations [4]:

$$\begin{aligned} &\dot{x}_i = f_i(\chi_i) + g_i(\chi_i)x_{i+1} + \Delta_i(\chi_n, t) \quad 1 \le i \le n-1 \\ &\dot{x}_n = f_n(\chi_n) + g_n(\chi_n)u + \Delta_n(\chi_n, t) \\ &y = x_1 \end{aligned} \tag{1}$$

where, $\chi_i=[x_1, x_2,\ldots, x_i]^T \in R^i$ (i=1,…,n); $\chi=\chi_n\in R^n$, u, $y\in R$ are the state vector, system input and output, respectively; $f_i(\chi_i)$, $g_i(\chi_i)$, i=1,…,n are unknown smooth functions, and they cannot be linearly parameterized; Δ_i are uncertain continuous functions representing the unmodelled dynamics or uncertain disturbances. The main difficulty of control problem comes from uncertainties gains $g_i(\chi_i)$. The singularity problem is not avoided due to transient behavior of the estimates of gains ($\hat{g}_i\rightarrow 0$). In order to overcome it, a novel quasi-weighted Lyapunov function is modified, and the uncertain nonlinear dynamic is estimated by NN algorithm, rather than $g_i(\chi_i)$ are approximated directly. In the paper, the following assumptions are made:

Assumption 1. The signs of $g_i(\chi_i)$ are known. These exist constants $g_{i0}>0$ and known smooth functions $G_i(\chi_i)$, which have the same signs as $g_i(\chi_i)$, such that $g_{i0}\le|g_i(\chi_i)|\le |G_i(\chi_i)|$, and the derivatives of $\beta_i(\chi_i)=|G_i(\chi_i)/g_i(\chi_i)|$ exist and are bounded.

Assumption 2. There exists an unknown positive constant λ_i and a known smooth and globally bounded function $\mu_i(\chi_i)$ so that $|\Delta_i|\le\lambda_i\,|\mu_i|$, ($i$=1,…,$n$).

Assumption 3. The desired trajectory $y_r^{(i)}\in R$, ($0\le i\le n$) is piecewise continuous bounded real-valued function. And $y_r\in \Omega_{di}$ with Ω_{di} known compact sets.

Remark 1. The Assumption 1 implies that the smooth functions $g_i(\chi_i)$ are strictly either positive or negative. Without losing generality, let $g_i(\chi_i)>0$, $\beta_{imax}(\chi_i)=|G_i(\chi_i)/g_{i0}|$ and $\beta'_i(\chi_i)\le 2\kappa_i$, $\kappa_i\ge 0$. The assumption is reasonable because the system is controllable and $g_i(\chi_i)$ are away from zeros. For most physical plant, the bounds of $g_i(\chi_i)$ are not difficult to be determined.

2.2 Neural Network for Approximation

In this paper, the three-layer feedforward neural network is used to approximate a smooth function as follow: $f(\chi)=W^T\Psi(V^TZ)+\rho(\chi)$, where, $Z=[\chi^T,1]^T\in \Omega_z$ is the NN input vector; $W=[w_1, w_2, \ldots, w_l]^T\in R^l$ and $V=[v_1, v_2, \ldots, v_l]\in R^{(m+1)\times l}$ are the first-to-second layer and the second-to-third layer weights, respectively; the NN node number is l; $\rho(\chi)$ is the NN approximation error. The activation function of hidden layer is Sigmoid, as follows: $\Psi(x)=(1+exp(-\gamma x))^{-1}$, where $\gamma>0$. Then, $\Psi(V^{T}Z)=[\Psi(v_1^{T}Z), \Psi(v_2^{T}Z), \ldots, \Psi(v_l^{T}Z), 1]^{T}$. According to the universal approximation theorem [7], for a given smooth function $f(\chi)$, there exist idea constant weights W^{*}, V^{*} such that $|\rho(\chi)|\le\rho_N$ with any constant $\rho_N>0$ for all $Z\in\Omega_Z$. Unfortunately, W^{*} and V^{*} are unknown. Let $\hat{W}$ and $\hat{V}$ be the estimates of W^{*} and V^{*}, respectively. And the weight estimation errors are $\tilde{W}=\hat{W}-W^{*}$ and $\tilde{V}=\hat{V}-V^{*}$. There exists the Lemma 1:

Lemma 1. [11], the estimation error of NN can be expressed as:

$$\begin{aligned}
&\hat{W}^{T}\Psi(\hat{V}^{T}Z)-W^{*T}\Psi(V^{*T}Z)=\tilde{W}^{T}(\hat{\Psi}-\hat{\Psi}'\hat{V}^{T}Z)+\hat{W}^{T}\hat{\Psi}'\tilde{V}^{T}Z+d_u\\
&\hat{\Psi}=\Psi(\hat{V}^{T}Z),\ \hat{\Psi}'=diag\{\hat{\psi}_1'\ \ \hat{\psi}_2'\ \ \ldots\ \ \hat{\psi}_l'\}\\
&\hat{\psi}_i'=\psi'(\hat{v}_i^{T}Z)=d[\psi(z_a)]/dz_a\,|_{z_z=\hat{v}_i^{T}Z}\quad i=1,2,\ldots,l\\
&|d_u|\le\|V^{*}\|_F\|Z\hat{W}^{T}\hat{\Psi}'\|_F+\|W^{*}\|\|\hat{\Psi}'\hat{V}^{T}Z\|+\|W^{*}\|_1
\end{aligned}\tag{2}$$

where, $\|A\|_1$, $\|A\|$ and $\|A\|_F$ are 1-norm, Euclid norm and Frobenius norm, respectively.

Lemma 2. For the $z\in R$, the weight matrix and the input vector of NN, the follow inequality hold with $\varepsilon>0$:

$$\begin{aligned}
&\|V^{*}\|_F\|Z\hat{W}^{T}\hat{\Psi}'\|_F|z|\le\frac{z^2}{\varepsilon}\|Z\hat{W}^{T}\hat{\Psi}'\|_F^2+\frac{\varepsilon}{4}\|V^{*}\|_F^2\\
&(\|W^{*}\|_1+|\rho|)|z|\le\frac{z^2}{2\varepsilon}+\varepsilon(\|W^{*}\|_1^2+\rho^2)\\
&|W^{*}\|\|\hat{\Psi}'\hat{V}^{T}Z\||z|\le\frac{z^2}{\varepsilon}\|\hat{\Psi}'\hat{V}^{T}Z\|^2+\frac{\varepsilon}{4}\|W^{*}\|^2
\end{aligned}\tag{3}$$

The proof can be gain by completing squares inequality.

3 Neural Adaptive Singularity-Free Controller by Backstepping

In this section, the neural adaptive backstepping controller is design. The quasi-weighted Lyapunov function is modified to avoid the singularity of adaptive control. Define the control error as follow: $z_1=x_1-y_r$, $z_2=x_2-\alpha_1,\ldots$, $z_n=x_n-\alpha_{n-1}$, where, α_i is virtual control of backstepping.

Step 1: $\dot{z}_1=\dot{x}_1-\dot{y}_r=f_1(\chi_1)+g_1(\chi_1)(z_2+\alpha_1)+\Delta_1(\chi,t)-\dot{y}_r$. Consider the positive definite function $V_0=\beta_1(\chi_1)z_1^2/2$. $\beta_1(\chi_1)$ is the variable weight of Lyapunov function V_0. According to Assumption 1 and Remark 1, the derivative of V_0 is

$$\begin{aligned}
\dot{V}_0&=\beta_1z_1\dot{z}_1+\dot{\beta}_1z_1^2/2=z_1\beta_1[f_1(\chi_1)+g_1(\chi_1)(z_2+\alpha_1)+\Delta_1(\chi,t)-\dot{y}_r]+\dot{\beta}_1z_1^2/2\\
&\le z_1\beta_1[f_1(\chi_1)+g_1(\chi_1)(z_2+\alpha_1)+\Delta_1(\chi,t)-\dot{y}_r]+\kappa_1z_1^2
\end{aligned}\tag{4}$$

If f_1, g_1 and Δ_1 are known accurately, the virtual control is directly constructed by the expression (4). Due to the uncertainties of them, the NN is used to approximate these functions. Define: $F_1 = \beta_1 f(\chi_1) + \kappa_1 z_1 - \beta_1 \dot{y}_r = W_1^{*\mathrm{T}} \Psi(V_1^{*\mathrm{T}} Z_1) + \rho_1$, where, $Z_1 = [\chi_1^{\mathrm{T}}, y_r, y_r', 1]^{\mathrm{T}}$ as the NN input vector, such that the Lyapunov function is as follow:

$$V_1 = V_0 + \frac{1}{2}\left[\tilde{W}_1^{\mathrm{T}} \Gamma_{w1}^{-1} \tilde{W}_1\right] + \frac{1}{2} tr\left\{\tilde{V}_1^{\mathrm{T}} \Gamma_{v1}^{-1} \tilde{V}_1\right\} \tag{5}$$

If the virtual control law and adaptive law are chosen as

$$\alpha_1 = -\frac{1}{G_1(\chi_1)}\left[\hat{W}_1^{\mathrm{T}} \Psi(\hat{V}_1^{\mathrm{T}} Z_1) + c_1 \beta_{1\max} |\mu_1|^2 z_1 + k_1(t) z_1\right] \tag{6}$$

$$\begin{aligned} \dot{\hat{W}}_1 &= \Gamma_{w1}[(\hat{\Psi} - \hat{\Psi}' \hat{V}_1^{\mathrm{T}} Z_1) z_1 - \sigma_{w1} \hat{W}_1] \quad \sigma_{w1} > 0 \\ \dot{\hat{V}}_1 &= \Gamma_{v1}[Z_1 \hat{W}_1^{\mathrm{T}} \hat{\Psi}' z_1 - \sigma_{v1} \hat{V}_1] \quad \sigma_{v1} > 0 \end{aligned} \tag{7}$$

according to the Assumption 2, Lemma 1, and the inequalities: $2\tilde{W}_1^{\mathrm{T}} \hat{W}_1 \geq \| \tilde{W}_1 \|^2 - \| W_1^* \|^2$ and $2tr\{\tilde{V}_1^{\mathrm{T}} \hat{V}_1\} \geq \| \tilde{V}_1 \|_{\mathrm{F}}^2 - \| V_1^* \|_{\mathrm{F}}^2$, the derivative of V_1 is

$$\begin{aligned} \dot{V}_1 &\leq z_1 G_1(\chi_1) z_2 + z_1[W_1^{*\mathrm{T}} \Psi(V_1^{*\mathrm{T}} Z_1) + \rho_1 - \hat{W}_1^{\mathrm{T}} \Psi(\hat{V}_1^{\mathrm{T}} Z_1) + \beta_1 \Delta_1(\chi, t) - k_1(t) z_1 \\ &\quad - c_1 \beta_{1\max} |\mu_1|^2 z_1] + \tilde{W}_1^{\mathrm{T}} \Gamma_{w1}^{-1} \dot{\hat{W}}_1 + tr\{\tilde{V}_1^{\mathrm{T}} \Gamma_{v1}^{-1} \dot{\hat{V}}_1\} \\ &\leq z_1 G_1(\chi_1) z_2 - k_1(t) z_1^2 + [|\rho_1| + \| V_1^* \|_{\mathrm{F}} \| Z_1 \hat{W}_1^{\mathrm{T}} \hat{\Psi}' \|_{\mathrm{F}} + \| W_1^* \| \| \hat{\Psi}' \hat{V}_1^{\mathrm{T}} Z_1 \| \\ &\quad + | W_1^* |_1] |z_1| - \frac{1}{2} \sigma_{w1} (\| \tilde{W}_1 \|^2 - \| W_1^* \|^2) - \frac{1}{2} \sigma_{v1} (\| \tilde{V}_1 \|_{\mathrm{F}}^2 - \| V_1^* \|_{\mathrm{F}}^2) + \frac{\beta_1 \lambda_1^2}{4 c_1} \end{aligned} \tag{8}$$

Let $k_1(t) = \frac{1}{\varepsilon_1}[\| Z_1 \hat{W}_1^{\mathrm{T}} \hat{\Psi}' \|_{\mathrm{F}}^2 + \| \hat{\Psi}' \hat{V}_1^{\mathrm{T}} Z_1 \|^2 + G_1(x_1) + \frac{1}{2}]$, and Substitute $k_1(t)$ into (8),

$$\begin{aligned} \dot{V}_1 &\leq z_1 G_1(\chi_1) z_2 - \frac{G_1(\chi_1)}{\varepsilon_1} z_1^2 - \frac{1}{2} \sigma_{w1} \| \tilde{W}_1 \|^2 - \frac{1}{2} \sigma_{v1} \| \tilde{V}_1 \|_{\mathrm{F}}^2 + N_1 \\ &\leq -\frac{g_{10}}{\varepsilon_1} \beta_1 z_1^2 - \frac{1}{2} \sigma_{w1} \| \tilde{W}_1 \|^2 - \frac{1}{2} \sigma_{v1} \| \tilde{V}_1 \|_{\mathrm{F}}^2 + N_1 + z_1 G_1(\chi_1) z_2 \\ &\leq -\xi_1 V_1 + N_1 + z_1 G_1(\chi_1) z_2 \end{aligned} \tag{9}$$

where, $N_1 = \varepsilon_1[\frac{1}{4} \| V_1^* \|_{\mathrm{F}}^2 + \frac{1}{4} \| W_1^* \|^2 + \| W_1^* \|_1^2 + |\rho_1|^2] + \frac{1}{2} \sigma_{w1} \| W_1^* \|^2 + \frac{1}{2} \sigma_{v1} \| V_1^* \|_{\mathrm{F}}^2 + \frac{\beta_1 \lambda_1^2}{4 c_1}$; $\xi_1 = \min\left\{\frac{g_{10}}{\varepsilon_1}, \frac{\sigma_{w1}}{\lambda_{\max}(\Gamma_{w1}^{-1})}, \frac{\sigma_{v1}}{\lambda_{\max}(\Gamma_{v1}^{-1})}\right\}$; $\lambda_{\max}(\cdot)$ denotes the largest eigenvalue of the matrix. The design process at step i ($2 \leq i \leq n$-1) is same as step 1, and the expresses of $k_i(t)$ and N_i are chosen as $k_1(t)$ and N_1, expect the subscripts are changed as i.

Step n: Let the Lyapunov function of whole system as follow:

$$V_n = V_{n-1} + \frac{1}{2} \beta_n z_n^2 + \frac{1}{2}\left[\tilde{W}_n^{\mathrm{T}} \Gamma_{wn}^{-1} \tilde{W}_2\right] + \frac{1}{2} tr\left\{\tilde{V}_n^{\mathrm{T}} \Gamma_{vn}^{-1} \tilde{V}_n\right\} \tag{10}$$

Let $F_n = \beta_n f(\chi) + \kappa_n z_n - \beta_n \dot{\alpha}_{n-1} = W_n^{*\mathrm{T}} \Psi(V_n^{*\mathrm{T}} Z_n) + \rho_n$, where, $Z_n = [\chi_n^{\mathrm{T}}, \alpha_{n-1}, \dot{\alpha}_{n-1}, 1]^{\mathrm{T}}$ as NN input vector. The actual control law and adaptive law are

$$u = -\frac{1}{G_n(\chi)}\left[\hat{W}_n^{\mathrm{T}} \Psi(\hat{V}_n^{\mathrm{T}} Z_n) + c_n \beta_{n\max} |\mu_n|^2 z_n + k_n(t) z_n + G_{n-1}(\chi_{n-1}) z_{n-1}\right] \cdot \tag{11}$$

$$\begin{aligned} \dot{\hat{W}}_n &= \Gamma_{wn}[(\hat{\Psi} - \hat{\Psi}' \hat{V}_n^{\mathrm{T}} Z_n) z_n - \sigma_{wn} \hat{W}_n] \quad \sigma_{wn} > 0 \\ \dot{\hat{V}}_n &= \Gamma_{vn}[Z_n \hat{W}_n^{\mathrm{T}} \hat{\Psi}' z_n - \sigma_{vn} \hat{V}_n] \quad \sigma_{vn} > 0 \end{aligned} \tag{12}$$

The derivation of (10) is,

$$\dot{V}_n \le \dot{V}_{n-1} + z_n[F_n + G_n(\chi)u + \beta_n \Delta_n(\chi,t)] + \tilde{W}_n^{\mathrm{T}} \Gamma_{wn}^{-1} \dot{\hat{W}}_n + tr\left\{\tilde{V}_n^{\mathrm{T}} \Gamma_{vn}^{-1} \dot{\hat{V}}_n\right\} \le -\xi_n V_n + \sum_{j=1}^{n} N_j \tag{13}$$

where, $\xi_n = \min\left(\frac{g_{10}}{\varepsilon_1}, \ldots, \frac{g_{n0}}{\varepsilon_n}, \frac{\sigma_{w1}}{\lambda_{\max}(\Gamma_{w1}^{-1})}, \ldots, \frac{\sigma_{wn}}{\lambda_{\max}(\Gamma_{wn}^{-1})}, \frac{\sigma_{v1}}{\lambda_{\max}(\Gamma_{v1}^{-1})}, \ldots, \frac{\sigma_{vn}}{\lambda_{\max}(\Gamma_{vn}^{-1})}\right)$.

Remark 2. In [5],[6], the NN input vector is $Z_i = [\chi_i^{\mathrm{T}}, \alpha_{i-1}, \partial\alpha_{i-1}/\partial\chi_i, \omega_{i-1}, 1]^{\mathrm{T}}$ where,

$$\dot{\alpha}_{i-1} = \frac{\partial \alpha_{i-1}}{\partial \chi_{i-1}} \dot{\chi}_{i-1} + \omega_{i-1}, \quad \omega_{i-1} = \sum_{j=1}^{i-1} \frac{\partial \alpha_{i-1}}{\partial y_r^{(j-1)}} y_r^{(j)} + \sum_{j=1}^{i-1} [\frac{\partial \alpha_{i-1}}{\partial \hat{W}_j} \dot{\hat{W}}_j + \sum_{r=1}^{N} \{\frac{\partial \alpha_{i-1}}{\partial \hat{V}_{jr}} \dot{\hat{V}}_{jr}\}]$$

Because of the complexity of α_i, it is difficult to solute the partial derivation. In this paper, the NN input is $Z_i = [\chi_i^{\mathrm{T}}, \alpha_{i-1}, \dot{\alpha}_{i-1}, 1]^{\mathrm{T}}$, and the derivation of α_i can be expressed as the numerical form directly if the sampling time is short enough.

Remark 3. Our approach is quite similar to that developed in [5],[10] where the integral Lyapunov function is applied to recursive backstepping design. However, the quasi-weighted Lyapunov function as (10) for backstepping procedure is modified here.

4 Stability Analysis

Theorem 1 Consider the nonlinear system (1), satisfying the Assumptions 1 and 2, under the control law (11) and adaptive law (12). For the bounded initial conditions, all state variables of closed-loop system and the weight of NN are the global bounded, and the tracking error of system satisfies the expression as follow:

$$|y - y_r| \le \sqrt{2\left|V(0) - \frac{1}{\xi_n}\sum_{j=1}^{n} N_j\right| e^{-\xi_n t}} + \sqrt{\frac{2}{\xi_n}\sum_{j=1}^{n} N_j} \tag{14}$$

Proof. Let the Lyapunov function be $V(t)$ as (10). Using the inequality (13), yields

$$\frac{d}{dt}\left(V(t)e^{\xi_n t}\right) = \dot{V}e^{\xi_n t} + \xi_n V e^{\xi_n t} \le e^{\xi_n t}\sum_{j=1}^{n} N_j \tag{15}$$

Integrating the above inequality over [0, *t*] and multiplying $e^{-\xi_n t}$ at both sides of the inequality, leads to

$$V(t) \le \left(V(0) - \frac{1}{\xi_n}\sum_{j=1}^{n} N_j\right) e^{-\xi_n t} + \frac{1}{\xi_n}\sum_{j=1}^{n} N_j \tag{16}$$

If $t \to \infty$, the upper bound of $V(t)$ just depends on the second term of (16). Because of the boundedness of N_i, the state variables and NN weights are global bounded. Considering $V(t) \geq (y - y_r)^2 / 2$,

$$(y - y_r)^2 \leq 2\left(V(0) - \frac{1}{\xi_n}\sum_{j=1}^{n} N_j\right)e^{-\xi_n t} + \frac{2}{\xi_n}\sum_{j=1}^{n} N_j \tag{17}$$

Apparently, the expression (14) can be derived. Due to the second term in (16), the control law can guarantee the global bounded tracking of all closed-loop signals rather than the exponential convergence of the tracking error. The error can decreases by setting in reason the value of ε_i, Γ_{wi}, Γ_{vi}, σ_{wi}, σ_{vi}.

5 Simulation Example

In this section, the neural adaptive singularity-free controller by backstepping is applied to the continuous stirred-tank reactor system (CSTR) as Fig. 1. In CSTR, the materiel assumes well-mixedness, and the volume is constant. An irreversible exothermic reaction A→B occurs. The dynamic equation of CSTR is shown in (18). The corresponding physical constants and their normal values are given in Table 1.

$$\begin{aligned}
\frac{dC}{dt} &= \frac{Q}{V}(C_f - C) - k_0 \times 10^7 \cdot \exp\left(-\frac{E}{RT}\right)C \\
\frac{dT}{dt} &= \frac{Q}{V}(T_f - T) + \frac{-\Delta H}{\rho c_p} k_0 \times 10^7 \cdot \exp\left(-\frac{E}{RT}\right)C + \frac{UA}{\rho c_p V}(T_c - T) \\
\frac{dT_c}{dt} &= \frac{Q_c}{V_c}(T_{cf} - T_c) + \frac{\tilde{U}A_c}{\rho_c c_{pc} V_c}(T - T_c)
\end{aligned} \tag{18}$$

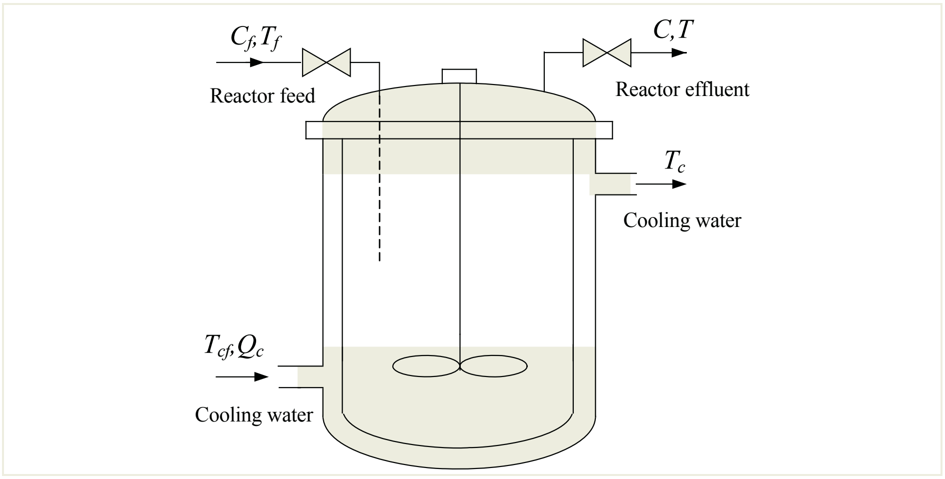


Fig. 1. Schematic of a continuous stirred-tank reactor system

Table 1. Normal Values of CSTR

Symbol	Quantity	Normal value	Units
Q	Flow of feed	10	L/min
V	Volume of CSTR	10	L
C	Concentration of A in CSTR	0.55	mol/L
C_f	Concentration of A in feed	1.0	mol/L
E/R	Activation energy	6000	K
T_f	Temperature of the feed	300	K
T	Temperature in CSTR	317.16	K
T_{cf}	Temperature of cooling water	280	K
T_c	Temperature in jacket	300.02	K
V_c	Volume of jacket	5	L
Q_c	Flow of cooling water	8	L/min
k_0	Rate constant	5.34	min^{-1}
$-\Delta H/(\rho c_p)$		105	K·L/mol
$UA/(\rho c_p V)$		0.5	min^{-1}
$\tilde{U}A_c/(\rho_c c_{pc} V_c)$		0.8	min^{-1}

Let $x_1=C$, $x_2=10^7\times\exp(-E/(RT))$, $x_3=T_c$ are the states respectively. The manipulated variable is Q_c; the concentration of A in CSTR is the output of system. The system (18) can be changed into the 3-order affine nonlinear one as (1), where, Δ_1, $\Delta_2\in$ [-0.0025, 0.0025]; $\Delta_3\in$ [-1, 1] are the unknown stochastic disturbances. According to (18) and Table 1, the gains satisfy: $-6<g_1(\chi_1)<0$; $-0.1< g_2(\chi_2)<0$; $-6<g_3(\chi_3)<0$. So, we choose $G_1(\chi_1)=-6$; $G_2(\chi_2)=-0.1$; $G_3(\chi_3)=-6$. The actual control u and adaptive law of the NN weights are defined as

$$u=-\frac{1}{G_3(\chi)}\left[\hat{W}_3^{\mathrm{T}}\Psi(\hat{V}_3^{\mathrm{T}}Z_3)+c_3\beta_{3\max}\,|\mu_3|^2\,z_3+k_3(t)z_3+G_2(\chi_2)z_2\right]$$

$$\dot{\hat{W}}_i=\Gamma_{wi}[(\hat{\Psi}-\hat{\Psi}'\hat{V}_i^{\mathrm{T}}Z_i)z_i-\sigma_{wi}\hat{W}_i],\ \dot{\hat{V}}_i=\Gamma_{vi}[Z_i\hat{W}_i^{\mathrm{T}}\hat{\Psi}'z_i-\sigma_{vi}\hat{V}_i]\ i=1,2,3.$$

where, $z_1=x_1-y_r$, $z_2=x_2-\alpha_1$, $z_3=x_3-\alpha_2$, $Z_1=[\chi_1^{\mathrm{T}},y_r,y_r',1]^{\mathrm{T}}$, $Z_2=[\chi_2^{\mathrm{T}},\alpha_1,\dot{\alpha}_1,1]^{\mathrm{T}}$, $Z_3=[\chi_3^{\mathrm{T}},\alpha_2,\dot{\alpha}_2,1]^{\mathrm{T}}$, $k_i(t)=[\|Z_i\hat{W}_i^{\mathrm{T}}\hat{\Psi}_i'\|_{\mathrm{F}}^2+\|\hat{\Psi}_i'\hat{V}_i^{\mathrm{T}}Z_i\|^2-G_i(\chi_i)+0.5]/\varepsilon_i$, $i=1,2,3$; l=10; $\hat{W}_1,\hat{W}_2,\hat{W}_3\in R^{10}$, $\hat{V}_1\in R^{4\times10}$, $\hat{V}_2\in R^{5\times10}$, $\hat{V}_3\in R^{6\times10}$. $\hat{W}_1(0),\hat{W}_2(0),\hat{W}_3(0),\hat{V}_1(0),\hat{V}_2(0),\hat{V}_3(0)$ is 0; $x(0)$=[0.55, 0.0632, 280]. Other parameters are as follow: $\Gamma_{w1}=diag\{2.0\}$, $\Gamma_{w2}=diag\{2.0\}$, $\Gamma_{w3}=diag\{2.0\}$, $\Gamma_{v1}=diag\{2.0\}$, $\Gamma_{v2}=diag\{2.0\}$, $\Gamma_{v3}=diag\{2.0\}$ $\sigma_{w1}=\sigma_{w2}=\sigma_{v1}=\sigma_{v2}=0.01$, $\sigma_{w3}=\sigma_{v3}=0.001$, $\varepsilon_1=\varepsilon_2=\varepsilon_3=0.1$, and $\gamma=3.0$. Let $c_i\beta_{imax}|\mu_i|^2=2.0$, i=1, 2, 3 and the sampling time is 0.01min. In order to obtain the smooth output, the filter is used in command input as follow: $0.5/(s^2+0.8s+0.5)$.

In the simulation, the constrained manipulated variable and time-variable parameters of CSTR is considered. Let $Q_c\in$ [0, 14]; the value of $\tilde{U}A_c/(\rho_c c_{pc} V_c)$ is changed from 0.8 to 1 at 30 min; the value of $-\Delta H/(\rho c_p)$ is changed from 105 to 115 at 50 min; k_0 is changed from 5.33 to 5.04 at 70 min. The result of simulation is shown as Fig.2 and Fig.3.

In Fig. 2, a) shows the output of closed-loop system ant its expect; b) are temperatures of CSTR and jacket; c) is the flow of cooling water; d) shows the tracking error of closed-loop system. Fig.3 is the Euclid norm of the NN weight matrix in the process of nonlinear estimation. The curves in Fig.3 show the adaptive ability of NN for the parameter change.

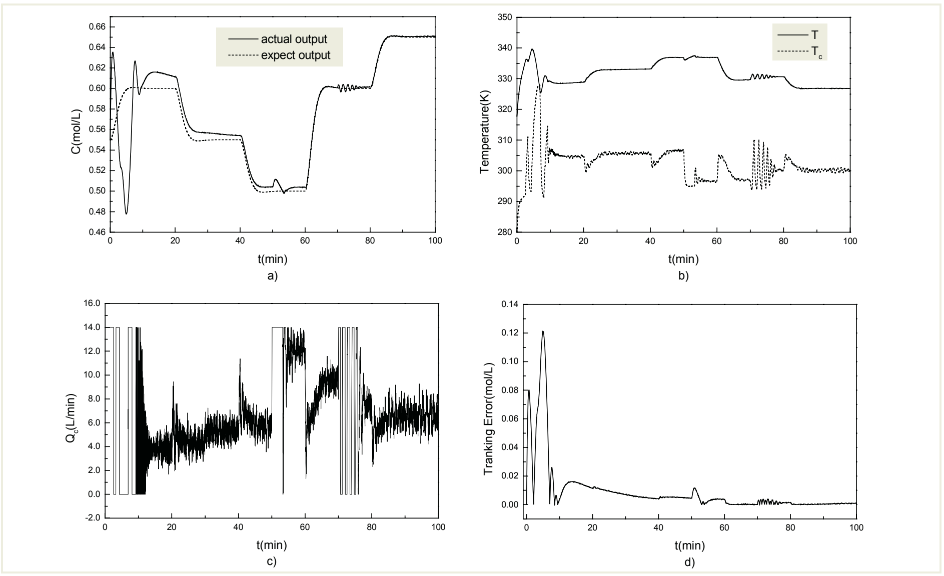


Fig. 2. Result of simulation of system

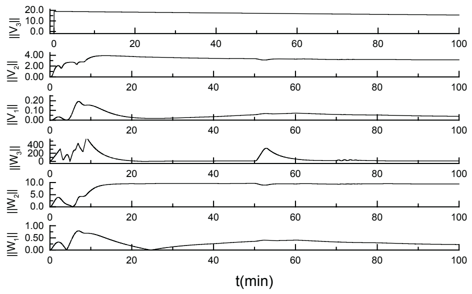


Fig. 3. Euclid norm of the NN weight matrix

6 Conclusion

The control problem of unknown semi-strict feedback nonlinear systems not transformable into the linear parameterized forms is discussed. The procedure is developed for the design of neural adaptive controller by backstepping technique. In order to avoid the zero crossing of the estimated control gains, a novel quasi-weighted Lyapunov function is modified in design. The result of the simulation means that the controller is robust to some nonlinear uncertainties and bounded disturbance, and the control scheme can guarantee the global boundedness of all closed-loop signals.

Acknowledgements

We gratefully acknowledge the support from the National Natural Science Foundation of China: No. 60274002.

References

1. Kanellakopoulos, L., Kokotovic, P.V., Morse, A.S.: Systematic Design of Adaptive Controllers for Feedback Linearizable Systems. IEEE Transactions on Automatic Control. **36** (1991) 1241-1253
2. Krstic, M., Kanellakopoulos, I., Kokotovic, P.V.: Adaptive Nonlinear Control without Overparametrization. Systems & Control Letters, **19** (1992) 177-185
3. Polycarpou, M.M., Ioannou, P.A.: A Robust Adaptive Nonlinear Control Design. Automatica, **32** (1996) 423-427
4. Yao, B., Tomizuka, M.: Adaptive Robust Control of Nonlinear Systems in a Semi-Strict Feedback Form. Automatica, **33** (1997) 893-900
5. Zhang, T., Ge, S.S., Hang, C.C.: Adaptive Neural Network Control for Strict-Feedback Nonlinear Systems Using Backstepping Design. Automatica, **36** (2000) 1835-1846
6. Wang, D., and Huang, J.: Adaptive Neural Network Control for a Class of Uncertain Nonlinear Systems in Pure-Feedback Form. Automatica, **38** (2002) 1365-1372
7. Hornik, K.: Approximation Capabilities of Multilayer Feed-Forward Networks. Neural Networks, **4** (1991) 251-257
8. Kwan, C., Lewis, F.L., Dawson, D.M.: Robust Neural-Network Control of Rigid-Link Electrically Driven Robots. IEEE Transactions on Neural Networks, **9** (1998) 581-588
9. Polycarpou, M.M.: Stable Adaptive Neural Control Scheme for Nonlinear Systems. IEEE Transactions on Automatic Control, **41** (1996) 447-451
10. Ge, S.S., Hang, C.C., Zhang T.: A Direct Adaptive Controller for Dynamic Systems with a Class of Nonlinear Parameterizations. Automatica, **35** (1999) 741-747
11. Ge, S.S., Hang, C.C., Zhang, T.: Nonlinear Adaptive Control Using Neural Networks and its Application to CSTR Systems. Journal of Process Control, **9** (1998) 313-323

Parameter Estimation of Fuzzy Controller Using Genetic Optimization and Neurofuzzy Networks

Sungkwun Oh[1], Seokbeom Roh[2], and Taechon Ahn[2]

[1] 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

[2] Department of Electrical Electronic and Information Engineering, Wonkwang University, 344-2, Shinyong-Dong, Iksan, Chon-Buk 570-749, South Korea

Abstract. In this study, we introduce a noble neurogenetic approach to the design of fuzzy controller. The design procedure dwells on the use of Computational Intelligence (CI), namely genetic algorithms and neurofuzzy networks (NFN). The crux of the design methodology is based on the selection and determination of optimal values of the scaling factors of the fuzzy controllers, which are essential to the entire optimization process. First, the tuning of the scaling factors of the fuzzy controller is carried out, and then the development of a nonlinear mapping for the scaling factors is realized by using GA based NFN. The developed approach is applied to a nonlinear system such as an inverted pendulum where we show the results of comprehensive numerical studies and carry out a detailed comparative analysis.

1 Introduction

In parallel to PID controllers that are regarded nowadays as the standard control constructs of numeric control [1], [2], [3], [4] fuzzy controllers have positioned themselves in a similar dominant role at the knowledge-rich end of the entire spectrum of control algorithms. PID controllers are superb when it comes to linear systems or nonlinear systems with an operation mode confined to a small neighborhood around a given setpoint. The advantages of the fuzzy controllers are positioned at the opposite end of the scale as we envision their full strength in the setting of nonlinear systems (as these controllers are nonlinear mappings in the first place) and when dealing with high deviations from the setpoint. One of the difficulties in the construction of the fuzzy controller is to derive a set of optimal control parameters of the controller such as linguistic control rules, scaling factors, and membership functions of the fuzzy controller. With an attempt to enhance the quality of the control knowledge conveyed by the expert (and this usually applies to the matter of calibration of such initial domain knowledge), genetic algorithms (GAs) have already started playing a pivotal role. More specifically considering a vast number of parameters of the fuzzy controller they are instrumental in carrying out a global search in the overall parameter space. One should stress however that evolutionary computing (such as GAs) is computationally intensive and this may be a point of concern when dealing with amount of time available to such search. For instance, when controlling a nonlinear plant such as an inverted pendulum of which initial states vary in each case, the search time required by GAs could be prohibitively high when dealing with dynamic systems. As a consequence, the parameters of the fuzzy controller cannot be easily adapted to the changing initial states of this system such as an angular position and an angular velocity of the pendulum. To alleviate this shortcoming, we introduce a nonlinear mapping from the initial states of the system and the corresponding optimal values of the parameters. With anticipation of the nonlinearity residing within such transformation, in its realization we consider GA-based NFN. Bearing this mind, the development process consists of two main phases. First, using genetic optimization we determine optimal parameters of the fuzzy controller for various initial states (conditions) of the dynamic system. Second, we build up a nonlinear model that captures a relationship between the initial states of the system and the corresponding genetically optimized control parameters. The paper includes the experimental study dealing the inverted pendulum with the initial states changed.

2 Design Methodology of Fuzzy Controller

The block diagram of fuzzy PID controller is shown in Fig. 1. Note that the input variables to the fuzzy controller are transformed by the scaling factors (GE, GD, GH, and GC) whose role is to allow the fuzzy controller to properly "perceive" the external world to be controlled.

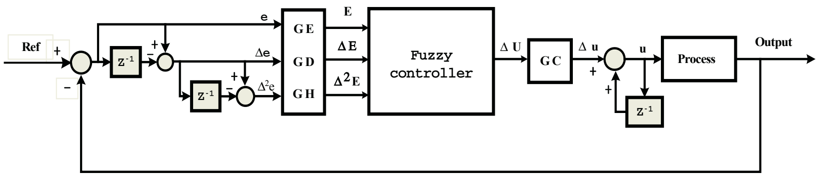


Fig. 1. An overall architecture of the fuzzy PID controller

The above fuzzy PID controller consists of rules of the following form, cf. [3]

$$R_j : \text{if } E \text{ is } A_{1j} \text{ and } \Delta E \text{ is } A_{2j} \text{ and } \Delta^2 E \text{ is } A_{3j} \text{ then } \Delta U_j \text{ is } D_j$$

The capital letters standing in the rule (R_j) denote fuzzy variables whereas D is a numeric value of the control action.

3 Auto-tuning of the Fuzzy Controller by GAs

The genetic search is guided by a reproduction, mutation, and crossover. The standard ITAE expressed for the reference and the output of the system under control is treated as a fitness function [2].

The overall design procedure of the fuzzy PID controller realized by means of GAs is illustrated in Fig. 2. It consists of the following steps

[step 1] Select the general structure of the fuzzy controller

[step 2] Define the number of fuzzy sets for each variable and set up initial control rules.

[step 3] Form a collection of initial individuals of GAs. We set the initial individuals of GAs for the scaling factor of fuzzy controller.
[step 4] Here, all the control parameters such as the scaling factors GE, GD, GH and GC are tuned at the same time.

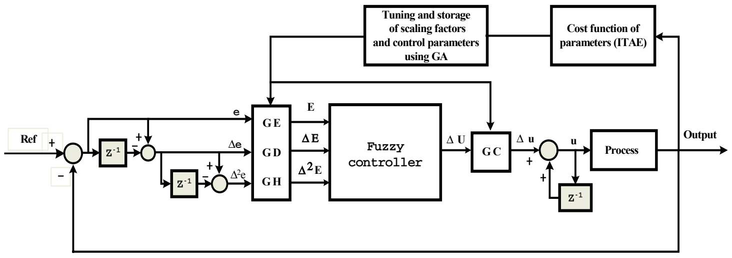

Tuning and storage of scaling factors and control parameters using GA
Cost function of parameters (ITAE)
Ref
+
-
e
Δe
Δ²e
GE
GD
GH
E
ΔE
Δ²E
Fuzzy controller
Δ U
GC
Δ u
u
Process
Output
z⁻¹

Fig. 2. The scheme of auto-tuning of the fuzzy PID controller involving estimation of the scaling factors

4 The Estimation Algorithm by Means of GA-Based Neurofuzzy Networks (NFN)

Fig. 3 visualizes an architecture of such NFN for two-input and one-output, where each input assumes three membership functions. The circles denote processing units of the NFN. The node indicated ∏ denotes a Cartesian product, whose output is the product of all the incoming signals. And N denotes the normalization of the membership grades.

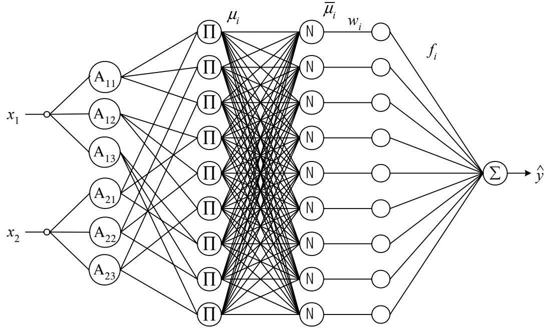


Fig. 3. NFN structure by means of the fuzzy space partition realized by fuzzy relations

The output f_i of each node generates a final output $\hat{y}$ of the form

$$\hat{y} = \sum_{i=1}^{n} f_i = \sum_{i=1}^{n} \bar{\mu}_i \cdot w_i = \sum_{i=1}^{n} \frac{\mu_i \cdot w_i}{\sum_{i=1}^{n} \mu_i} \tag{1}$$

The learning of the NFN is realized by adjusting connections of the neurons and as such it follows a standard Back-Propagation (BP) algorithm

As far as learning is concerned, the connections change as follows

$$w(new) = w(old) + \Delta w \tag{2}$$

where the update formula follows the gradient descent method

$$\Delta w_{ij} = \eta \cdot \left(-\frac{\partial E_p}{\partial w_i} \right) = -\eta \cdot \frac{\partial E_p}{\partial \hat{y}_p} \cdot \frac{\partial \hat{y}_p}{\partial f_i} \cdot \frac{\partial f_i}{\partial w_i} = 2 \cdot \eta \cdot (y_p - \hat{y}_p) \cdot \overline{\mu}_i \tag{3}$$

with η being a positive learning rate. The complete update formula combining the already discussed components is

$$\Delta w_{ij} = 2 \cdot \eta \cdot (y_p - \hat{y}_p) \cdot \mu_i + \alpha(w_{ij}(t) - w_{ij}(t-1)) \tag{4}$$

(Here the momentum coefficient, α, is constrained to the unit interval).

In this algorithm, to optimize the learning rate, momentum term and fuzzy membership function of the above NFN we use the genetic algorithm.

5 Experimental Studies

In this section, we demonstrate the effectiveness of the fuzzy PID controller by applying it to the inverted pendulum system. Our control goal here is to balance the pole without regard to the cart's position and velocity [7].

Tuning of Control Parameters and Estimation

We genetically optimize control parameters with a clear intent of achieving the best performance of the controller. GAs are powerful nonlinear optimization techniques. However, the powerful performance is obtained at the cost of expensive computational requirements and much time. To overcome this weak point, first, we select several initial angular positions and angular velocities and obtain the auto-tuned control parameters by means of GAs according to the change of each selected initial angular positions and velocities, then build a table. Secondly, we use GA-based NFN to estimate the control parameters in the case that the initial angular positions and velocities of the inverted pendulum are selected arbitrarily within the given range. Here we show that the control parameters under the arbitrarily selected initial angular position are not tuned by the GAs but estimated by the estimation algorithm of GA-based NFN. Table 1 shows the estimated scaling factors of the fuzzy PID controller and describes performance index (ITAE, Overshoot(%))of the fuzzy PID controller with the estimated scaling factors.

Table 1. The estimated parameters by means of the GA-based NFN and performance index(ITAE, Overshoot(%)) of the fuzzy PID controller

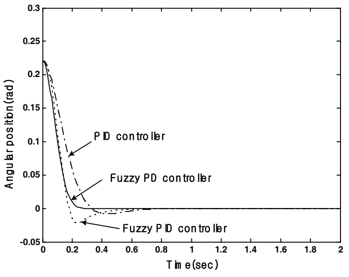
Case	Initial Angle(rad)	Initial Angular Velocity	GE	GD	GH	GC	ITAE	Over Shoot(%)
1	0.22	0.22	3.5149	1.152	0.046	0.811	0.255	9.889
2	0.22	0.45	3.457	1.174	0.045	0.818	0.255	7.495
3	0.22	0.78	3.275	1.362	0.046	0.744	0.315	6.899
4	0.45	0.22	1.829	1.049	0.043	0.934	1.012	0.0
5	0.45	0.45	1.763	1.295	0.045	0.807	1.566	0.0
6	0.45	0.78	1712	0.986	0.048	0.805	1.350	0.0
7	0.78	0.22	1.156	1.140	0.051	0.851	3.412	1.147
8	0.78	0.45	1.126	0.988	0.042	0.791	3.872	2.979
9	0.78	0.78	1.190	1.650	0.053	0.952	5.843	3.723

In case of the fuzzy PD controller, the estimated scaling factors and performance index are shown Table 2.

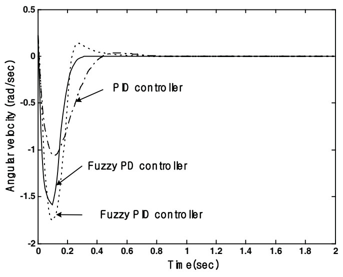
Table 2. The estimated parameters by means of the GA-based NFN and performance index(ITAE, Overshoot(%), Rising time(sec)) of the fuzzy PD controller

Case	Initial Angle(rad)	Initial Angular velocity	GE	GD	GC	ITAE	Over Shoot(%)
1	0.22	0.22	8.478	0.604	1.481	0.142	0.103
2	0.22	0.45	80381	0.600139	1.487	0.155	0.097
3	0.22	0.78	8.589	0.610	1.472	0.181	0.096
4	0.45	0.22	7.301	0.558	1.475	0.860	0.123
5	0.45	0.45	7.139	0.557	1.483	0.916	0.081
6	0.45	0.78	7.422	0.559	1.484	1.010	0.133
7	0.78	0.22	7.204	0.550	1.473	4.744	0.100
8	0.78	0.45	7.188	0.547	1.483	4.965	0.101
9	0.78	0.78	7.311	0.553	1.480	5.561	0.094

Fig. 4 demonstrates (a)pole angle (b)pole angular velocity for initial angle θ = 0.22(rad) and initial angular velocity $\dot{\theta}$=0.22(rad/sec) (Case 1).



(a)



(b)

Fig. 4. (a)pole angle (b)pole angular velocity for initial angle θ= 0.22(rad) and initial angular velocity $\dot{\theta}$=0.22(rad/sec) (Case 1)

From the above Fig. 4, we know that the fuzzy PD and fuzzy PID control effectively the inverted pendulum system. The output performance of the fuzzy controllers such as the fuzzy PD and the fuzzy PID controller including nonlinear characteristics are superior to that of the PID controller especially when using the nonlinear dynamic equation of the inverted pendulum. Moreover, the proposed estimation algorithm such as GA-based NFN generates the preferred model architectures. Especially the fuzzy PD controller describes the preferred one among the controllers.

6 Conclusions

In this paper, we have proposed a two-phase optimization scheme of the fuzzy PID and PD controllers. The parameters under optimization concern scaling factors of the input and output variables of the controller that are known to exhibit an immense impact on its quality. The first phase of the design of the controller uses genetic computing that aims at the global optimization of its scaling factors. In the second phase, we construct a nonlinear mapping between the initial conditions of the system and the corresponding values of the scaling factors. From the simulation studies, using genetic optimization and the estimation algorithm of the GA-based neurofuzzy networks model, we showed that the fuzzy PD/PID controller controls effectively the inverted pendulum system. While the study showed the development of the controller in the experimental framework of control of a specific dynamic system (inverted pendulum), this methodology is general and can be directly utilized to any other system.

Acknowledgements

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. Oh, S.K., Pedrycz, W.: The Design of Hybrid Fuzzy Controllers Based on Genetic Algorithms and Estimation Techniques. Kybernetes. **31** (2002) 909-917
2. Oh, S.K., Ahn, T., Hwang, H., Park, J., Woo, K.: Design of a Hybrid Fuzzy Controller with the Optimal Auto-tuning Method. Journal of Control, Automation and Systems Engineering. **1** (1995) 63-70
3. Li, H.X.: A comparative design and tuning for conventional fuzzy control. IEEE Trans. Syst., Man, Cybern. B. **27** (1997) 884-889
4. Goldberg, D.E.: Genetic algorithms in Search, Optimization, and Machine Learning. Addison-Wesley (1989)
5. Yamakawa, T.: A New Effective Learning Algorithm for a Neo Fuzzy Neuron Model. 5th IFSA World Conference (1993) 1017-1020
6. Park, B.J., Oh, S.K., Pedrycz, W.: The Hybrid Multi-layer Inference Architecture and Algorithm of FPNN Based on FNN and PNN. 9th IFSA World Congress. (2001) 1361-1366
7. Jang, J.R.: Self-Learning Fuzzy Controllers Based on Temporal Back Propagation. IEEE Trans. On Neural Networks, **3** (1992) 714-723
8. 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**((2004) 362-373
9. Park, H.S., Oh, S.K.: Fuzzy Relation-based Fuzzy Neural-Networks Using a Hybrid Identification Algorithm. International Journal of Control, Automations, and Systems, **1** (2003) 289-300

A Fuzzy CMAC Controller with Eligibility

Zhipeng Shen[1], Chen Guo[1,2], Jianbo Sun[1], and Chenjun Shi[1]

[1] School of Automation and Electrical Enginerring,
Dalian Maritime University, Dalian, Liaoning 116026, China
[2] State Key Laboratory of Intelligent Technology and system,
Tsinghua University, Beijing 100084, China
s_z_p@263.net, guoc@dlmu.edu.cn

Abstract. Based on eligibility a fuzzy CMAC controller (FCE) is proposed. The structure of FCE system is presented, and its learning algorithm is deduced. To make the algorithm fit to on-line control, an efficient implementation of FCE method is also given. Applying the FCE controller in a ship steering control system, the simulation results show that the ship course can be properly controlled in case of the disturbances of wave, wind, current. It is demonstrated that the proposed algorithm is a promising alternative to conventional autopilots.

1 Introduction

CMAC is an acronym for Cerebellar Model Articulation Controller, which was first described by Albus in 1975 as a simple model of the cortex of the cerebellum [1], [2]. Since then it has been extended in many different ways, and used in a wide range of different applications. It has already been successfully applied in the approximation of complicated functions, and used in the realm of system recognition and control. Despite its biological relevance, the main reason for using the CMAC is that it operates very fast, which makes it suitable for real-time adaptive control. Eligibility is an idea described firstly by Croft, and it has been used for many years as part of the reinforcement learning paradigm [3]. Many scholars are doing research on it, and achieve a lot beneficial result. Thereinto, Sutton had made systematic and future work on it in his doctoral thesis[4].

Combining eligibility into Fuzzy CMAC neural network, a Fuzzy CMAC controller with Eligibility (FCE) is proposed in the paper. The basic idea is that each weight in FCMAC neural network [5] is given an associated value called its "eligibility". Applying the FCE controller to ship steering control, the parameters of controller are on-line learned and adjusted. And the FCE algorithm will be formally derived by considering how to optimize an error function of the controlled system's inputs and outputs. To make the algorithm fit to on-line control, the efficient implementation of FCE method is also given. Simulation results show that the ship course can be properly controlled in case of the disturbances of wave, wind. It is demonstrated that the proposed algorithm is a promising alternative to conventional autopilots.

2 FCE System Structure

The basic FCE system shown in Figure 1. The block F is the controlled system that implements:

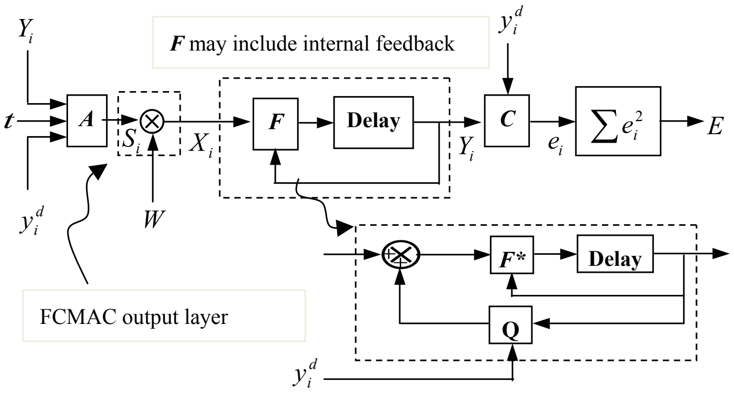


Fig. 1. FCE controller and controlled system

$$Y_{i+1} = F(Y_i, X_i)\,. \tag{1}$$

As shown in the figure the block *F* can also incorporate any additional feedback controller Q that the basic system might have. The FCMAC has been split into two parts in the figure:

(1) The block *A* which represents the input layers (or association unit transformations). At each time step this block reads the system state Y_i (one expected value y_i^d or perhaps the current time *t*) and encodes it in the "sensor" vector S_i.
(2) The output layer which multiplies the vector S with the weight matrix *W* to get the output *X* ($X = SW$).At each time step the "critic" block *C* computes a scalar error value e_i that depends on the current state. The squares of these error values are summed over time to get the global error value *E*.

If the system's desired behavior is to track a scalar reference position y_i^d ,thus

$$e_i = y_i^d - CY_i\,. \tag{2}$$

Here *C* is a matrix that selects whatever element of *y* corresponds to a position. Then the total error *E* is given by:

$$E = \sum_{i=1}^{T} e_i^2\ . \tag{3}$$

The critic function is chosen so that when E is minimized the system achieves some desired behavior.

3 FCE Learning Algorithm

The purpose of the FCE learning algorithm is to modify *W* using gradient descent so that the error *E* is minimized. The gradient descent process for this system is:

$$w_j \leftarrow w_j + \alpha \frac{dE}{dw_j}\,. \tag{4}$$

Where w_j is an element of the matrix W and α is a scalar learning rate. This equation gives the modification made to weight w_j as a result of one iteration of learning. Now, from the chain rule:

$$\frac{dE}{dw_j} = \sum_{i=1}^{T-1} \left[2 \sum_{k=i+1}^{T} e_k \frac{de_k}{dX_i} \right] \cdot \frac{dX_i}{dw_j} . \tag{5}$$

Now for a key step to determine the meaning of equation (6), and to derive a practical algorithm, F is approximated by a linear system F^*:

$$Y_{i+1} = AY_i + BX_i \tag{6}$$

$$e_i = CY_i - y_i^d . \tag{7}$$

Combining equation (6) and (7) with equation (5):

$$\frac{de_{i+k}}{dX_i} = CA^{k-1}B \qquad (k > 0) . \tag{8}$$

thus

$$\frac{dE}{dw_j} = 2\sum_{k=2}^{T} e_k C\left[A^{k-i-1}B\hat{S}_i^j\right] = 2\sum_{k=2}^{T} e_k C\xi_k^j . \tag{9}$$

where $\xi_k^j = \sum_{i=1}^{k-1} A^{k-i-1}B\hat{S}_i^j \ , \hat{S}_i^j = \dfrac{\partial X_i}{\partial w_j}$

$\hat{S}_i$ is all zero except for the element whose corresponding neural weight w_j is excited. And here ξ_k^j is called the *eligibility* signal. Based on the above equations, the FCE learning algorithm can be deduced:

$$\xi_1^j = 0 \quad , \quad \xi_{i+1}^j = A\xi_i^j + B\hat{S}_i^j . \tag{10}$$

$$w_j \leftarrow w_j + \alpha\sum_{k=2}^{T} e_k C\xi_k^j . \tag{11}$$

Note that a factor of 2 has been combined into .Every FCMAC weight w_j requires an associated eligibility vector ξ^j .The order of the eligibility model is the size of the matrix A. There is a relationship between the two constants α and C : if the magnitude of C is adjusted then α can be changed to compensate. Because of this the convention will be adopted that the magnitude of C is always set to one ($|C| = 1$) and then the resulting α is the main FCE learning rate.

4 The Efficient Implementation of FCE Learning Algorithm

A naive implementation of the training equations is very simple – just update the eligibility state for every weight during each time step. Consider a FCMAC with n_w weights and n_a association units, the naive approach usually requires too much computation to be practical in an online controller.

As follows, the FCE algorithm performs the same computations using a far more efficient approach which eliminates the recalculation of redundant information. FCE performs just O(n_a) extra operations per time step, so the total computation time is still O(n_a) – the same order as the basic CMAC algorithm.

The algorithm described below requires the system F^* to have an impulse response that eventually decays to zero. This is equivalent to requiring that the eigenvalues of A all have a magnitude less than one. This will be called the "decay-to-zero" assumption. The next simulation part will explain how to get around this requirement in the ship steering system.

The weights is divided into three categories according to their values:

(1) *Active weights*: where the weight is one of the n_a currently being accessed by the FCMAC. There are always n_a active weights.
(2) *Inactive weights*: where the weight was previously active and its eligibility has yet to decay to zero.
(3) *Retired weights*: where the weight's eligibility has decayed sufficiently close to zero, so no further weight change will be allowed to take place until this weight becomes active again.

Figure 2 shows how a weight makes the transition between these different states. FCE does not have to process the retired weights because their values do not change (their eligibilities are zero and will remain that way) and they do not affect the FCMAC output. An active weight turns in to an inactive weight when the weight is no longer being accessed by the FCMAC (transition 1 in figure 2). An inactive weight turns in to a retired weight after σ time steps have gone past (transition 3 in figure 2).The value of σ is chosen so that after σ time steps a decaying eligibility value is small enough to be set to zero. At each new time step a new set of weights are made active. Some of these would have been active on the previous time step, others are transferred from the inactive and retired states as necessary (transitions 2 and 4 respectively in Figure 2).

If a weight w is made inactive at time i_1 and it becomes active or retired at time i_2, its value (and the corresponding eligibility value) must be corrected to account for the interval $i_1 \ldots i_2$ during which it was not modified.

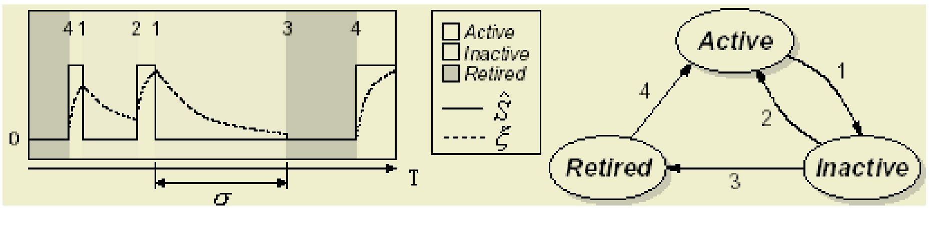


Fig. 2. The three states of FCE weight, and the transition between them

5 Simulation Study

Figure 3 shows the ship steering control system applying FCE controller. Its input are course error $\Delta\psi=\psi_r(k)-\psi(k)$ and fore turning angular velocity $r(k)$. Its output is the rudder angle $\delta(k)$. $\Delta\psi$ varies between (-20°,20°), r between (-0.9°/sec,0.9°/sec),and δ is (-35°,35°).

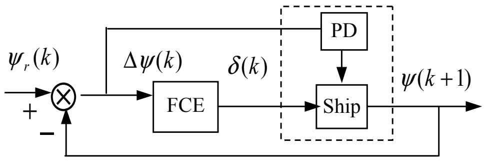


Fig. 3. Ship steering control system

Ship motion model can be described as second-order non-linear ship response model:

$$\ddot{\psi}+\frac{K}{T}H(\dot{\psi})=\frac{K}{T}\delta \cdot \tag{12}$$

To some unstable ship, $\dot{\psi}/T$ must be replaced with a non-linear term. So the second-order non-linear ship response model is expressed

$$\ddot{\psi}+\frac{K}{T}H(\dot{\psi})=\frac{K}{T}\delta \cdot \tag{13}$$

parameters a,β and K,T is related to ship's velocity.

The FCE algorithm described above requires the system F^* to have an impulse response that eventually decays to zero. So a PD feedback control element is joined, then the ship state model is changed:

$$X = AX + Bu \ . \tag{14}$$

$$Y = CX \, . \tag{15}$$

where

$$X=[\dot{\varphi},\varphi]^T, A=\begin{bmatrix} \text{-(1+Kkd)/T} & \text{-Kkp/T} \\ 1 & 0 \end{bmatrix}$$

$$B=[K/T,0]^T, C=[0,1], u=\delta$$

Transfer the state matrix into discrete format, the eligibility curve can be attained as fig.4 shown. Here, K=0.36,T=230,K_p=1.2,K_d=15.The eligibility decays to zero about 80s from fig.8, so the eligibility decay parameter can be selected as $\sigma = 100$.

Fig.5 shows the control curve result when set course is 10°, wind force is Beaufort 5 and wind direction is 30°.While Fig.6 and Fig.7 show the control curve result when set course is 5°~15°~20°. From the compared curves ,the proposed FCE control has better real-time quality and fast tracking speed. In term of course, it has no over-training

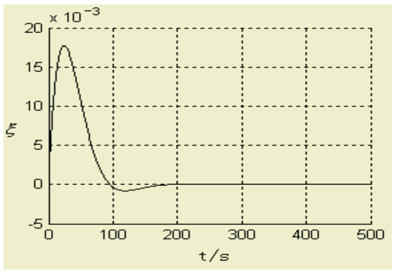


Fig. 4. Eligibility curve

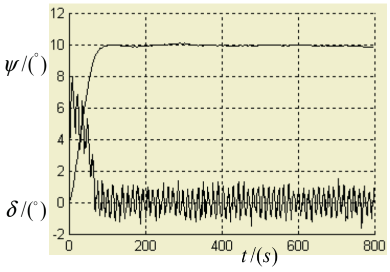


Fig. 5. Control curve, course 10°

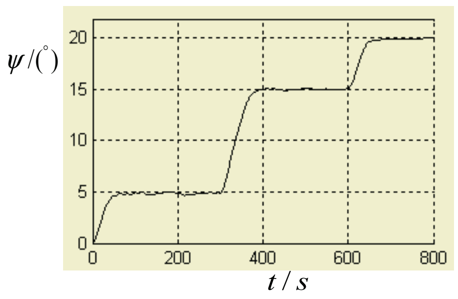


Fig. 6. Control curve, course 5° ~15°~20°

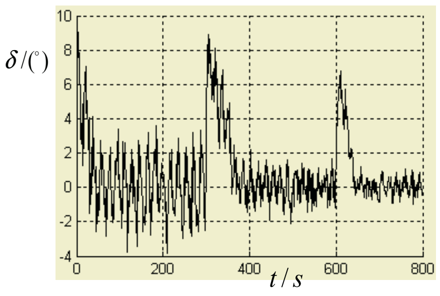


Fig. 7. Control curve, course 5° ~15°~20°

results and has satisfied tracking effect; as to rudder angle, at beginning the bigger angle is accelerated to start up ,then regained to stable angle needed. The curves indicate that the course tracking is fast, control action reasonable and meet the performance of ship steering. The control result is partial satisfied.

6 Conclusions

A fuzzified CMAC controller with eligibility (FCE) is proposed. The eligibility can predict the uncertainty of controlled system, and improve the system stability. The structure of FCE system is presented, and its learning algorithm is deduced. To make the algorithm fit to on-line control ,the efficient implementation of FCE method is also given. Applying the FCE controller in a ship steering control system, the simulation results show that the ship course can be properly controlled when changeable wind and wave exist. It is demonstrated that the proposed algorithm is a promising alternative to conventional autopilots.

References

1. Albus, J.S.: A New Approach to Manipulator Control: The Cerebellar Model Articulation Controller(CMAC). Trans ASME-J Dyn Syst Meas Control, **97** (1975) 220-227
2. Albus, J.S.: Data Storage in the Cerebellar Model Articulator Control (CMAC). Trans ASME-J Dyn Syst Meas Control, **97** (1975) 228-233
3. Malaka, R., Hammer, M.: Real-time Models of Classical Conditioning. In Proceedings of the International Conference on Neural Networks (ICNN'96), Washington, **2** (1996) 768-773
4. Sutton, R.S.: Temporal Credit Assignment in Reinforcement Learning. PhD thesis, University of Massachusetts, Amherst, MA (1984) 41-75
5. Deng, Z., Sun, Z.: A Fuzzy CMAC Network., Acta Automatica Sinica, China, 21 (1995) 288-293

A Novel Intelligent Controller Based on Modulation of Neuroendocrine System

Bao Liu[1], Lihong Ren[1], and Yongsheng Ding[1,2]

[1] College of Information Sciences and Technology
[2] MOE Engineering Research Center of Digitized Textile & Fashion Technology,
Donghua University, Shanghai 200051, China
ysding@dhu.edu.cn

Abstract. Based on the modulation of neuroendocrine system and the hypothalamo-pituitary-adrenal axis model, we present a novel intelligent controller (NEIC). The NEIC, with a typical two-level structure, is composed of a primary control unit and a secondary one. According to the real-time control error, the primary control unit can adjust the control parameters of the secondary unit dynamically based on the hormone-release law. The NEIC can strengthen or weaken control action during the different control error range. The simulation results demonstrate that the NEIC's control stability and adaptability are better than those of the conventional PID controller.

1 Introduction

Since the 1970s, the biocybernetics has got great attention and developed rapidly. Currently, intelligent control theory and biological technology have been combined more closely [1]. More bio-intelligent controllers and algorithms based on neural network or immune system are introduced, e.g. [1-2]. As we know, the neuroendocrine system is also one of the major physiological systems and has some special modulation mechanism. And it has better self-adaptability and stability [3]. The study on the artificial neuroendocrine system may inspire some novel methods to solve complicated problems and will also bring a little influence on the conventional control theory. However, there are few reports on the intelligent controller based on the neuroendocrine system. Some researchers have presented several simulation models of the hormone release [4,5], which are only used in the medical field.

In this paper, we present a novel intelligent controller (NEIC) based on the regulation mechanism of the endocrine adrenal hormone (cortisol). The NEIC, with a typical two-level structure, is composed of a primary control unit and a secondary unit. According to the real-time control error, the primary unit can adjust the control parameters of the secondary unit dynamically based on the hormone regulation law. Thus the control performance can be improved. Simulation results demonstrate that the NEIC's control stability and adaptability are better than those of the conventional PID controller.

In Section 2, the regulation mechanism of neuroendocrine hormone is briefly introduced. In Section 3, the design and implementation of NEIC are explained. In Section 4, the NEIC control performance is examined through several computer simulations. And we concluded the paper in Section 5.

2 Regulation Mechanism of Neuroendocrine Adrenal Hormone

2.1 Regulating Mechanism for Cortisol

The regulation mechanism of neuroendocrine system is similar to the closed-loop feedback control theory [4]. The cortisol is one of most important hormones in the body. The regulation process of cortisol secretion is first the hypothalamo releases adrenocorticotropic-releasing hormone (CRH), which stimulates the pituitary to secret adrenocorticotropic (ACTH). Then the ACTH stimulates adrenal to secret cortisol. The concentration of cortisol in the blood circulation system feeds back to the hypothalamo and pituitary via the cell-sensors to reduce the releasing of CRH and ACTH. It causes the decreasing of the cortisol concentration, and thus the concentration becomes stable fast [4, 5]. The regulation mechanism can be described as a two-level control system, as shown in Fig. 1.

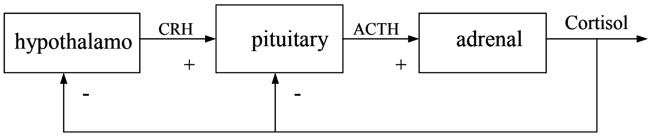


Fig. 1. The regulation mechanism of cortisol.

2.2 Hormone Secreting Theory

A general hormone releasing law is that the hormone secretion rate is monotone and nonnegative [3], and the increasing and decreasing secretion regulation mechanism follows the Hill Function as follows

$$F_{up}(G) = \frac{G^n}{T^n + G^n} \tag{1}$$

and

$$F_{down}(G) = \frac{T^n}{T^n + G^n}, \tag{2}$$

where, $T>0$ is a threshold, and $n \geq 1$ is a Hill coefficient. The threshold T and Hill coefficient n both control the function curve's slope. The function has the following properties: (a) $F_{up} = 1 - F_{down}$, (b) $F_{up,(down)}(T) = 1/2$, and (c) $0 \leq F_{up,(down)}(G) \leq 1$.

If the secretion of hormone x is regulated by hormone y, the relationship between the secretion rate of hormone x and the concentration of hormone y is [3]

$$S_x = aF_{up,(down)}(C_y) + S_{x,basal}, \tag{3}$$

where S_x is the secretion rate of hormone x, C_y is the concentration of hormone y, and $S_{x,basal}$ is the basal secretion rate of hormone x.

3 Design of the Neuroendocrine-Based Intelligent Controller

3.1 Structure of the NEIC

According to Fig. 1, we first map the hormone concentration in the blood circulation system to the process variable (*pv*). Next, map the center nerve system including hypothalamo and pituitary to the control unit. Then, map the normal hormone concentration value to the set point (*sp*). Finally, map the secreting tissue to the actuator. Thus the cortisol regulation model can be regarded as a typical structure of a two-level controller. The hypothalamo is the primary control unit, and the pituitary is the secondary unit. The abstracted NEIC system is as shown in Fig. 2.

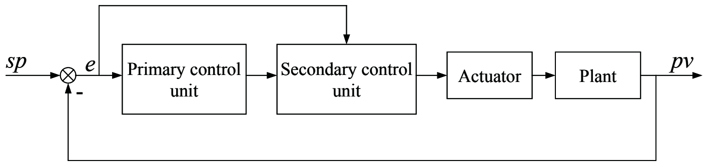


Fig. 2. The abstracted structure of NEIC system.

3.2 Design of the NEIC

3.2.1 Design of the Primary Control Unit

We design the primary unit based on the hormone secreting theory. The control error $e(t)$ is regarded as the stimulate signal of the primary control unit, and its output signal follows the Hill Function (refer to equations (1),(2), and (3)). And the output signal of primary control unit includes the corrected proportional gain $K_p(t)$, integral time $T_i(t)$, and derivative time $T_d(t)$, which are the dynamical control parameters of the secondary unit.

(1) **Standardize the control error.** We first calculate the absolute value of control error $e(t)$, then standardize the absolute value according to equation $E(t) = |e(t)|/(pv_H - pv_L)\times 100\%$, where $0 \le E(t) \le 100\%$ is the standardized control error, pv_H and pv_L is the high and low limited process variable, respectively.

(2) **Calculate the real-time corrected factors.** The real-time corrected factors $\alpha_j(t)$ of the proportional action, the integral action, and the derivative action are calculated according to the following equation $\alpha_j(t) = E(t)/(A_j + E(t)) + B_j$, where $j=p$, i, d, and A_j and B_j are set according to the different features of proportional action, integral action and derivative action. Their value ranges are both from 0 to 1. And the simplest Hill Function, i.e. coefficient n equals to 1, is selected.

(3) **Calculate the corrected control parameters.** When control error $e(t)$ is too big, the proportion gain K_p should decrease to weaken the control action, thus reduce the overshoot. In contrast, the proportion gain K_p should increase to enhance the control action to eliminate control error quickly. Therefore, the correcting factor $\alpha_p(t)$ of the proportional gain is firstly calculated referring to step (2), then the real-time proportional gain $K_p(t)$ will be got on the basis of the equation $K_p(t) = K_p^0/\alpha_p(t)$, where, K_p^0 is the initial proportional gain of the secondary control unit. When $\alpha_p(t)$ is larger than 1.0, $K_p(t)$ will be reduced; When $\alpha_p(t)$ is smaller than 1.0, $K_p(t)$ will be increased. When $\alpha_p(t)$ equals to 1.0, $K_p(t)$ will not be changed.

The correcting algorithm of integral action is similar to that of the proportional action, but the correcting direction of integral action is inverse. And the point E_i of the correction-direction for the integral action is also different from E_p. The correcting equation is $T_i(t) = T_i^0\alpha_i(t)$, where $\alpha_i(t)$ is the correcting factor of the integral action, and T_i^0 is the initial integral time of secondary control unit.

In the similar way, the correcting algorithm of derivative action is $T_d(t) = T_d^0\alpha_d(t)$, where, T_d^0 is the initial derivative time of secondary control unit.

3.2.2 Design of the Secondary Control Unit

When the secondary controller works independently, its control law obeys the conventional PID control algorithm

$$U(t) = K_p\left[e(t) + \frac{1}{T_i}\int_0^t e(t)dt + T_d\frac{de(t)}{dt}\right]. \tag{4}$$

When the control parameters is adjusted dynamically by the primary control unit, its control law is as follows:

$$U(t) = K_p(t)\left[e(t) + \frac{1}{T_i(t)}\int_0^t e(t)dt + T_d(t)\frac{de(t)}{dt}\right], \tag{5}$$

where $K_p(t)$, $T_i(t)$, and $T_d(t)$ are the corrected control parameters of the secondary control unit.

3.2.3 The NEIC Control Parameters Tuning

(1) **Tune the initial control parameters.** First only take the secondary control unit into action, then tune the initial control parameters K_p^0, T_i^0, and T_d^0 approximately.

(2) **Determine the factors *B*$_j$.** Decide the maximal or the minimal multiple of the various action gain according to a special process, and then obtain the factors B_j of various control action approximately.

(3) **Determine the factors *A*$_j$.** Decide the point E_j, and then obtain every factor A_j.

(4) **Tune the factors *A*$_j$ and *B*$_j$.** Take the primary control unit into action, and then tune the factors A_j and B_j repeatedly until the control performance is satisfied.

4 Simulation Results

In order to examine the NEIC's control performance, we chose plant A and B for the simulation experiments. The plant A is

$$G(S) = e^{-105S}/(95S+1)$$

And plant B is

$$\dot{y}(t) = 0.7y^2(t) - y(t) + x(t)$$

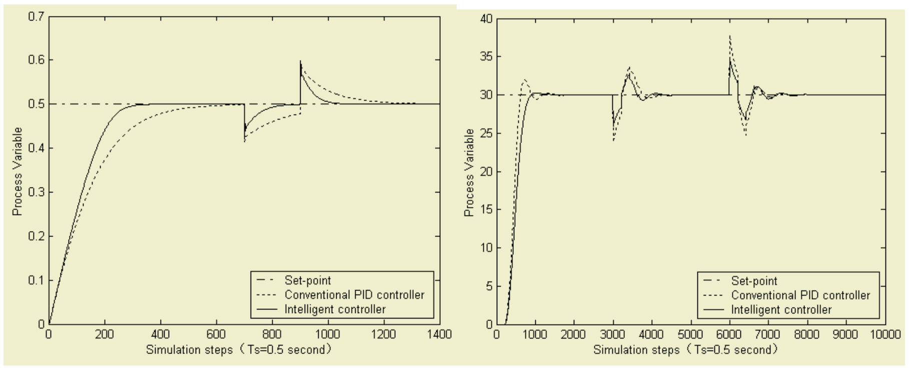
where $x(t)$ and $y(t)$ are input and output variables respectively. We also compare its control performance with that of the conventional PID controller. To make the contrast effectiveness more clearly, we let the initial PID control parameters of the secondary control unit be the same as those of the conventional PID controller. The control parameter sets are shown in Table 1. From the contrast control effectiveness in Fig. 3, we can see that the NEIC control system can become stable faster and with little or without overshoot compared with the conventional PID controller. To examine its adaptability, we change the time constant of plant A from 95 to 120 at 1500 seconds, and recover them at 3000 seconds. Similarly, change the parameters of plant B at 350 seconds: $\dot{y}(t) = 0.3y^2(t) - y(t) + x(t)$, and recover them at 450 seconds. From Fig. 3, we also can find that the NEIC control system can still become stable faster with less overshoot even after changing of system parameters. The simulation results show that the NEIC's salability and adaptability are superior to those of conventional PID controller.

5 Conclusions

Based on the modulation mechanism of neuroendocrine system and the hypothalamo-pituitary-adrenal axis model, we present a novel intelligent controller NEIC and provide a method to adjust its control parameters dynamically, as thus to improve its control performance. The simulation results demonstrate that the stability and adaptability of the NEIC are better than those of the conventional PID controller, and it provides a new and efficient method for the complicated control system.

Table 1. The control parameters

Plant	K_p	T_i	T_d	A_P	B_P	A_i	B_i	A_d	B_d
A	1.3	120	30.	0.3	0.85	0.05	0.99	0.0	0.0
B	0.5	30.	0.0	0.2	0.4	0.15	0.4	0.0	0.0



(plant A) (plant B)

Fig. 3. The contrast control effectiveness.

Acknowledgments

This work was supported in part by the National Nature Science Foundation of China (No. 60474037 and 60004006), Program for New Century Excellent Talents in University, and Specialized Research Fund for the Doctoral Program of Higher Education from Educational Committee of China (No. 20030255009).

References

1. Ding, Y.S., Ren, L.H.: Fuzzy Self-Tuning Immune Feedback Controller for Tissue Hyperthermia. IEEE Int. Conf. on Fuzzy Systems, San Antonio, **1** (2000) 534-538.
2. Kim, D.H.: PID Controller Tuning of a Boiler Control System Using Immune Algorithm Typed Neural Network. Computational Science-ICCS 2004, Krakow, Poland, (2004) 695-698.
3. Farhy, L.S.: Modeling of Oscillations of Endocrine Networks with Feedback. Methods Enzymol, **384** (2004) 54-81
4. Timmis, J., Neal M.: Artificial Homeostasis: Integrating Biologically Inspired Computing. www.cs.kent.ac.uk/pubs/2003/1586/content.pdf, July 10, (2004)
5. Keenan, D.M, Licinio J., Veldhuis, J. D.: A Feedback-Controlled Ensemble Model of the Stress-Responsive. Hypothalamo-Pituitary-Adrenal Axis, PNAS, **98** (2001) 4028-4033

Batch-to-Batch Optimal Control Based on Support Vector Regression Model

Yi Liu[1], Xianhui Yang[1], Zhihua Xiong[1], and Jie Zhang[2]

[1] Institution of Process Control Engineering, Department of Automation, Tsinghua University, Beijing 100084, China
liuyi00@mails.tsinghua.edu.cn,
{yangxh,zhxiong}@mail.tsinghua.edu.cn,
[2] Centre for Process Analytics and Control Technology, School of Chemical Engineering and Advanced Materials, University of Newcastle, Newcastle upon Tyne, NE1 7RU, UK
jie.zhang@ncl.ac.uk

Abstract. A support vector regression (SVR) model based batch to batch optimal control strategy is proposed in this paper. Because of model plant mismatches and unknown disturbances the control performance of optimal control profile calculated from empirical model is deteriorated. Due to the repetitive nature of batch processes, it is possible to improve the operation of the next batch using the information of the current and previous batch runs. A batch to batch optimal control strategy based on the linearization of the SVR model around the control profile is proposed in this paper. Applications to a simulated batch styrene polymerization reactor demonstrate that the proposed method can improve process performance from batch to batch in the presence of model plant mismatches and unknown disturbances.

1 Introduction

Batch processes are suitable for the manufacturing of high value added products such as pharmaceuticals, specialty chemicals, and biochemicals. Unlike continuous processes, batch processes are inherently transient and typically also nonlinear [1]. For the purpose of batch process optimization, mechanistic or empirical models are developed to predict the final product quality from a given control profile. The development of a detailed mechanistic model is usually time consuming and effort demanding. To overcome this difficulty, data based empirical models can be utilized. However, mismatch between the model and the actual plant often exists due to low-quality or limited operational data or variations in process conditions. Thus the optimal control profile calculated from the model may not be optimal when applied to the actual plant. Due to the repetitive nature of batch process operations, it is possible to improve the operations of the next batch using the information of the current and previous batch runs. There are many contributions on batch-to-batch optimization [2],[3].

Support vector machine (SVM) and SVR are powerful for the problems characterized by small samples, nonlinearity, high dimension, local minima [4] and have been applied to nonlinear process modeling [5] and control [6]. SVR method can be used for modeling of the relationship of the control input and the endpoint product quality.

For the purpose of batch to batch optimization, the model is linearized around the nominal control profile. And the control policy for the next batch is modified to minimize the control errors at the end of the next batch. The procedure is repeated from batch to batch.

The paper is organized as follows. In section 2 we introduce SVR and its modeling of batch processes. A SVR model based batch to batch optimal control strategy is presented in Section 3. Section 4 presents an application to a simulated batch polymerization reactor. The last section concludes this paper.

2 SVR and Modeling of Batch Processes

The SVM is derived from the statistical learning theory [4]. The formulation of SVM embodies the structural risk minimization principle, which has been shown to be superior to traditional empirical risk minimization principle employed by conventional neural networks [4]. By involving both empirical and anticipant risk in the training cost function, SVM could successfully avoid stepping into local optimal points and the structure and the parameters of SVM are relatively easy to determine. In the algorithm of SVM, the input nonlinear space is mapped into a high dimensional feature space via a nonlinear mapping, and linear classification and regression are performed in that space.

Consider a given set of N data points $\{x_n, y_n\}_{n=1}^N$ with input data $x_n \in \mathbb{R}^d$ and output $y_n \in \mathbb{R}$. In feature space SVR models take the form:

$$f(x) = w^T \varphi(x) + b\,, \tag{1}$$

where $\varphi(\cdot)$ is the nonlinear mapping. In least squares support vector regression the following optimization problem is formulated [6]:

$$\min_{w,e} J(w,e) = \frac{1}{2} w^T w + \gamma \frac{1}{2} \sum_{k=1}^{N} e_k^2\,, \quad s.t.\ y_n = w^T \varphi(x_n) + b + e_n,\ n = 1,\cdots,N. \tag{2}$$

By employing Lagrange multipliers α_n and exploiting Karush-Kuhn-Tucker condition, the solution is given b and y by the following linear equations:

$$\begin{pmatrix} 0 & \vec{1}^T \\ \vec{1} & \Omega + \gamma^{-1} I \end{pmatrix} \begin{pmatrix} b \\ \alpha \end{pmatrix} = \begin{pmatrix} 0 \\ y \end{pmatrix}, \quad \begin{aligned} \Omega_{nm} &= \varphi(x_n)^T \varphi(x_m) \\ &= \Psi(x_n, x_m),\ n,m = 1,\cdots,N \end{aligned}, \tag{3}$$

where $y = [y_1;\cdots;y_N]$, $\vec{1} = [1;\cdots;1]$, $\alpha = [\alpha_1;\cdots;\alpha_N]$ and the Mercer's condition [4] has been applied. This finally results into the following least squares SVR model for function estimation:

$$f(x) = \sum_{n=1}^{N} \alpha_n \Psi(x, x_n) + b\,\cdot \tag{4}$$

For the choice of the kernel function $\Psi(\cdot,\cdot)$ one has several possibilities: $\psi(x,x_n) = x_n{}^T x$ (linear SVR); $\psi(x,x_n) = (x_n{}^T x + 1)^d$ (polynomial SVR); $\psi(x,x_n) = \exp(-\|x - x_n\|_2^2 / 2\sigma^2)$ (RBF SVR). Here RBF kernel is adopted.

In batch processes, the ultimate interest lies in the end of batch product quality. To calculate the optimal control policy for a batch process, the model utilized in the optimization calculation should be able to provide product quality prediction. This paper proposes to use the following model to describe the endpoint product quality variables of a batch:

$$y(t_f) = f(S_0, U), \tag{5}$$

where $y(t_f)$ is the product quality variable at the endpoint time t_f, S_0 is the batch process initial condition, and $U = [u_1\, u_2 \cdots u_L]^T$ is a vector of control actions. The above nonlinear function $f(\cdot,\cdot)$ is represented using a SVR model developed using process operational data.

3 Batch-to-Batch Optimal Control

Based on the model Eq. (5), the optimal control policy U can be obtained by solving the following optimization problem:

$$\begin{aligned} &\min_U J[y(t_f)] \\ &s.t. \text{ product quality and process constraints} \end{aligned}, \tag{6}$$

However, model plant mismatches always exist, especially when the model is a data based empirical model developed from a limited amount of process operational data. In this case, the optimal control policy calculated from Eq. (6) is only optimal on the model and may not be optimal when applied to the real process. Furthermore, there always exist disturbances such as raw material variations, reactive impurities, and reactor fouling. The presence of such disturbances also deteriorates the control performance of the previously obtained optimal control policy.

To limit the deterioration of control performance due to model plant mismatches and unknown disturbances, a batch to batch optimal control strategy is proposed. This control strategy utilizes the information of the current and previous batch runs to enhance the operation of the next batch.

The first order Taylor series expansion of Eq. (5) around a nominal control profile can be expressed as

$$\hat{y}(t_f) = f_0 + \sum_{l=1}^{L} \frac{\partial f}{\partial u_l} \Delta u_l, \tag{7}$$

where Δu_1 to Δu_L are deviations in the control profile from the nominal control profile.

For the kth batch, the actual product quality can be written as the model prediction plus an error term

$$y_k(t_f) = \hat{y}_k(t_f) + e_k, \tag{8}$$

where $y_k(t_f)$ and $\hat{y}_k(t_f)$ are the actual and predicted product quality values at the end of a batch respectively, and e_k is the model prediction error.

The prediction for the (k+1)th batch can be approximated using the first order Taylor series expansion based on the kth batch:

$$\hat{y}_{k+1}(t_f) = \hat{y}_k(t_f) + \sum_{l=1}^{L} \left.\frac{\partial f}{\partial u_l}\right|_{U_k} (u_l^{k+1} - u_l^{k}) = \hat{y}_k(t_f) + G^T \Delta U^{k+1}, \tag{9}$$

where

$$\Delta U^{k+1} = \left[\Delta u_1^{k+1} \Delta u_2^{k+1} \cdots \Delta u_L^{k+1}\right]^T, G^T = \left[\left.\frac{\partial f}{\partial u_1}\right|_{U^k} \left.\frac{\partial f}{\partial u_2}\right|_{U^k} \cdots \left.\frac{\partial f}{\partial u_L}\right|_{U^k}\right]^T. \tag{10}$$

Assuming that the model prediction errors for the kth batch and the (k+1)th batch are the same, then optimal control of the (k+1)th batch can be represented as [7]

$$\min_{\Delta U^{k+1}} J = \left\|\hat{y}_k(t_f) + G^T \Delta U^{k+1} + e_k - y_d\right\|_Q^2 + \left\|\Delta U^{k+1}\right\|_R^2, \tag{11}$$

where Q is a weighting matrix for the product quality control errors and R is a weighting matrix for the control effort.

Set $\partial J / \partial \Delta U^{k+1} = 0$, the optimal control updating can be calculated as

$$\Delta U^{k+1} = (GQG^T + R)^{-1} GQ(y_d - \hat{y}_k(t_f) - e_k), \quad U^{k+1} = U^k + \Delta U^{k+1}. \tag{12}$$

For the SVR model of Eq.(4) with RBF kernel function, the gradient of model output with respect to the U and G, can be calculated analytically:

$$G = \left.\frac{\partial f}{\partial U}\right|_{U_k} = \sum_{n=1}^{N} \alpha_n \left.\frac{\partial \Psi(U, U_n)}{\partial U}\right|_{U_k} = \sum_{n=1}^{N} -\alpha_n \frac{(U_k - U_n)}{\sigma^2} \exp\left(-\frac{\|U_k - U_n\|_2^2}{2\sigma^2}\right). \tag{13}$$

4 Application to a Simulated Batch Polymerization Reactor

In this section, the algorithm is tested through simulations for the thermally initiated bulk polymerization of styrene in a batch reactor. The differential equations describing the polymerization system are given by Kwon and Evans [8]:

$$\begin{aligned}
\dot{x}_1(t) &= \frac{(r_1 + r_2 T_c)^2}{M_m}(1 - x_1)^2 \exp(2x_1 + 2\chi x_1^2)\left(\frac{1 - x_1}{r_1 + r_2 T_c} + \frac{x_1}{r_3 + r_4 T_c}\right) A_m \exp\left(-\frac{E_m}{T}\right), \\
\dot{x}_2(t) &= \frac{\dot{x}_1(t) x_2}{1 + x_1}\left(1 - \frac{1400 x_2}{A_w \exp(B/T)}\right), \quad \dot{x}_3(t) = \frac{\dot{x}_1(t)}{1 + x_1}\left(\frac{A_w \exp(B/T)}{1500} - x_3\right).
\end{aligned} \tag{14}$$

where the state variables x_1, x_2 and x_3 are, respectively, conversion, dimensionless number-average chain length (NACL) and dimensionless weight-average chain length (WACL). The nominal parameters used in the above equations are given in [8].Based on this model, a rigorous simulation program was developed and used to generate polymerization data under different batch operating conditions.

In the simulation study, the whole reaction time is set to 300 minutes. The temperature profile during a batch is taken as the control profile and is divided equally into 30 equal intervals. During each interval, the temperature is kept constant. The objective is to derive an optimal temperature profile that maximizes the conversion with values of dimensionless NACL and WACL as close to 1.0 as possible at the end of the batch.

In order to generate the operational data for building the model, a normal trajectory of the temperature is first derived which leads to satisfactory product quality. Then random changes with uniform distribution with a magnitude of ±30% are added to the normal trajectory and data for 60 batch runs are generated to develop the initial model. Base on the generated data, an SVR model is built which takes temperature T as the model input and $\mathbf{y}(t_f)$ ($\mathbf{y} = [x_1,x_2,x_3]^T$, t_f is the final time) as the model output.

With the implementation of the proposed batch to batch optimal control method, it can be seen from Fig.1(a) that the control performance is significantly improved in the 2nd batch and further improved in the 3rd batch. The quality index ($Index_k = \|y_k(t_f) - y_d\|_Q^2$) converges gradually afterwards. This demonstrates that the proposed batch to batch optimization technique can effectively overcome the problem of model plant mismatches. Fig.1(b) shows the final product quality variable variations from batch to batch.

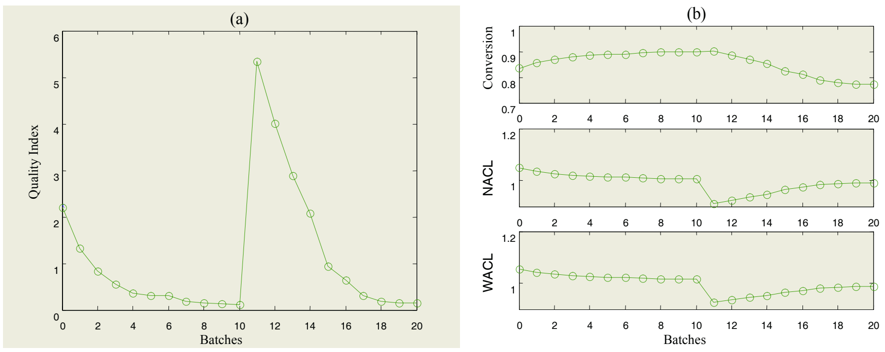


Fig. 1. Batch-to-batch optimal control performance: (a) the quality index, (b) the outputs

To test the performance of the proposed control strategy under unknown disturbances, parameter A_w is changed to $A_{w,\text{ plant}}=1.2A_w$ from the 11th batch forward. The final quality variables are far away from the desired values at the 11th batch as show in Fig.1(a). From the 12th batch, the recalculated optimal temperature profiles were implemented to the plant. It can be seen that the final quality become better gradually from batch to batch. After the 16th batches, the batch to batch optimal control almost converged to the satisfying quality. The results demonstrate that the batch to batch iterative optimal control strategy can effectively overcome the problems caused by the presence of unknown disturbances.

5 Conclusions

An SVR model based batch to batch iterative optimal control strategy is proposed in this paper. To avoid the difficulty in developing detailed mechanistic models, SVR methods are used to model batch processes from process operational data. Due to model plant mismatches and the presence of unknown disturbances, an optimal control policy calculated based on the SVR model may not give optimal performance when applied to the real processes. The repetitive nature of batch processes allows the information of previous batch runs to be used to update the control profile of the current batch. Linearization of the SVR model is used in calculating the optimal control profile updating. Applications to a simulated batch polymerization process demonstrate that the proposed method can effectively overcome the problems of model plant mismatches and unknown disturbances through batch to batch updating of control profile.

Acknowledgements

The work was supported by National Science Foundation of China (NSFC 60404012) UK EPSRC (GR/R10875).

References

1. Benson, R.: Process Control – the Future. Computing and Control Engineering Journal, **8** (1997) 161–166
2. Flores-Cerrillo, J., MacGregor, J.F.: Within-Batch and Batch-to-Batch Inferential-Adaptive Control of Semibatch Reactors: A Partial Least Squares Approach. Ind. Eng. Chem. Res, **42** (2003) 3334-3345
3. Xiong, Z.H., Zhang, J.: Product Quality Trajectory Tracking in Batch Processes Using Iterative Learning Control Based on Time-Varying Perturbation Models. Ind. Eng. Chem. Res., **42** (2003) 6802-6814
4. Vapnik, V.: The Nature of Statistical Learning Theory. Springer-Verlag, New-York (1995)
5. Smola, A., Scholkopf, B.: A Tutorial on Support Vector Regression. Technical Report NC2-TR-1998-030, NeuroCOLT2 (1998)
6. Suykens, J.A.K., Vandewalle, J., De Moor, B.: Optimal Control by Least Squares Support Vector Machines. Neural Networks, **14** (2001) 23-35
7. Zhang, J.: Neural Network Model Based Batch-to-Batch Optimal Control. Proceedings of the 2003 IEEE International Symposium on Intelligent Control, Houston, Texas (2003)
8. Kwon, Y.D., Evans, L.B.: A Coordinate Transformation Method for the Numerical Solution of Nonlinear Minimum-Time Control Problems. AIChE J., **21** (1975) 1158-1164

Nonlinear Predictive Control Based on Wavelet Neural Network Applied to Polypropylene Process

Xiaohua Xia, Zhiyan Luan, Dexian Huang, and Yihui Jin

Process Control Lab, Automation Department, Tsinghua University,
Beijing 100084, China
xiaxh99@mails.tsinghua.edu.cn

Abstract. A nonlinear adaptive predictive control strategy using orthogonal wavelet network model is presented. Based on a set of orthogonal wavelet functions, wavelet neural network performs a nonlinear mapping from the network input space to the wavelons output space in hidden layer. Its weight coefficients can be simply estimated by a linear least-square estimation algorithm. The excellent statistic properties of the weight parameters of wavelet network also can be obtained. A single input single output (SISO) nonlinear predictive control strategy is implemented in the simulation of a Polypropylene process.

1 Introduction

During the past twenty years, model predictive control (MPC) based on linear process models have been widely studied and applied in chemical process industry. However, there are cases where nonlinear effects are too significant for MPC to implement. So MPC techniques have been extended to nonlinear processes during the last decade.

Recently, neural networks have become a popular tool in non-parametric function learning due to their ability to learn complicated functions. It has been proven that multilayer back propagation networks can approximate any continuous function [1].

Wavelet is also a powerful tool for function approximation [2]. The idea of using wavelets in neural network has also been proposed recently by Zhang [3]. Under some mild conditions, it has been shown that the wavelet network has universal and L^2 approximation properties and is a consistent function estimator [3]. Based on a set of orthogonal wavelet functions, a least-square learning algorithm is adopted to train the wavelet network in contrast to the nonlinear gradient optimization used in standard feed forward networks [4]. The control performances superior to a standard PID controller were achieved [5],[6].

In this paper, a nonlinear adaptive predictive control strategy using orthogonal wavelet network model is presented. Since almost all dynamic processes in the chemical industry are low-pass systems, they can be approximated only by scale function terms at any accuracy. Therefore, only scale function is used in wavelons. This will simplify wavelet network and decrease network size in online training obviously. The output layer is linear structure. Its weight coefficients can be estimated by a linear least-square estimation algorithm. The excellent statistic property of the weight parameters of wavelet network as linear least-square estimation algorithm in system identification can be proved.

A recursive algorithm for online application in adaptive predictive control strategy is given in this paper, and the recursive algorithm is completely same as recursive linear least-square algorithm.

With the developed recursive algorithm, a SISO nonlinear adaptive predictive control strategy is implemented for the simulation of temperature control of an exothermic polypropylene process. The nonlinear adaptive predictive control strategy based on wavelet network is superior to the standard PID controller.

2 Approximation Principle of Wavelet Neural Networks

Multiresolution approximation by wavelet basis functions is a technique for representing a function on many different scales. The wavelets are functions whose dilations and translations form an orthonormal basis of L^2(R). Specifically, there exists a function of $\psi(x)$ (the "mother wavelet"), here $\psi_{m,n}(x)=2^{-m/2}\psi\left(2^{-n}x-n\right)$. $\psi_{m,n}(x)$ forms an orthonormal basis of L^2(R).

The wavelet $\psi(x)$ is often generated from a companion $\varphi(x)$, known as the scaling function (the "father wavelet"), through the follow "dilation" equations:

$$\varphi(x/2)=\sqrt{2}\sum_k h_k\varphi(x-k) \quad \psi(x/2)=\sqrt{2}\sum_k g_k\varphi(x-k). \tag{1}$$

Where $\{h_k\}$ and $\{g_k\}$ are a pair of discrete quadrature mirror filter.

The dilations and translations of scaling function induce a multiresolution analysis (MRA) of L^2(R), i.e., a nested chain of closed subspaces:

$\cdots\subset V_2\subset V_1\subset V_0\subset V_{-1}\subset V_{-2}\subset\cdots$ such that $\bigcap_m V_m=\{0\}$, $colse\left\{\bigcup_m V_m\right\}=L^2(R)$

Where V_m is a subspace spanned by $\left\{2^{-m/2}\varphi\left(2^{-m}x-n\right)\right\}_{n=-\infty}^{n=+\infty}$ [7]. Furthermore, V_m and W_m are related by $V_m=V_{m+1}\oplus W_{m+1}$.

The above discussions suggest the scheme for decomposing a L^2(R) function f(x),

$$f(x)=\sum_{m,n}\left\langle f,\psi_{m,n}\right\rangle\psi_{m,n}(x). \tag{2}$$

where m_0 is an arbitrary integer and represents the lowest resolution or scale.

The structure of a wavelet neural network is similar to that of an RBF network. However its structure can be decided using wavelet frames. Because only scale function is used, then

Consider a SISO nonlinear dynamic system denoted by the following equation:

$$Y(k)=f\left(Y(k-1),\cdots Y\left(k-n_y\right),U(k-1-\tau),\cdots U\left(k-n_u-\tau\right)\right). \tag{3}$$

where τ is the model input output time delay, $(Y,U)\to f(Y,U): R\times R\to R$. The network structure proposed by Narendra is adopted [8].

Firstly, we denote the optimum values of all $\left\langle f,\varphi_{M,n}\right\rangle$ as θ and all $\varphi_{M,n}(x)$ values in time k as h(k). Then

$$Y(k)=h^T(k)\theta+n(k). \tag{4}$$

where both vectors Y(k) and n(k) are one dimensional , the dimension of h(k) is N and that of θ is N. N is the number of hidden neurons.

For $k=1,2\cdots L$, the above equation constructs a linear equation group. It can be expressed in matrix form as following.

$$Y_L = H_L\theta + n_L . \tag{5}$$

where

$$\begin{gathered} Y_L = [Y(1), Y(2), \cdots, Y(L)]^T \; n_L = [n(1), n(2), \cdots, n(L)]^T , \\ H_L = [H(1), H(2), \cdots, H(L)]^T . \end{gathered} \tag{6}$$

By using linear least-square estimation, we can obtain the estimates of the weight parameters of the wavelet network as:

$$\hat{\theta}_{LS} = (H_L^T H_L)^{-1} H_L^T Y_L . \tag{7}$$

For online application in adaptive predictive control strategy, a recursive LS algorithm with exponential forgetting algorithm is as following.

$$\begin{aligned} &\hat{\theta}(t) = \hat{\theta}(t-1) + a(t)K(t)e(t) \quad e(t) = y(t) - \varphi^T(t)\hat{\theta}(t-1) \\ &K(t) = \frac{P(t-1)\varphi(t)}{1+\varphi^T(t)P(t-1)\varphi(t)} \end{aligned} . \tag{8}$$

$$P(t) = \frac{P(t-1)}{\lambda} - \alpha\frac{P(t-1)\varphi(t)\varphi^T(t)P(t-1)}{I+\varphi^T(t)P(t-1)\varphi(t)} + \beta I - \delta P(t-1)^2 . \tag{9}$$

$$K(t) = \frac{P(t-1)\varphi(t)}{\alpha + \varphi^T(t)P(t-1)\varphi(t)} . \tag{10}$$

$$P(t) = \frac{1}{\alpha}[P(t-1) - \frac{P(t-1)\varphi(t)\varphi^T(t)P(t-1)}{\alpha+\varphi^T(t)P(t-1)\varphi(t)}] . \tag{11}$$

3 Statistic Property of the Identification Algorithm

With the solid theory basis and special structure of wavelet neural network, wavelet neural network holds some advantages superior to other type neural networks. Firstly, it is easy to specify the network structure. Secondly, network training does not rely on stochastic gradient type techniques such as the “back propagation” and so avoids the problem of poor convergence or undesired local minimum.

We prove that the excellent statistic properties of the weight parameters of this wavelet network as linear least-square estimation algorithm in system identification can be obtained.

Firstly, we assume that noise sequence $\{n(k)\}$ is white noise, i.e.

$$E\{n_L\}=0 \quad Cov\{n_L\}=\sigma_n^2 I . \tag{12}$$

Then, we can prove the statistic property of the weight parameter estimation of wavelet network is type of linear LS.

Theorem 1. If the noise sequence n_L in (7) satisfies the conditions in (14), then the estimates of the weight parameters of a wavelet network are unbiased, i.e.,

$$E\left\{\hat{\theta}_{LS}\right\}=\theta . \tag{13}$$

Theorem 2. If the noise sequence n_L in (7) satisfies the conditions in (14), then the covariance matrix is

$$Cov\left\{\hat{\theta}_{LS}\right\}=\sigma_n^2 E\left\{(H_L^\tau H_L)^{-1}\right\}. \tag{14}$$

4 Adaptive Predictive Control Based on Wavelet Network

Model predictive control is widely accepted, primarily due to its ability in real-time prediction, optimisation and feedback correction.

In the nonlinear adaptive predictive control scheme shown in Fig. 1, a process model, i.e., a wavelet network, is explicitly used to predict future process behaviour. The same process model is also implicitly adopted to calculate control actions in such a way as to optimise the controller specifications at each sampling step. Furthermore, the difference between the current predicted output and the measured current process output is used to correct the model error and disturbances so as to improve its robustness. While predictive control is processed, the process model is updated by online recursive identification algorithm to enhance its robustness exteriorly.

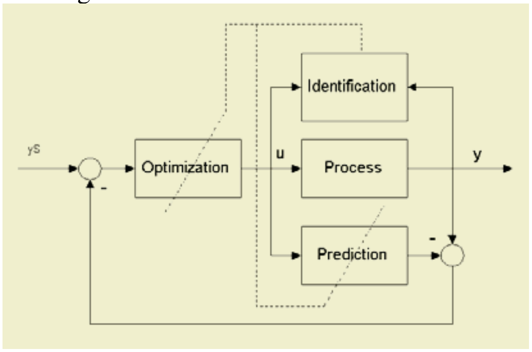


Fig. 1. Adaptive Predictive Control Scheme Based on Wavelet Network

Consider SISO nonlinear dynamic system denoted by (4). The selection of the control law is based on a quadratic performance index with a finite time horizon, resulting in the following quadratic programming (QP) problem at time k

$$\min_{\Delta u(k),\Delta u(k+1),\cdots,\Delta u(k+L-1)} J(k) \tag{15}$$

$$\begin{aligned} J(k) = & \sum_{i=1}^{P} \left\| Y^S(k+i)-\hat{Y}(k+i|k)-Y(k)+\hat{Y}(k|k-P-1)\right\|_Q^2 \\ & + \sum_{j=1}^{L} \left\| \Delta u(k+j-1|k)\right\|_R^2 \end{aligned} \tag{16}$$

where P is prediction horizon, L is control horizon, Q and R are weighting matrices.

The process model parameters, weight parameters of wavelet network, are updated by recursive identification algorithm with forgetting factor in (10), (11) and (12).

A simulated exothermic polypropylene process illustrates the application of this control scheme. The control object is the temperature of the reactor and the manipulate variable is the cooling or heating water. Control result is shown in Fig. 2. The nonlinear adaptive predictive control strategy based on wavelet network is superior to the standard PID controller whose control result.

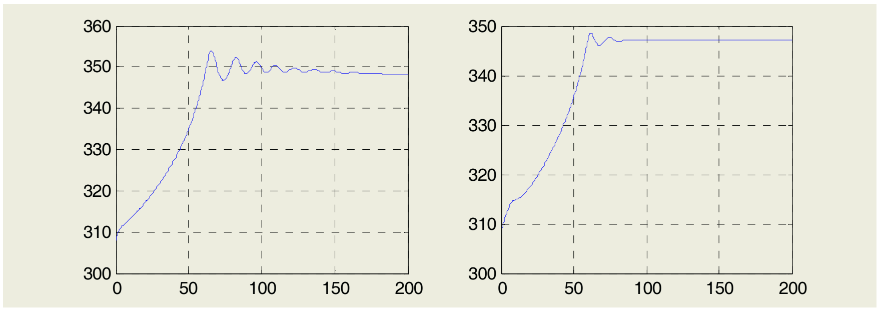


Fig. 2. Effects of PID Controller and Adaptive Predictive Control Scheme Based on Wavelet Network

5 Conclusions

In this paper, a nonlinear adaptive predictive control strategy based on orthogonal wavelet network model is realized. Wavelet network model only with scale function simplified wavelet network and decreased network size in online training obviously. Its weight coefficients can be estimated by a linear least-square estimation algorithm.

A recursive algorithm for online application in adaptive predictive control strategy is given in this paper, and the recursive algorithm is completely same as recursive linear least-square algorithm. In addition, with the adopting of orthogonal wavelet functions, the different wavelon outputs in hidden layer are irrelevant each other. The property that is similar to the recursive linear least-square algorithm can be obtained. So the closed-loop identifiability can be guaranteed.

With the developed recursive algorithm, a SISO nonlinear adaptive predictive control strategy is implemented in this paper. The simulation of a Polypropylene process illustrates the control scheme. Simulation results show the good performances. Online identification algorithm can track the parameter changing rapidly as the parameters of controlled system change. The nonlinear adaptive predictive control strategy based on wavelet network is superior to the standard PID controller.

References

1. Hornik, K., Stinchcombe, M., White, H.: Multilayer Feed-forward Networks Are Universal Approximators. Neural Networks, **2** (1989) 359–366
2. Daubechies, I.: Orthonomal Bases of Compactly Supported Wavelets. Comm. Pure. Applied. Math, **91** (1988) 909–996

3. Zhang, J., Walter, G.: Wavelet: Neural Networks for Function Learning. IEEE Transactions on Signal Processing, **43** (1995) 1485–1497
4. Bakshi, B.R., Stephanopoulos, G.: Wavelet: a Multiresolution, Hierarchical Neural network with Localised Learning. AIChE Journal, **39** (1993) 57–81
5. Huang, D., Jin, Y.: The Application of Wavelet Neural Networks to Nonlinear Predictive Control. ICNN97 IEEE Houston Texas US, **2** (1997)
6. Huang, D., Wang, J., Jin, Y.: Application Research of Wavelet Neural Networks in Process Predictive Control. Journal of Tsinghua University. 39 (1999) 91-94
7. Mallat, S.G.: A Theory of Multiresolution signal Decomposition: Wavelets Transform. IEEE Trans. PAMI-11, **7** (1989) 674–693
8. Narendra, K.S., Parthasarathy, K.: Identification and Control of Dynamic Systems Using Neural Networks. IEEE Transactions on Neural Network, **1** (1990) 4-27

Neural Network Control of Heat Exchanger Plant

Mahdi Jalili-Kharaajoo

Young Researchers Club, Islamic Azad University, Tehran, Iran
mahdijalili@ece.ut.ac.ir

Abstract. In this paper, a neural network based controller is applied to a heat exchanger pilot plant. The dynamics of the plant is identified using an MLP neural network. Then, the predictive control strategy based on the neural network model of the plant is applied to provide set point tracking of the output of the plant. Also, the performance of the proposed controller is compared with that of Generalized Predictive Control (GPC) through simulation studies. Obtained results demonstrate the effectiveness and superiority of the proposed approach.

1 Introduction

Predictive control is now widely used in industry and a large number of implementation algorithms [1]. Although industrial processes usually contain complex nonlinearities, most of the MPC algorithms are based on a linear model of the process. Linear models such as step response and impulse response models derived from the convolution integral are preferred, because they can be identified in a straightforward manner from process test data. In addition, the goal for most of the applications is to maintain the system at a desired steady state, rather than moving rapidly between different operating points, so a precisely identified linear model is sufficiently accurate in the neighborhood of a single operating point.

Recently, neural networks have become an attractive tool in the construction of models for complex nonlinear systems [2]. A large number of control and identifications structures based on neural networks have been proposed [3],[4]. Most of the nonlinear predictive control algorithms imply the minimization of a cost function, by using computational methods for obtaining the optimal command to be applied to the process. The implementation of the nonlinear predictive control algorithms becomes very difficult for real-time control because the minimization algorithm must converge at least to a sub-optimal solution and the operations involved must be completed in a very short time (corresponding to the sampling period). This paper analyzes an artificial neural network based nonlinear predictive controller for a heat exchanger, which is a highly nonlinear process [5]. The procedure is based on construction of a neural network model for the process and the proper use of that in the optimization process. The method eliminates the most significant obstacles for nonlinear MPC implementation by developing a nonlinear model, designing a neural predictor and providing a rapid, reliable solution for the control algorithm. Using the proposed controller, the output temperature tracking behavior of the plant is studied.

2 Neural Network Based Prediction and Control

2.1 Modeling of Nonlinear Systems Using Neural Networks

The use of neural networks for nonlinear process modeling and identification is justified by their capacity to approximate the dynamics of nonlinear systems including those with high nonlinearities or dead time [6],[7]. In order to estimate the nonlinear process, the neural network must be trained until the optimal values of the weight vectors (i.e. weights and biases in a vector form organization) are found. In most applications, feedforward neural networks are used, because the training algorithms are less complicated. When it comes to non-linear models, the most general one, which includes the largest class of non-linear processes, is doubtless the NARMAX model [2] given by

$$y(k)=F[u(k-d-1),...u(k-d-m),y(k-1),...y(k-n)] \tag{1}$$

where *F(·)* is a nonlinear function, *d* is the dead time, *n* and *m* are the orders of the nonlinear system model. A neural network-based model corresponding to the NARMAX model may be obtained by adjusting the weights of multi-layer perceptron architecture with adequately delayed inputs. In this case, the neural network output will be given by

$$y(k) = F^N[U(k-d-1), Y(k-1)] \tag{2}$$

where F^N denotes the input–output transfer function of the neural network which replaces the nonlinear function *F* in (1), and $U(k-d-1)$, $Y(K-1)$ are vectors which contain *m*, respectively, *n* delayed elements of *u* and *y* starting from the time instant *k-1*, i.e.

$$U(k-d-1) = [u(k-d-1),...,u(k-d-m)]^T \tag{3}$$

$$Y(k-1) = [y(k-1),..., y(k-n)]^T \tag{4}$$

The neural NARMAX corresponds to a recurrent neural network, because some of the network inputs are past values of the network output.

2.2 Neural Network Based Predictors

The predictors are necessary for the prediction of future values of the plant output that are considered in the predictive control strategy. The implementation approach of this paper uses neural predictors obtained by appropriately shifting the inputs of the neural based model. The predictive control algorithm utilizes them in order to calculate the future control signal. Neural predictors rely on the neural-based model of the process. In order to obtain the model of the nonlinear system, a neural network with a hidden layer is considered. A sequential algorithm based on the knowledge of current values of *u* and *y* together with the neural network system model gives the *i*-step ahead neural predictor. In this case one can properly derive the network output at the *k+1* time instant:

$$y(k+1)=\sum_{j=1}^{n} w_j\sigma_j\left(w_j^u U(k-d)+w_j^y Y(k)+b_j\right)+b \tag{5}$$

where N is the number of neurons in the hidden layer, σ_j is the activation function for the j^{th} neuron from the hidden layer, w_j^u is the weight vector (row vector) for the j^{th} neuron with respect to the inputs stored in $U(k-d-1)$, w_j^y *is* the weight vector (row vector) for the j^{th} neuron with respect to the inputs stored in $Y(k-1)$, b_j is the bias for the j^{th} neuron from the hidden layer, and w_j is the weight for the output layer corresponding to the *j*th neuron from the hidden layer, and *b* the bias for the output layer.
Extending (5) one step further ahead, *y(k+2)* can be obtained and generally, the *i*-step ahead predictor can be derived:

$$y(k+i) = \sum_{j=1}^{n} w_j \sigma_j \left(w_j^u U(k-d+i-1) + w_j^y Y(k+i-1) + b_j \right) + b \tag{6}$$

where

$$U(k-d+i-1) = \left[u(k-d+i-1),\dots,u(k-d+i-m) \right]^T \tag{7}$$

$$Y(k+i-1) = \left[y(k+i-1),\dots,y(k+i-n) \right]^T \tag{8}$$

The neural predictors will be used by the predictive control algorithm for calculating the *future control signal* to be applied to the non-linear system.

2.3 Predictive Control Design

The objective of the predictive control strategy using neural predictors is twofold: *(i)* to estimate the *future output* of the plant and *(ii)* to minimize a *cost function* based on the error between the predicted output of the processes and the reference trajectory. The cost function, which may be different from case to case, is minimized in order to obtain the optimum control input that is applied to the nonlinear plant. In most of the predictive control algorithms a quadratic form is utilized for the cost function:

$$J = \sum_{i=N_1}^{N_2} \left[y(k+i) - r(k+i) \right]^2 + \lambda \sum_{i=1}^{N_u} \Delta u^2 (k+i-1) \tag{9}$$

with the following requirements

$$\Delta u(k+i-1) = 0 \qquad 1 \le N_u < i \le N_2 \tag{10}$$

where N_u is the control horizon, N_1 and N_2 are the minimum and maximum prediction horizons respectively, *i* is the order of the predictor, *r* is the reference trajectory, *is* the weight factor, and Δ is the differentiation operator.
The command *u* may be subject to amplitude constraints:

$$u_{\min} \le u(k+i) \le u_{\max} \qquad i = 1,2,\dots,N_u \tag{11}$$

The cost function is often used with the weight factor =0. A very important parameter in the predictive control strategy is the control horizon N_u, which specifies the instant time, since when the output of the controller should be kept at a constant value. The output sequence of the optimal controller is obtained over the prediction horizon by minimizing the cost function J with respect to the vector U. This can be achieved by setting

$$\frac{\partial J}{\partial U}=0 \qquad U=\left[u(k-d),\ldots,u(k-d+N_u-1)\right]^T \tag{12}$$

However, when proceeding further with the calculation of $\partial J/\partial U$, a major inconvenience occurs. The *analytical approach* to the optimization problem needs for the differentiation of the cost function and, finally, leads to a nonlinear algebraic equation; unfortunately this equation cannot be solved by any analytic procedure. This is why a *computational method* is preferred for the minimization of the cost function, also complying with the typical requirements of the real-time implementations (guaranteed convergence, at least to a sub-optimal solution, within a given time interval).

For the minimization of the cost function, the Matlab's Optimal Toolbox functions *fminunc* and *fmincon* were used, which allow dealing with either unconstrained or constrained optimization problems. Unlike the *fminunc* function, *fmincon* allows imposing constraints with respect to the value of the control input such as upper or lower bounds, which are often required in practice. The cost function J is given as an input parameter for the functions mentioned above together with the initial values for the control input vector and some options regarding the minimization method (the maximum number of iterations, the minimum error value, the use of analytical gradients, etc.). In the case of *fmincon* the constraints must also be specified as input parameters in a matrix form.

The advantage of this nonlinear neural predictive controller consists in the implementation method that solves the key problems of the nonlinear MPC. The implementation is robust, easy to use and fulfills the requirements imposed for the minimization algorithm. Changes in the parameters of the neural predictive controller (such as the prediction horizons, the control horizon, as well as the necessary constraints) are straightforward operations. A simple block diagram of predictive control strategy is depicted in Fig. 1.

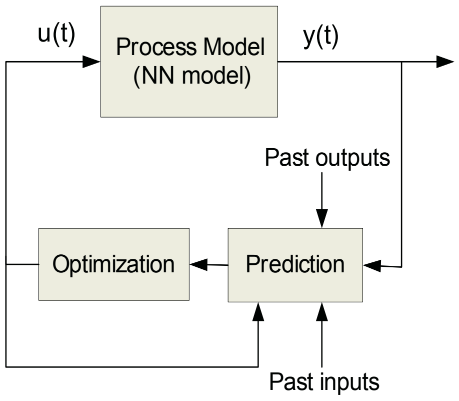


Fig. 1. The scheme of neural network based predictive control

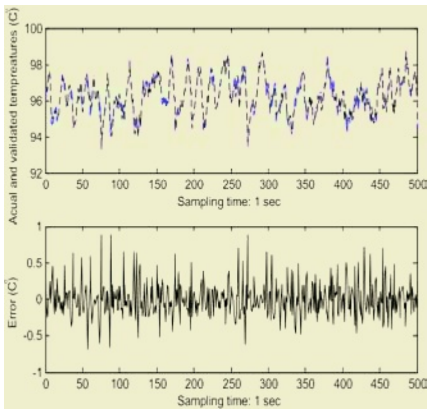


Fig. 2. Model validation: actual output and the output of the neural network model for test data

3 Simulation Results

In order to construct a neural network model for the heat exchanger, the input-output data of the plant is considered [5], where the output variable is the outlet liquid temperature and the input variables is the liquid flow rate. In this experiment the steam temperature and the inlet liquid temperature are kept constant to their nominal values. A multilayer perceptron neural network with *10* neurons in the hidden layer and $d = 0; m = 3, n = 2$ is used0. The output of the model for the test data is compared with the actual output in Fig. 2. As it is seen, the error is acceptable, so the model can be used for the objective of predictive control.

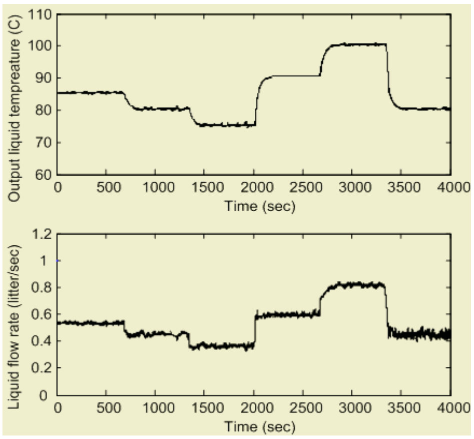


Fig. 3. Response of the plant with proper control signal for tracking the desired set points

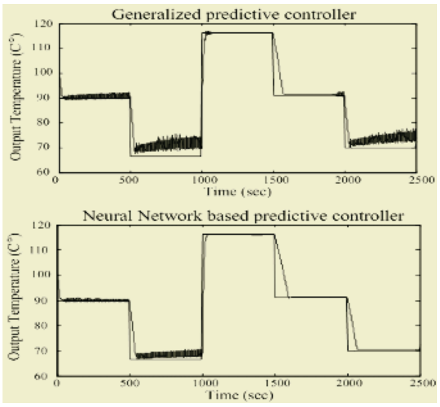


Fig. 4. Response of the plant for tracking the desired set points using neural network based controller and GPC

The *optimization problem* was addressed in accordance with the computational scenario built in the above. With respect to the notations introduced in the above, the following concrete values were chosen for the tuning parameters of the predictive control algorithm: $N_1 = 1, N_2 = 10, N_u = 5$.

The minimization algorithm gives the control input vector *U=[u(k), u(k+1), u(k+2), u(k+3), u(k+4)]*^{T} to be applied to the plant described by (3). The set point tracking results of the simulation on the plant and the corresponding input signal are depicted in Fig. 3. Clearly the system could track the set points with satisfactory settling time.

In order to investigate the effectiveness of the neural network based predictive controller, we will compare the performance of that with that of GPC controller. It is remarkable that the controllers are applied to a process which is modeled by neural network. The comparison is depicted in Figure 4. As it can be seen, the closed loop system with neural network based control action performs much better than the other one and the output temperature can track the set point values better.

4 Conclusions

In this paper, a neural network based predictive control strategy was applied to a heat exchanger pilot plant. Using the neuro predictive controller, the outlet liquid temperature of the plant tracked the desired set points by applying the liquid flow rate as a control signal. Simulation results showed the effectiveness of the proposed controller.

References

1. Camacho, E.F.: Model predictive control. Springer Verlag (1998)
2. Nelles, O.: Nonlinear System Identification: From Classical Approach to Neuro-Fuzzy Identification, Springer Verlag (2001)
3. Lennox, B., and Montague, G.: Neural Network Control of a Gasoline Engine with Rapid Sampling. In Nonlinear predictive Control Theory and Practice, Kouvaritakis, B, Cannon, M (Eds.), IEE Control Series (2001) 245-255
4. Zamarrano, J.M., Vega, P.: Neural Predictive Control. Application to a Highly Nonlinear System. Engineering Application of Artificial Intelligence, **12** (1999) 149-158
5. Bittanti, S. and Piroddi, L.: Nonlinear Identification and Control of a Heat Exchanger: A Neural Network Approach. Journal of the Franklin Institute (1996)
6. Schenker, B., and Agarwal, M.: Predictive Control of a Bench Scale Chemical Reactor Based on Neural-Network Models. IEEE Transactions on Control Systems Technology, **6** (1998) 388–400
7. Zeng, G.M., Qin, X.S., He, L., Huang, G.H., Liu, H.L. and Lin, Y.P.: A Neural Network Predictive Control System for Paper Mill Wastewater Treatment Engineering Applications Of Artificial Intelligence, **16** (2003) 121–129

Remote Controller Design of Networked Control Systems Based on Self-constructing Fuzzy Neural Network*

Yi Li, Qinke Peng, and Baosheng Hu

Systems Engineering Institute, Xi'an Jiaotong University,
Xi'an, Shaanxi 710049, China
liyi-hy@vip.sina.com

Abstract. A self-constructing fuzzy neural network (SCFNN) is proposed in this paper to design remote controller in networked control systems (NCSs). The structure and parameter learning phases are preformed concurrently in the SCFNN. The structure learning is used to obtain a proper fuzzy partition of input space, while the parameter learning is used to adjust parameters of the membership function and weights of the consequent part of the fuzzy rules based on the supervised gradient descent method. The initial SCFNN consists of input and output nodes only. In the learning process the nodes of the middle layers, which correspond to the membership functions and the fuzzy rules, are created gradually, so a set of fuzzy rules is achieved dynamically. Numerical results on a test system using Profibus-DP network are presented and compared with results of the modified Ziegler-Nichols method. The results show the effectiveness of SCFNN in designing remote controller for NCSs without any prior knowledge on network-induced delay.

1 Introduction

Fuzzy reasoning (FR) could handle uncertain information, while neural network (NN) could learn from process. Fuzzy neural networks (FNNs) integrate the merits of FR and NN. Most traditional FNNs consist of only parameters learning in which only the parameters of the membership functions (MFs) in fuzzy sets and the connected weights in NNs could be adjusted. A self-constructing fuzzy neural network (SCFNN) is proposed in [1],[2]. In SCFNN, structure learning and parameters learning can be performed concurrently, so it could create new MF and new fuzzy rule. In this study, a remote controller is designed using a SCFNN to satisfy the system specification for the networked control system (NCS).

2 Networked Control System

NCS is a feedback control system in which the control loop is closed through a real-time network. In NCS, information (reference input, plant output, control input, etc.) is exchanged through network among components (sensors, controllers, actuators) of the control system. The primary advantages are reduced system wiring, increased system agility, ease of system installation, diagnosis and maintenance [3].

* Imbursed by National Natural Science Fund of China (No.60175015 and 60373017)

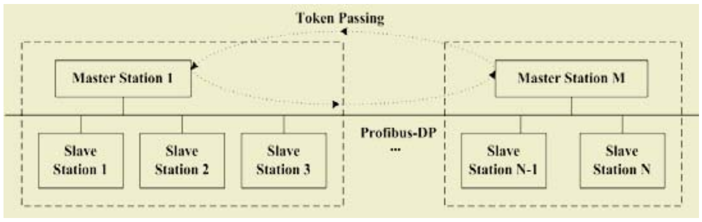


Fig. 1. The structure of Profibus-DP

In NCSs, the network-induced delay (sensor to controller delay and controller to actuator delay) occurs while exchanging data among components connected to the network. The delay would degrade the performance of control systems without considering the networked-induced delay, and even destabilize these systems. If the network is not reliable, some packets will be delayed or lost during transmissions. In addition, due to the bandwidth and packet size constraints of the network, data have to be transmitted with multiple packets (so called multiple-packet transmission). In the case of multiple-packet transmission, the packets might arrive in the wrong order, even worse part/none packets could arrive at the destination nodes when network suffers from disconnection or congestion.

In this paper, the network used in NCS is Profibus-DP, as shown in Fig.1. The MAC protocol of data link layer of Profibus-DP uses the token bus protocol and the master-slave mode. One or more master stations can be used on Profibus-DP. The token is passed on among the master stations. The master station which holds the token dominates the bus, thus this master station can communicate with its own slave stations or the other master stations.

3 Self-constructing Fuzzy Neural Network

3.1 Structure of SCFNN

Fuzzy logic rules with constant consequent part are adopted in the SCFNN. The l th rule R_l is as follows:

$$R_l: \text{IF } x_1 \text{ is } A_1^{S(l,1)} \text{ and} \ldots \text{and } x_n \text{ is } A_n^{S(l,n)}, \text{ then } y = b_l \quad l = 1,\cdots,M \tag{1}$$

where y is output variable; $\vec{x} = [x_1 \quad \cdots \quad x_n] \in R^n$ is input vector; $A_i^{S(l,i)}$ is the linguistic term of the precondition part in the l th rule, which corresponds to the i th input variable x_i; $j = S(l,i)$ is the serial number of this linguistic term in the set of linguistic terms on the universe of discourse of x_i; the MF of $A_i^{S(l,i)}$ (that is A_i^j) is $\mu_{A_{i,j}}(x_i)$; b_l is the constant consequent part; M is the number of fuzzy rules.

The structure of the SCFNN is shown in Fig. 2. The functions of the nodes in each layer are described as follows:

Each node in layer 1 is an input node corresponding to a diverse input variable. These nodes only pass on the input data to the nodes in the next layer. In this study, the input variables are $x_1 = e$ (the error) and $x_2 = \Delta e$ (the increment of error), respectively. The output error is defined as $e(t) \equiv (o(t) - r(t)) / r(t)$, where $o(t)$ and $r(t)$ are the actual and desired output of the plant, respectively.

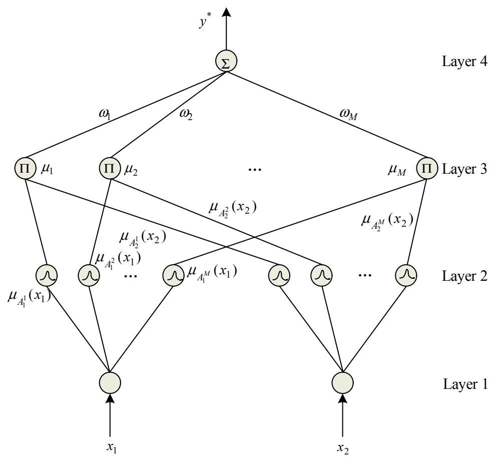


Fig. 2. Schematic diagram of SCFNN

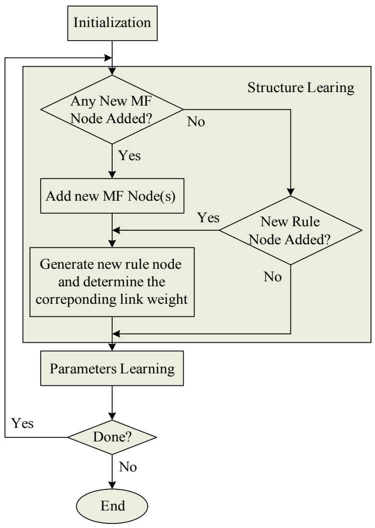


Fig. 3. Learning algorithm of SCFNN

Each node in layer 2 calculates the membership value which means the degree of an input value belonging to a fuzzy set in its universe of discourse. The MF is adopted as follows:

$$\mu_{i,j}(x_i) \equiv \mu_{A_i^j}(x_i) = \exp\{-[(x_i - m_{ji})/\sigma_{ji}]^2\} \tag{2}$$

where m_{ji} and σ_{ji} are the mean and standard deviation of MF of A_i^j, respectively.

Each node in layer 3 represents the precondition part of a fuzzy logic rule. For the l th node in this layer, it multiply all its incoming signals:

$$\psi_l(x) = \mu_{1,S(l,1)}(x_1)\cdots\mu_{n,S(l,n)}(x_n) = \prod_{i=1}^{n}\mu_{i,S(l,i)}(x_i) \tag{3}$$

where ψ_l is the firing strength of the l th rule.

The layer 4 acts a defuzzifier. The single output node in this layer sums all incoming signals to obtain the final inferred result

$$y^* = \sum_{l=1}^{M}\omega_l\psi_l \tag{4}$$

where y^* is the output of the SCFNN; ω_l is the link weight between the output node and the l th rule node, which represents the consequent part of the l th rule, so $\omega_l = b_l$.

3.2 Learning Algorithm for SCFNN

There are two types of learning algorithms, the structure learning and the parameter learning, in the construction of SCFNN. The structure learning is used to find proper fuzzy partitions of the input space and construct the fuzzy logic rule. The parameter learning uses the supervised learning algorithms to adjust the parameters of MF and the link weights between the rule nodes and output node. Initially, SCFNN only consists of input and output nodes. The MF nodes and the rule nodes are generated gradually in the learning process.

Structure Learning Phase. In this structure learning round, first make a decision whether new MF nodes need to be generated in SCFNN. For the new incoming data of the i th input variable, calculate its membership value of A_i^j, that is $\mu_{i,j}(x_i)$. Let

$$M(i,x_i) = \arg\max_{1\le j\le P_i} \mu_{A_i^j}(x_i)\,,\ i=1,\cdots,n$$

that is $\mu_{i,M(i,x_i)} = \max\limits_{1\le j\le P_i} \mu_{i,j}$,where P_i is the number of all the linguistic terms on the universe of discourse of the i th input variable. If $\mu_{i,M(i,x_i)} \le C_i$ then a new MF node is generated for the i th input variable, and furthermore P_i is incremented and $M(i,x_i) = P_i$, where $C_i \in (0,1)$ is a preset threshold. Check the new incoming data for each input variable.

Second make a decision whether a new fuzzy rule node needs to be generated in SCFNN. As long as any new MF node(s) is added in the same structure learning round, a new rule node must be added. Assume Q is number of the existing rules. If there exist a rule R_l: IF x_1 is $A_1^{S(l,1)}$ and…and x_n is $A_n^{S(l,n)}$, then $y = b_l$, such that $M(i,x_i) = S(l,n)$ for $i = 1,2,\cdots,n$, then enter parameters learning phase directly; otherwise a new rule node is added.

If there is any new MF node generated in this structure learning round, a new rule node needs to be generated; otherwise enter parameters learning phase directly. For the i th input variable, let

$$S(Q+1,i) = M(i,x_i)\,,\quad i=1,\cdots,n\,. \tag{5}$$

Then the new rule is generated as follows:

$$R_{Q+1}: \text{IF } x_1 \text{ is } A_1^{S(Q+1,1)} \text{and…and } x_n \text{ is } A_n^{S(Q+1,n)}, \text{then } y = b_{Q+1}\,. \tag{6}$$

where $b_{Q+1} = w_{Q+1}$ is selected with a random or preset constant. Q is incremented.

Parameter Learning Phase. The core of the parameter learning algorithm for the SCFNN is based on the backpropagation learning algorithm. The backpropagation adjust the link weights and the parameters of MF.

First the energy function is defined as

$$V \equiv e^2/2 \tag{7}$$

The parameter learning algorithm is described as follows:

In layer 4, the error term to be propagated is computed as

$$\delta^4 = -\frac{\partial V}{\partial y^*} = \left[-\frac{\partial V}{\partial e}\frac{\partial e}{\partial y^*}\right] = \left[-\frac{\partial V}{\partial e}\frac{\partial e}{\partial o}\frac{\partial o}{\partial y^*}\right] = -\frac{e}{r}\frac{\partial o}{\partial y^*} \tag{8}$$

and the link weight is updated by the amount

$$\Delta\omega_l = -\eta_\omega \frac{\partial V}{\partial \omega_l} = -\left[\eta_\omega \frac{\partial V}{\partial y^*}\right]\left[\frac{\partial y^*}{\partial \omega_l}\right] = \eta_\omega \delta^4 \psi_l \tag{9}$$

where factor η_ω is the learning-rate parameter of the link weight. The link weights in layer 4 are updated according to the following equation:

$$\omega_l(N+1) = \omega_l(N) + \Delta\omega_l \tag{10}$$

where N denotes the iteration number of the l th link.

In layer 3, only the error term needs to be calculated and propagated

$$\delta_l^3 = -\frac{\partial V}{\partial \psi_l} = \left[-\frac{\partial V}{\partial y^*}\right]\left[\frac{\partial y^*}{\partial \psi_l}\right] = \delta^4 \omega_l \tag{11}$$

In layer 2, the error term is computed as follows:

$$\delta_{ji}^2 = -\frac{\partial V}{\partial \mu_{i,j}} = -\sum_l \left[\frac{\partial V}{\partial y^*}\frac{\partial y^*}{\partial \psi_l}\right]\left[\frac{\partial \psi_l}{\partial \mu_{i,j}}\right] = \sum_l \delta_l^3 \frac{\partial \psi_l}{\partial \mu_{i,j}} \tag{12}$$

If A_i^j does not appear in the l th rule, then $j \neq S(l,i)$ and $\frac{\partial \psi_l}{\partial \mu_{i,j}} = 0$; otherwise then $j = S(l,i)$ and $\frac{\partial \psi_l}{\partial \mu_{i,j}} = \prod_{k \neq i} \mu_{k,S(l,k)}$. Introduce a symbol function $\theta(l,i,j)$:

$$\theta(l,i,j) = \begin{cases} 1 & j = S(l,i) \\ 0 & j \neq S(l,i) \end{cases}. \tag{13}$$

(12) can be rewritten as

$$\delta_{ji}^2 = \sum_l \theta(l,i,j)\delta_l^3 \prod_{k \neq i} \mu_{k,S(l,k)} \tag{14}$$

The update law of m_{ji} is

$$\Delta m_{ji} = -\eta_m \frac{\partial V}{\partial m_{ji}} = -\eta_m \frac{\partial V}{\partial \mu_{i,j}}\frac{\partial \mu_{i,j}}{\partial m_{ji}} = \eta_m \sum_l \theta(l,i,j)\delta_l^3 \prod_{k \neq i} \mu_{k,S(l,k)} \mu_{i,j} \frac{2(x - m_{ji})}{\sigma_{ji}^2} \tag{15}$$

If $j = S(l,i)$, that is $\theta(l,i,j) = 1$, then $\theta(l,i,j)\mu_{i,j} = \theta(l,i,j)\mu_{i,S(l,i)}$, otherwise $\theta(l,i,j) = 0$ and $\theta(l,i,j)\mu_{i,j} = \theta(l,i,j)\mu_{i,S(l,i)} = 0$. So according to (4), the equation above can be rewritten as

$$\Delta m_{ji} = -\eta_m \frac{\partial V}{\partial m_{ji}} = = 2\eta_m \sum_l \theta(l,i,j)\delta_l^3 \psi_l \frac{(x_i - m_{ji})}{\sigma_{ji}^2} \tag{16}$$

The update law of σ_{ji} is

$$\Delta\sigma_{ji} = -\eta_\sigma \frac{\partial V}{\partial \sigma_{ji}} = = 2\sigma_m \sum_l \theta(l,i,j)\delta_l^3 \psi_l \frac{(x_i - m_{ji})^2}{\sigma_{ji}^3} \tag{17}$$

where η_m and η_σ are the learning-rate parameters of m_{ji} and σ_{ji}, respectively. m_{ji} and σ_{ji} are updated as follows:

$$m_{ji}(N+1) = m_{ji}(N) + \Delta m_{ji} \tag{18}$$

$$\sigma_{ji}(N+1) = \sigma_{ji}(N) + \Delta\sigma_{ji} \tag{19}$$

The exact calculation of the Jacobian of the system $\partial o / \partial y^*$ which is contained in $\partial V / \partial y^*$ cannot be determined due to the uncertainties of the plant dynamic such as parameter variations, external disturbances and network-induced delay. To overcome this problem and to increase the learning rate of the network parameters, a similar calculation $\Delta o / \Delta y^*$ is adopted.

4 Results of Simulation

The structure of NCS implemented by Profibus-DP is shown as Fig.5. The remote controller receives the feedback signal from plant and transmits the control signal to plant, while the plant receives the control signal from controller and transmits the feedback signal to controller.

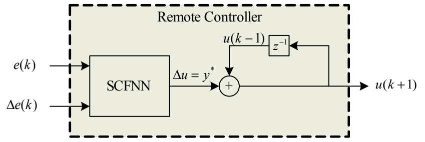


Fig. 4. Remote Controller based on SCFNN

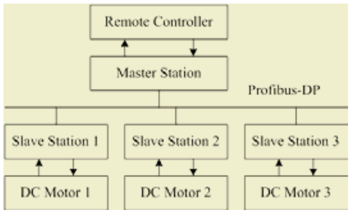


Fig. 5. Experimental model of NCS

A DC motor is used as the plant of NCS. The transfer function of the motor is

$$G(s) = \frac{543.47}{0.12s + 1} . \tag{20}$$

In the experimental model, a remote controller and three DC motors are connected through a Profibus-DP network. The remote controller can control these three motors through the network simultaneously. The transmission speed of network is 1.5Mbps, and the packet size is 2 bytes. The rolling period of network is 2msec.The sampling period of NCS is 10msec, and reference input is assigned as 700rpm. The design specifications of system are as follows: percent overshoot $\leq 10\%$, 5% settling time ≤ 0.8 sec. The simulation is carried out using Matlab package.

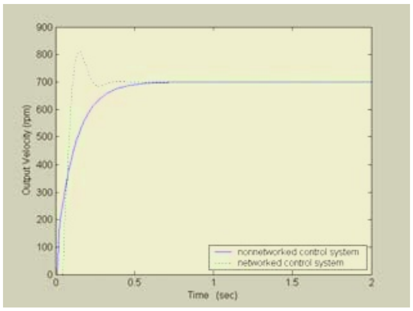


Fig. 6. The influence of network-induced delay on the system behavior

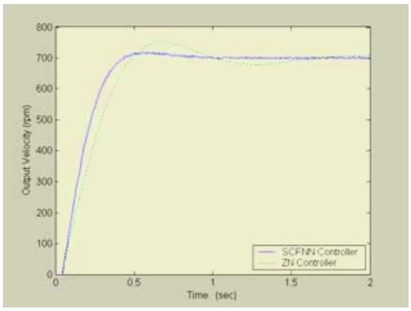


Fig. 7. Comparison of ZN controller and SCFNN controller

In the nonnetworked control system, PID controller is used as remote controller. The parameters of the controller are $K_P = 0.001451$, $K_i = 0.0395$ and $K_d = 0.0000327$, respectively. The percent overshoot and settling time of system are 0 and 0.45msec, respectively. When the Profubus-DP is introduced in the feedback loop, the percent overshoot and settling time of NCS are 15.88% and 0.41msec, respectively, which are worse than design specification.

To evaluate the effectiveness of the controller designed by SCFNN (SCFNN controller for short), the SCFNN controller is compared to the controller designed by a modified Ziegler-Nichols method (ZN controller for short). The network-induced delay is treated as the time delay of the plant in applying Ziegler-Nichols method. From the experiments, the initial network-induced delay is found to be equal to three sampling intervals, i.e. 30msec. The gains of ZN controller are $K_P = 0.008832$, $K_i = 0.147$ and $K_d = 0.000132$.

The comparison of step response by SCFNN controller and ZN controller is shown in Fig. 7. The percent overshoot and settling time of SFCNN controller are 2.2% and 0.37msec, respectively, while those of ZN controller are 8% and 0.85msec, respectively.

5 Conclusions

SCFNN is proposed in this paper to design remote controller in NCSs. The initial SCFNN consists of input and output nodes only. In the learning process the nodes of the middle layers, which correspond to the membership functions and the fuzzy rules, are created gradually, so a set of fuzzy rules is achieved dynamically. In the study we can observe:

1. In NCS, the existence of network-induced delay will degrade the performance of control systems, so network-induced delayed should be taken into accounted when designing remote controller for NCS.
2. SCFNN integrates the merits of fuzzy control and neural network, so it is effective in designing remote controller for NCS even without any prior knowledge on network-induced delay.

References

1. Lin, J., Lin, C.H.: A Permanent-Magnet Synchronous Motor Servo Drive Using Self-Constructing Fuzzy Neural Network Controller, IEEE Trans. on Ener. Conv., **16** (2004) 66-72
2. Lin, F.J., Lin, C.H.: Self-Constructing Fuzzy Neural Network Speed Controller for Permanent-Magnet Synchronous Motor Servo Drive. IEEE Trans. on Fuzz. Syst., **9** (2001) 751-759.
3. Zhang, W., Branicky, M.S., Phillips, S.M.: Stability of Networked Control Systems. IEEE Control System Magazine, **21** (2001) 85-99
4. Walsh, G.C., Ye, H.: Scheduling of Networked Control Systems, IEEE Control System Magazine, **21** (2001)57-65
5. Park, H.S., Kim, Y.H., Kim, D.S., Kwon, W.H.: A Scheduling Method for Network-Based Control Systems. IEEE Trans. on Cont. Syst., **10** (2002) 318-330
6. Walsh, G.C., Ye, H., Bushnell, L.G.: Stability Analysis of Networked Control Systems. IEEE Trans. on Cont. Syst,, May, **10** (2002) 438-446
7. Walsh, G.C., Beldiman, O., Bushnell L.G.: Asymptotic Behavior of Networked Control Systems, IEEE Trans. On Cont. Syst., May, **46** (2001) 1093-1097
8. Lee, K. C., Lee, S., Lee, M. Y.: Remote Fuzzy Logic Control of Networked Control System via Profibus-DP, IEEE Trans. on Indu. Elec., 50 (2003) 784-792

Sliding Mode Control for Cross Beam Simulation System via Neural Network

Hongchao Zhao[1], Qingjiu Xu[2], Wenjin Gu[1], and Tingxue Xu[3]

[1] Faculty 301, Naval Aeronautical Engineering Institute, Yantai, Shandong 264001, China
playzhc@163.com, kzgwj@sina.com

[2] Faculty 302, Naval Aeronautical Engineering Institute, Yantai, Shandong 264001, China
navyqjxu@126.com

[3] Faculty 203, Naval Aeronautical Engineering Institute, Yantai, Shandong 264001, China

Abstract. The coupled nonlinear model of the cross beam simulation system (CBSS) was transformed into a special form. This form consists of linear terms and nonlinear terms. The nonlinear terms were treated as uncertaimties and compensated by neural network, thus the model was decoupled into three independent channels. The sliding mode control via neural network was used to design a controller for each channel. Stability analysis showed the asymptotical stability of the system under the control of above controller. The effectiveness of the sliding mode control via neural network is verified by the simulation.

1 Introduction

In the middle of 1950's, the United States began to research the cross beam simulation system (CBSS), which was mainly applied to reaction attitude control for supersonic and hypersonic vehicles. As a simulation and experiment sytem, the CBSS has very high application value in the military field. The research on the CBSS in our country is at the elementary stage. The model of the CBSS was constructed in [1], which was a high order coupled nonlinear system. The adaptive variable structure controller and discrete nonlinear PID controller were designed, respectively, for the CBSS in [1] and [2], and their simulation results were satisfactory. However, above researches lack of theory analysis, and their controllers are designed only through simulation.

In this paper, the model of the CBSS is transformed into a special form, which consists of linear terms and nonlinear terms. The nonlinear terms are treated as uncertaimties, which are compensated by neural networks. Therefore, the model is decoupled into three independent channels: pitch channel, yaw channel and roll channel. For each channel, we design a sliding mode controller to ensure the asymptotic stability of the system in the sense of Lyapunov stability.

2 Model of the CBSS

The mathematical model of the CBSS has already been given in [1] and [2], which can be transformed into the following form in three channels,

$$\begin{cases}\dot{\vartheta}=\omega_{y_1}\sin\gamma+\omega_{z_1}\cos\gamma \\ \dot{\omega}_{z_1}=\left(F_{z_1}l-f_{z_1}\dot{\vartheta}\right)/J_{z_1}-\left(J_{y_1}-J_{x_1}\right)\omega_{y_1}\omega_{x_1}/J_{z_1}\end{cases} \tag{1}$$

$$\begin{cases}\dot{\psi}=\left(\omega_{y_1}\cos\gamma-\omega_{z_1}\sin\gamma\right)/\cos\vartheta \\ \dot{\omega}_{y_1}=\left(F_{y_1}l-f_{y_1}\dot{\psi}\right)/J_{y_1}\end{cases} \tag{2}$$

$$\begin{cases}\dot{\gamma}=\omega_{x_1}-tg\vartheta\left(\omega_{y_1}\cos\gamma-\omega_{z_1}\sin\gamma\right) \\ \dot{\omega}_{x_1}=\left(F_{x_1}l-f_{x_1}\dot{\gamma}\right)/J_{x_1}-\left(J_{z_1}-J_{y_1}\right)\omega_{z_1}\omega_{y_1}/J_{x_1}\end{cases} \tag{3}$$

where the meanings of all notations have already been described in [1] and [2]. F_{x_1}, F_{y_1} and F_{z_1} are the aerodynamic forces of each channel, which are the reaction of the jet thrust of the solenoid valve. The dynamics of the solenoid valve is much faster than that of the cross beam, so we think it as a proportional loop. The jet thrust of the solenoid valve is proportional to its voltage input, i.e.,

$$F_{x_1}=k_Fu_1,\ F_{y_1}=k_Fu_2,\ F_{z_1}=k_Fu_3 \tag{4}$$

where k_F is the gain of the solenoid valve, u_1, u_2 and u_3 are the control voltage inputs of the solenoid valves in each channel.

We take the pitch channel (1) as an example and further transform it as follows,

$$\begin{cases}\dot{\vartheta}=\omega_{z_1}+\Delta_1(\omega_{z_1},\omega_{y_1},\gamma) \\ \dot{\omega}_{z_1}=-f_{z_1}\omega_{z_1}/J_{z_1}+k_Flu_3/J_{z_1}-\Delta_2(\omega_{z_1},\omega_{y_1},\omega_{x_1},\gamma)\end{cases} \tag{5}$$

where $\Delta_1(\omega_{z_1},\omega_{y_1},\gamma)=\omega_{y_1}\sin\gamma+\omega_{z_1}(\cos\gamma-1)$, which is simply defined as Δ_1, $\Delta_2(\omega_{z_1},\omega_{y_1},\omega_{x_1},\gamma)=f_{z_1}\Delta_1/J_{z_1}+(J_{y_1}-J_{x_1})\omega_{y_1}\omega_{x_1}/J_{z_1}$, which is simply defined as Δ_2. The output is pitch angle ϑ, and the input is u_3. The system (5) consists of linear terms and nonlinear terms. The nonlinear terms are Δ_1 and Δ_2, which are treated as uncertainties of the pitch channel. They are bounded because every variable of them is bounded. The pitch channel can be independently designed without considering the other two channels.

The other two channels can also be transformed into the form of (5), then, the same approach may be adopted to dispose of them.

3 Sliding Mode Control via Neural Network

3.1 Sliding Mode Controller Design

Considering the output tracking problem of (5), we define the tracking error and its derivative as follows,

$$e=\vartheta-\vartheta^*,\ \dot{e}=\dot{\vartheta}-\dot{\vartheta}^* \tag{6}$$

where ϑ^* is the desired value of the output. A switching surface is defined as

$$s=c_1e+c_2\dot{e} \tag{7}$$

where $c_1,\ c_2 > 0$. A reaching condition of the sliding mode control is [3]

$$\dot{V} < 0\,(s \neq 0)\text{, where } V = s^2/2 \tag{8}$$

Using (5), (6) and (7) in (8), we have

$$\begin{aligned} \dot{V} &= s\dot{s} = s(c_1\dot{e} + c_2\ddot{e}) \\ &= s[c_1(\omega_{z_1} + \Delta_1 - \dot{\vartheta}^*) + c_2(k_F l u_3 / J_{z_1} - f_{z_1}\omega_{z_1}/J_{z_1} - \Delta_2 + \dot{\Delta}_1 - \ddot{\vartheta}^*)] \\ &= s[c_1(\omega_{z_1} - \dot{\vartheta}^*) - c_2(f_{z_1}\omega_{z_1}/J_{z_1} + \ddot{\vartheta}^*) + c_2 k_F l u_3/J_{z_1} + f(\Delta)] \end{aligned} \tag{9}$$

where $f(\Delta) = c_1\Delta_1 + c_2\dot{\Delta}_1 - c_2\Delta_2$ is named the lumped uncertainty. We assume that the lumped uncertainty satisfies the bound condition, i.e. $|f(\Delta)| \leq F_0$, where F_0 is a positive function. If F_0 is known, the sliding mode controller can be designed as follows,

$$\begin{aligned} u_3 = (c_2 k_F l)^{-1}[-J_{z_1}c_1(\omega_{z_1} - \dot{\vartheta}^*) + c_2(f_{z_1}\omega_{z_1} \\ + J_{z_1}\ddot{\vartheta}^*) - J_{z_1}(F_0 + \rho)\operatorname{sgn}(s)] \end{aligned} \tag{10}$$

where $\rho > 0$, sgn(·) is a sign function. Using (10) in (9) gives: $\dot{V} \leq -\rho|s| < 0$, so the reaching condition is guaranteed. However, the bound F_0 is difficult to obtain in advance for practical application. In next section the neural network will be used to approximate the bound F_0.

3.2 Neural Network Design

A RBF neural network can be used to adaptively learn the bound of the lumped uncertainty [4], [5], that is,

$$\hat{F}_0 = \hat{W}^T \phi(X) \tag{11}$$

where $\hat{F}_0$ is the estimate of F_0, $\hat{W}$ is weight vector of the RBF neural network, $X = [e \quad \dot{e}]^T$, $\phi(X) = [\varphi_1(X),\ \cdots,\ \varphi_n(X)]^T$, $\varphi_i(X)$ is Gaussian basis function. We take $\varphi_i(X) = 0$ if $\varphi_i(X) < 0.05$. For the further analysis, we assume as follows,

Assumption 1: For arbitrarily small positive number δ, there exists the optimal weight vector W^*, such that

$$|\varepsilon(X)| = |W^{*T}\phi(X) - F_0| < \delta \tag{12}$$

where $\varepsilon(X)$ is the approximation error of the RBF neural network. Note that F_0 is positive, hence $W^{*T}\phi(X)$ is positive. Form (12) we can get

$$F_0 < W^{*T}\phi(X) + \delta \tag{13}$$

The controller design and stability analysis are stated in the following theorem.

Theorem 1: For the system (5) with the tracking error (6), the switching surface is defined as (7), the sliding mode controller via neural network is designed as

$$u_3 = (c_2 k_F l)^{-1}[-J_{z_1} c_1(\omega_{z_1} - \dot{\vartheta}^*) + c_2(f_{z_1}\omega_{z_1} + J_{z_1}\ddot{\vartheta}^*) - J_{z_1}(\hat{W}^T\phi(X) + \rho)\operatorname{sgn}(s)] \tag{14}$$

and the weight vector is online modified according to

$$\dot{\hat{W}} = \Gamma\phi(X)\,|\,s\,| \tag{15}$$

where $\Gamma = diag(\eta_1, \cdots, \eta_n)$ with $0 < \eta_i < 1$ is the learning rate, then the system will be asymptotically stable and the tracking error will asymptotically converge to zero.

Proof: For the system (5), define a Lyapunov function as

$$V = s^2/2 + \tilde{W}^T\Gamma^{-1}\tilde{W}/2 \tag{16}$$

where $\tilde{W} = \hat{W} - W^*$, $\dot{\tilde{W}} = \dot{\hat{W}}$. The derivative of V is as follows,

$$\begin{aligned}\dot{V} &= s\dot{s} + \tilde{W}^T\Gamma^{-1}\dot{\hat{W}} \\ &= s[c_1(\omega_{z_1} - \dot{\vartheta}^*) - c_2(f_{z_1}\omega_{z_1}/J_{z_1} + \ddot{\vartheta}^*) + c_2 k_F l u_3/J_{z_1} + f(\Delta)] + \tilde{W}^T\Gamma^{-1}\dot{\hat{W}} \\ &\le s[c_1(\omega_{z_1} - \dot{\vartheta}^*) - c_2(f_{z_1}\omega_{z_1}/J_{z_1} + \ddot{\vartheta}^*) + c_2 k_F l u_3/J_{z_1}] + |\,s\,|F_0 + \tilde{W}^T\Gamma^{-1}\dot{\hat{W}}\end{aligned} \tag{17}$$

Substituting (13) (14) and (15) into (17) yields

$$\begin{aligned}\dot{V} &< -(\hat{W}^T\phi(X) + \rho)|\,s\,| + (W^{*T}\phi(X) + \delta)|\,s\,| + \tilde{W}^T\Gamma^{-1}\dot{\hat{W}} \\ &= -(\rho - \delta)|\,s\,| - \tilde{W}^T\phi(X)|\,s\,| + \tilde{W}^T\Gamma^{-1}\dot{\hat{W}} \\ &= -(\rho - \delta)|\,s\,| < 0\end{aligned} \tag{18}$$

where $\rho > \delta$ for δ is arbitrarily small, so the system (5) is asymptotically stable in the sense of Lyapunov stability, and the system trajectory will reach the switching surface and then stay in it, i.e., $s = c_1 e + c_2\dot{e} = 0$, $\dot{e} = -c_1 e/c_2$, which means that the tracking error will asymptotically converge to zero.

The controllers u_1 and u_2 can also be designed in the same form of u_3.

4 Simulation Example

In order to evaluate the effect of the sliding mode control via neural network approach, the simulation is performed. The model of the CBSS is described by (1)~(4) with the parameters: $J_{x_1} = J_{z_1} = 5kg \cdot m^2$, $J_{y_1} = 1.46kg \cdot m^2$, $l = 1m$, $k_F = 1.96$, $f_{z_1} = f_{y_1} = f_{x_1} = 1.46N \cdot m \cdot s$. The desired attitude angles of three channels are $\vartheta^* = 10\deg$, $\psi^* = -10\deg$, $\gamma^* = 5\deg$. The controllers of three channels are designed according to (14) and (15). In order to reduce the chattering in sliding mode control system, the sign function sgn(s) is replaced by a continuous function $s/(|s|+\varepsilon)$, where ε is a little positive number, here $\varepsilon = 0.001$. Figure 1 shows the output responses of the CBSS. Comparing with the simulation result of reference [2], the simulation result of this paper shows faster response speed and better tracking performance. The approximation ability to uncertainty of the RBF neural network is also demonstrated by the simulation result.

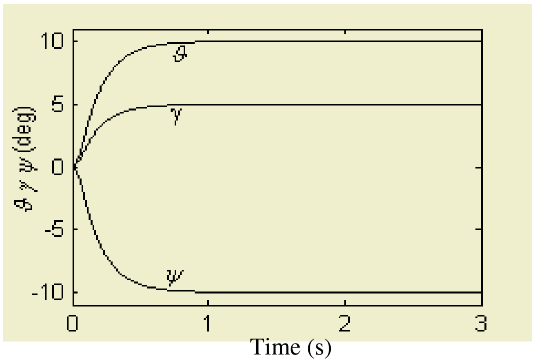


Fig. 1. Output responses of the CBSS

5 Conclusions

The CBSS is a coupled nonlinear system. However, with the nonlinear terms compensated by the RBF neural network, the model of the CBSS can be transformed into three decoupled systems in three channels. The sliding mode control via neural network approach is used in each channel to design a controller, which guarantees the asymptotic stability of the system. Finally, a simulation example is illustrated to test the tracking ability of the CBSS under the control of the designed controller. the simulation result shows the effectiveness of the sliding mode control via neural network approach.

References

1. Gu, W., Guo, Z.: Design of Cross Beam Simulation System. Proceedings of Asian Simulation Conference, Shanghai (2002) 154-158
2. Guo, Z., Gu, W., Wang, H.: Research on Cross Beam Experiment System Based on Discrete Nonlinear PID Control. Journal of System Simulation, **9** (2003) 1322-1324
3. Hung, J. Y., Gao, W., Hung, J. C.: Variable Structure Control: A Survey. IEEE Transactions on Industrial Electronics, **1** (1993) 2-22
4. Man, Z., Yu, X., Eshraghian, K., et al.: A Robust Adaptive Sliding Mode Tracking Control Using a RBF Neural Network for Robotic Manipulators. Proceedings of IEEE International Conference on Neural Networks, San Diego (1995) 2403-2408
5. Niu, Y., Yang, C., Chen, X.: Adaptive Sliding Mode Control for Robot Manipulators Based on Neural Network. Control and Decision, **1** (2001) 79-82

Vibration Suppression of Adaptive Truss Structure Using Fuzzy Neural Network

Shaoze Yan[1], Kai Zheng[1], and Yangmin Li[2]

[1] Department of Precision Instruments and Mechanology, Tsinghua University, Beijing 100084, P.R. China
yansz@mail.tsinghua.edu.cn
[2] Department of Electromechanical Engineering, University of Macau, Macao SAR, P.R. China
ymli@umac.mo

Abstract. An adaptive truss structure with self-learning active vibration control system is developed. A fuzzy-neural network (FNN) controller with adaptive membership functions is presented. The experimental setup of a two-bay truss structure with active members is constructed, and the FNN controller is applied to vibration suppression of the truss. The controller first senses the output of the accelerometer as an error to activate the adaptation of the weights of the controller, and then a control command signal is calculated based on the FNN inference mechanism to drive the active members. This paper describes active vibration control experiments of the truss structure using fuzzy neural network.

1 Introduction

Among the modern structures of huge spacecrafts, the truss is one of the most commonly used structures. In order to reduce the cost of launching and to increase the payload, the structural weight must be as light as possible. On the other hand, they must possess excellent dynamic behaviors to ensure the safety and stability of the structures, and be able to keep the instruments and equipment fixed on them in good condition. However, these two requirements are often contradictory. It may be possible to create adaptive space structures capable of adapting to or correcting for changing operating conditions, since the subject areas of smart/intelligent materials and intelligent control have experienced tremendous growth in terms of research and development in the last two decades. The fuzzy adaptive back-propagation algorithm which combined fuzzy theory with artificial neural network techniques was applied to the identification of restoring forces in non-linear vibration systems [1]. An artificial neural network was used for the active vibration control of a flexible aluminum cantilever beam employing a piezoactuator [2]. The length adjustable active members were employed to suppress vibration of adaptive truss structures with a direct output feedback control [3]. Adaptive neural control for space structure vibration suppression was also developed [4].

This article describes vibration suppression of adaptive truss structures using fuzzy neural network (FNN) control scheme. The proposed scheme is based on the structure of the self-tuning regulator and employs neural network and genetic algorithm (GA) techniques. The scheme is described in Section 2. An adaptive truss structure with self-learning active vibration control system using the FNN and experimental results are presented in Section 3. The final section is the conclusions.

2 Fuzzy Neural Network

A five-layered fuzzy neural network (FNN) is presented to construct self-learning active vibration control system of adaptive truss structure with PZT active members. The structure diagram of the FNN with two inputs and one output is shown in Fig.1. The coherent binding of the neural network and fuzzy logic represents an advanced intelligent system architecture.

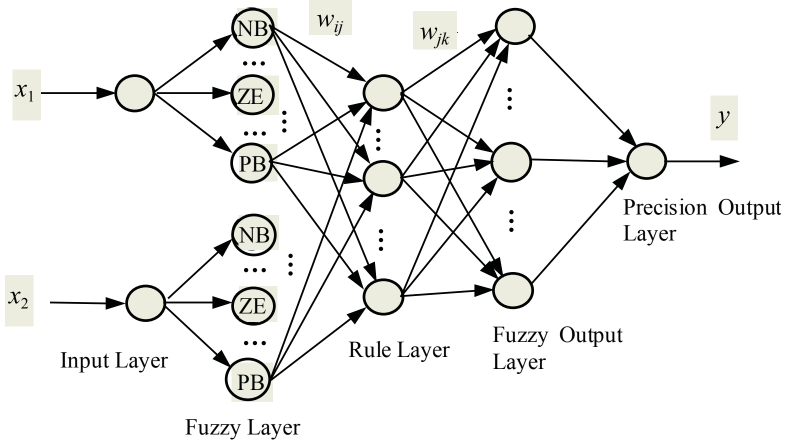


Fig. 1. The structure diagram of the FNN

2.1 Layered Operation of FNN

Let x_{ki} be the ith input of the kth layer. net_{kj} and y_{kj} denote the input and the output of the jth node in the kth layer respectively, so we can obtain $y_{kj} = x_{(k+1)j}$. In the following, we will explain the physical meaning and the node functions of each layer. Layer 1 is the input layer whose nodes are the input ones,

$$y_{1j} = net_{1j},\ \ net_{1j} = x_{1i},\quad j = i\,, \tag{1}$$

where x_{1i} is the ith input of the first layer, i=1,2. Layer 2 is the linguistic term layer which uses Gaussian function as membership function, so these term nodes map input x_i onto their membership degree y_j by using the jth term node of x_i , i.e.,

$$y_{2j} = \exp(-(x_i - m_{ij})^2 / \sigma_{ij}^2)\,, \tag{2}$$

where m_{ij} and σ_{ij} denote mean (center) and variance (width) with respect to the j th fuzzy set of the i th input. Layer 3 is the rule layer that implements the related link for preconditions (term node) and consequence (output node). The jth output of layer 3 is

$$y_{3j} = net_{3j} = \prod_{i=1}^{p} x_{3i}\quad . \tag{3}$$

Layer 4 is the fuzzy output layer whose output is membership degree. Layer 5 is the output layer, which performs the centroid defuzzification to get numerical output, i.e.,

$$y_{5j} = net_{5j} / \sum(\sigma_{ij} x_{5i}), \qquad net_{5j} = \sum w_{5ij} x_i = \sum (m_{ij}\sigma_{ij}) x_{5i} \ , \tag{4}$$

where y_{51} is the final output of the neutral net, and w_{5ij} denotes the weight value of the 5th layer, $w_{5ij} = m_{ij}\sigma_{ij}$.

2.2 Definition and Adjustment of Fuzzy Rules

The adjusted parameters in the FNN can be divided into two categories based on IF (premise) part and THEN (consequence) part in the fuzzy rules. A gradient descent based BP algorithm is employed to adjust the FNN's parameters [5]. In the premise part, the parameters are mean and width of Gaussian function. For initializing these terms, we use normal fuzzy sets.

The problem in designing fuzzy control systems is how to determine the free parameters of the fuzzy controller, i.e. rules and membership functions. Membership functions are used for the fuzzy input and output variables. We use the membership functions with seven linguistic variables (NB, NM, NS, ZE, PS, PM, PB), which represent negative big, negative medium, negative small, zero, positive small, positive medium, and positive big, respectively. They can be scaled into the range of -6 and +6 with equal span. For the vibration suppression of the truss structure, the Gaussian function is employed as initial membership function, shown in Fig. 2. Forty-nine fuzzy rules based on the displacement or acceleration and the difference in subsequent time are shown in Table 1. We can describe the fuzzy rules in Table 1 as follows,

$$\text{If } (A \text{ is } A_i \ , B \text{ is } B_j) \text{ then } (U \text{ is } U_{ij}), \qquad i, j = 1,2,...,7 \ , \tag{5}$$

where A and B are inputs of the controller, and U is output of the controller; A_i and B_j are the antecedent linguistic values for the *i*th and *j*th rule respectively, and U_{ij} is the consequent linguistic values.

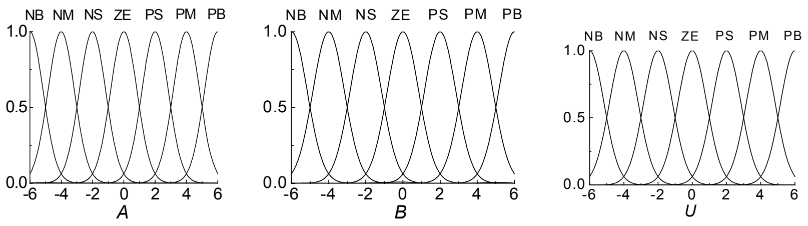
The training set consists of input vectors and corresponding desired output vectors. The output of the neuron is a function of the total weighted input. The membership functions are tuned by the delta rule [6]. The error function for the FNN controller is defined as

$$E = (y_d - y_{51})^2 / 2 \ , \tag{6}$$

where y_d and y_{51} are the desired and actual outputs of the FNN.

Table 1. The fuzzy rules of the FNN

U_{ij} \ B_j / A_i	NB	NM	NS	ZE	PS	PM	PB
NB	PB	PB	PB	PM	PM	ZE	ZE
NM	PB	PB	PB	PM	PM	ZE	ZE
NS	PM	PM	PM	PS	ZE	NS	NS
ZE	PM	PM	PS	ZE	NS	NS	NS
PS	PM	PS	ZE	NS	NM	NM	NM
PM	PS	ZE	ZE	NM	NM	NB	NB
PB	ZE	ZE	ZE	NM	NM	NB	NB



(a) Input *A* (b) Input *B* (c) Output *U*

Fig. 2. Fuzzy input and output membership functions

The error signals are propagated backward from the 5th layer to the 1st layer in order to align themselves with the error signals. The controller weights are updated according to

$$w(t+1) = w(t) - \eta \partial E / \partial w, \tag{7}$$

where η is the learning rate which is generally between 0.01 and 1, and

$$\frac{\partial E}{\partial w} = \frac{\partial E}{\partial net} \cdot \frac{\partial net}{\partial y} \cdot \frac{\partial y}{\partial E}. \tag{8}$$

The training continues until the error is within acceptable limits. Since the system dynamics are usually unknown, the sensitivity is unknown. However, this can be estimated using the neural identifier. After the identifier is sufficiently trained, the dynamic behavior of the identifier is close to that of the structure [5]. In order to avoid BP network converging to local optimum, a hybrid training algorithm combined genetic algorithm (GA) with BP algorithm is established. The mean and width of Gaussian function can be adjusted and optimized by GA. We can obtain structure coarse optimization of fuzzy neural network by using the coding method, the fitness function, and the genetic operations (crossover, mutation, and selection), and fine tuning of weights w_{ij} can be performed by BP [6].

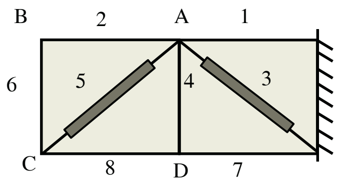


Fig. 3. Two-bay truss structure with active members

Fig. 4. Experimental setup

3 Active Vibration Control Experiments

Figures 3 and 4 show a two-bay truss structure and the photograph of the experimental setup with active members. The active members 3 and 5 shown in Fig.5 are posed as a diagonal brace of each bay truss for vibration control. The other truss bars are made of aluminum alloy tubs. The FNN controller is applied to the truss vibration suppression. The FNN controller first senses the output of the accelerometer as an error to activate the adaptation of the weights of the controllers. Then a control command signal is calculate based on the FNN inference mechanism to drive the active members. The initial populations size of GA is chosen as 20. Probability of the mutation operation is 0.02, and the crossover rate 0.5. The BP algorithm with learning rate $\alpha = 0.02$ and $\beta = 0.02$ is used to train the FNN for obtaining weight value of the nodes. The sampling frequency is selected as 1kHz.

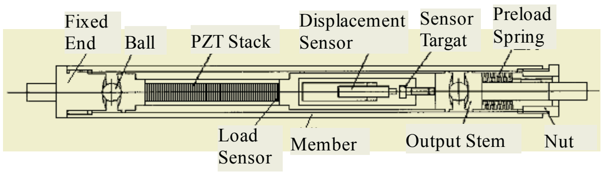


Fig. 5. PZT active member architecture

An impulse force excitation is given at the node *C* of the truss, and we only use the active member 3 to control vibration of the truss. Figure 6 shows acceleration responses at the node *A* of the truss in the time domain. As expected, the amplitude of vibrations decreases significantly by using the closed-loop control with the FNN. The stable times of vibration amplitudes for open-loop (uncontrolled) and closed-loop control are about 0.15s and 0.06s respectively under an impulse excitation. It can be seen that the controller provides excellent damping. We have also swiped the excitation frequency to examine the adaptation capability of this control system. The controller of the adaptive truss structure offered the similar performance for different frequency disturbance.

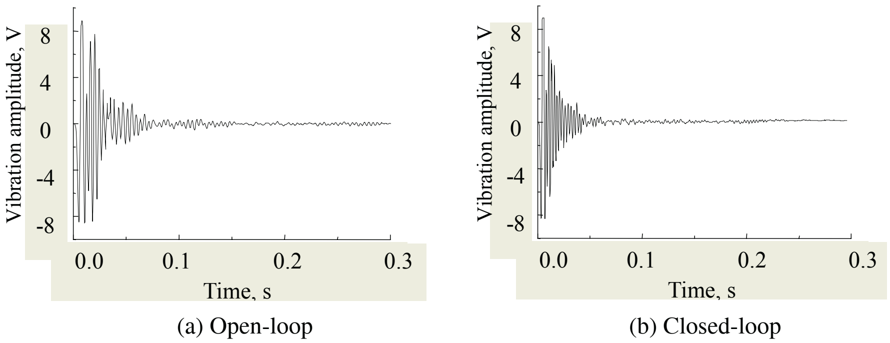


Fig. 6. Responses of the truss under a pulse excitation

4 Conclusions

In this paper the adaptive truss structure with active vibration control system is developed. A fuzzy neural network architecture to design and implement a self-learning control system for active vibration suppression applications is described. Active control is achieved by processing sensor voltage through a compensator and amplifying the voltage signal to the PZT active members. We have experimentally demonstrated the application of a FNN controller to active vibration control of the truss structure with the active members. Experimental results show that the FNN controller is efficient for vibration control of the truss structure.

Acknowledgement

This work is supported by the National Science Foundation of China under grant 59905016 and 50275080.

References

1. Liang, Y.C., Feng, D.P., Cooper, J.E.: Identification of Restoring Forces in Nonlinear Vibration Systems Using Fuzzy Adaptive Neural Networks. Journal of Sound and Vibration, 1 (2001) 47-58
2. Choi, W. C., Kim, N.W.: Experimental Study on Active Vibration Control of a Flexible Cantilever Using an Artificial Neural-Network State Predictor. Smart Mater. Struct. 5 (1996) 751-758.
3. Lyan-Ywan, L., Utku, S., Wada, B.K.: Vibration Suppression for Large Scale Adaptive Truss Structures Using Direct Output Feedback Control. Journal of Intelligent Material Systems and Structures, 3 (1993) 385-397
4. Davis, L., Hyland, D., Yen, G.: Adaptive Neural Control For Space Structure Vibration Suppression. Smart Mater. Struct. 8 (1999) 753-766.
5. Chen, Y.J., Teng, C.C.: Rule Combination in a Fuzzy Neural Network. Fuzzy Sets and Systems, 2 (1996) 161-166
6. Ishigami, H., Fukuda, T., Shibata, T., Arai, F.: Structure Optimization of Fuzzy Neural Network by Genetic Algorithm. Fuzzy Sets and Systems, 3 (1995) 257-264

Experimental Investigation of Active Vibration Control Using a Filtered-Error Neural Network and Piezoelectric Actuators

Yali Zhou[1], Qizhi Zhang[1], Xiaodong Li[2], and Woonseng Gan[3]

[1] Department of Computer Science and Automation, Beijing Institute of Machinery, P. O. Box 2865, Beijing 100085, China
zhouyali@yahoo.com

[2] Institute of Acoustic, Academia Sinica, People's Republic of China

[3] School of EEE, Nanyang Technological University, Singapore

Abstract. The filtered-error back-propagation neural network (FEBPNN) algorithm for control of smart structure is investigated experimentally. Piezoelectric actuator is employed to suppress the structural vibration. The controllers based on the FEBPNN algorithm and the filtered-x least mean square (FXLMS) algorithm are implemented on a digital signal processor (DSP) TMS320VC33. The experimental verification tests show that the FEBPNN algorithm is effective for a nonlinear control problem, and has better tracking ability under change of the primary disturbance signal.

1 Introduction

Vibration control is of importance in many areas of engineering, in recent years, active vibration control (AVC) of smart structures has been the subject of intensive research [1]. The real time sensing and actuation capabilities of smart structure provide a powerful means for AVC. A smart structure involves the integration of a physical structure with distributed actuators and sensors and a controller. Based on the piezoelectric effect, piezoelectric transducers have been used extensively in active control of smart structure vibrations as sensors and also as actuators [2]. For nonlinear systems, the most common algorithm is filtered-x back-propagation neural network (FXBPNN) [3], but it has relatively high computational load, so, a filtered-error back-propagation neural network (FEBPNN) algorithm is proposed here. It can reduce the computational load greatly, which will be discussed in detail in this paper.

2 Algorithms Development

The block diagram of an AVC system using the neural network algorithm and the neural network controller are shown in Fig.1 and Fig.2, respectively. P(Z) and S(Z) are the primary path and secondary path, respectively. In Fig.2, for the output node, the function was linear, for the hidden node, a sigmoidal nonlinear function was used [3].

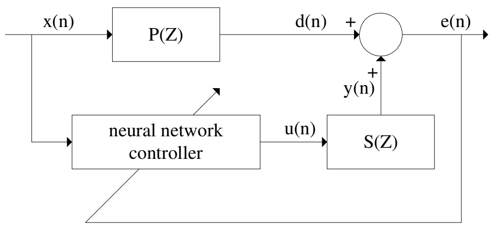


Fig. 1. The block diagram of an AVC system

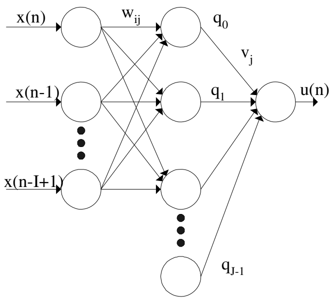


Fig. 2. The neural network controller

2.1 FXBPNN Algorithm

In order to compare the FEBPNN algorithm with the FXBPNN algorithm, the FXBPNN algorithm will first be derived, with the arrangement shown here, the reference input signal x(n) is used to derive the control signal, via the neural network controller. Each control input to the system is modified by the secondary path transfer function to produce the control signal y(n), each error signal is then the sum of the primary and control components.

The error criterion is to minimize the mean square value of the error signal e(n):

$$\min E[e^2(n)] \ . \tag{1}$$

For practicality, the instantaneous estimate of squared error is used to approximate the mean square value of the error signal e(n):

$$\min[e^2(n)] \ . \tag{2}$$

The gradient descent algorithm is employed to adjust the weights of the control system. The weights are updated by adding a negative gradient of the error criterion with respect to the weights of interest:

$$w(n+1) = w(n) - \mu\Delta w(n) \ . \tag{3}$$

where μ is the learning rate(step size).

Now, the objective is to find a solution to Eq. (3) for each weight in the neural network controller. Due to being limited by the length of the paper, only the results are given here.

The adaptation algorithm of the weights in the output layer can be written as:

$$v_j(n+1) = v_j(n) - \mu e(n) \sum_{i=0}^{M-1} h_i q_j(n-i) \ . \qquad (4)$$

$$j=0,1,\ldots J-1.$$

Where $H=[h_0\ h_1\ldots\ h_{M-1}]$ is the M-th-order impulse response of the secondary path model, whose z-transform is H(z) (where $H(Z) = \hat{S}(Z)$), which is modeled as a FIR filter during training stage prior to the start of the AVC [4].

The adaptation algorithm of the weights in the hidden layer can be written as:

$$w_{ij}(n+1) = w_{ij}(n) - \mu e(n) \sum_{k=0}^{M-1} h_k v_j(n-k) f_j'(net_j(n-k))\, x(n-i-k) \ . \qquad (5)$$

$$i=0,1,\ldots I-1;\ j=0,1,\ldots J-2.$$

Where $net_j(n) = \sum_{i=0}^{I-1} w_{ij}(n)\, x(n-i)$, f(.) is a smoothing nonlinear activation function.

From Eqs. (4) and (5), it can be seen that in the AVC system, the reference signal x(n) is filtered by the estimate of the secondary path transfer function H (i.e., the reference signal is convolved with h_j). Therefore, this algorithm is called the FXBPNN algorithm.

2.2 FEBPNN Algorithm

In the FEBPNN algorithm, instead of filtering the reference signal, the error signal is filtered by the estimate of the secondary path transfer function. The filtered-error Least Mean Square (LMS) algorithm for a linear system has been derived by S.J.Elliott [5], on the basis of this, the FEBPNN algorithm is derived in the following analysis.

Step 1: Update the weights in the output layer of the neural network $v_j(n)$

Using Eqs. (3) and (4), then

$$\Delta v_j(n) = e(n) \sum_{i=0}^{M-1} h_i q_j(n-i) \ . \qquad (6)$$

Recalling Eq. (1), then the gradient estimate of the time-averaged square error in the output layer can be written as:

$$\Delta v_j(n) = \lim_{N\to\infty} \frac{1}{2N} \sum_{n=-N}^{N} [e(n) \sum_{i=0}^{M-1} h_i q_j(n-i)] \ . \qquad (7)$$

Let k = n-i so that n = k+i then, Eq. (7) can now be expressed as:

$$\Delta v_j(n) = \lim_{N\to\infty} \frac{1}{2N} \sum_{i=0}^{M-1} \sum_{k+i=-N}^{N} q_j(k)\, h_i e(k+i) \ . \qquad (8)$$

Noting that because i is always finite, the summation from –N to N as N tends to ∞ on the right hand side of Eq. (8) will be the same for k= – ∞ to ∞ as for k+i= – ∞ to ∞. At the same time, it is noted that an instantaneous estimate of Eq. (8) cannot be implemented with a causal system. This can be overcome by delaying both e(k+i) and $q_j(k)$ in Eq. (8) by M-1 samples. The final form of the FEBPNN algorithm for the output layer is obtained by adapting the controller's coefficients with an instantaneous version of the gradient estimate given by Eq. (8) delayed by M-1 samples, which can be written as:

$$v_j(n+1) = v_j(n) - \mu q_j(n-M+1)\sum_{i=0}^{M-1} h_{M-1-i} e(n-i) \ . \tag{9}$$

Step 2: Update the weights in the hidden layer of the neural network $w_{ij}(n)$

Using the similar method in Eq. (5), we can derive the formula for updating the weights in the hidden layer of the neural network $w_{ij}(n)$ as:

$$w_{ij}(n+1) = w_{ij}(n) - \\ \mu v_j(n-M+1) f_j'(net_j(n-M+1))\, x(n-i-M+1)\sum_{k=0}^{M-1} h_{M-1-k} e(n-k) \ . \tag{10}$$

2.3 Comparison of FXBPNN Algorithm and FEBPNN Algorithm

It is necessary to compare the two algorithms to show that the FEBPNN algorithm offer computational advantage over the FXBPNN algorithm. Considering the case with I input layer nodes and the J hidden layer nodes, for the FEBPNN algorithm using Eq. (9) and Eq. (10), the number of the multiplication operations required per sample to update the weights in the output layer is about M+J, and the number of the multiplication operations required per sample to update the weights in the hidden layer is about M+(2+I)(J-1). In contrast, for the FXBPNN algorithm using Eq. (4) and Eq. (5), the number of the multiplication operations required per sample to update the weights in the output layer and in the hidden layer is about JM+J and (3M+1)(J-1)I, respectively. For example when using M=256, J=15 and I=25, the FXBPNN and FEBPNN algorithms require about 273005 and 905 multiplications per sample for its implementation, respectively. So in terms of their computational efficiency, the FEBPNN algorithm is superior to the FXBPNN algorithm.

3 Experimental Implementation

3.1 Experimental Setup

Fig.3 and Fig.4 show the Experimental setup and the schematic diagram of the setup, respectively. The test-bed chosen was a simple three-dimensional cavity , the cavity had five rigid wall boundaries and the dimensions are 60cm×50cm×40cm. One aluminum plate was located at the top of the cavity and form the flexible boundary of the enclosure, the thickness of the plate is 0.5mm and was clamped around the cavity. A loudspeaker was used to generate the primary disturbance signal, which was also supplied to the controller as a reference signal. A piezoelectric ceramic(PZT) patch was surface bonded to the bottom of the aluminum plate to form the actuator and the dimension of the PZT patch is 50mm×25mm×0.5mm, the maximum supply voltage to the actuator was ±150V. An accelerometer was used as a sensor to measure plate strains and was bonded to the top of the aluminum plate. The signal processing board includes ADC/DAC, low-pass filters and preamplifiers. The heart of the control system was a TMS320VC33 DSP board, which was used to implement modal identification and control algorithm. The sampling rate was chosen to be 2k HZ.

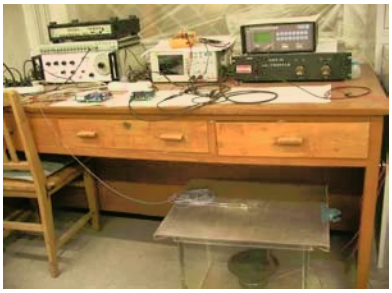


Fig. 3. The experimental setup

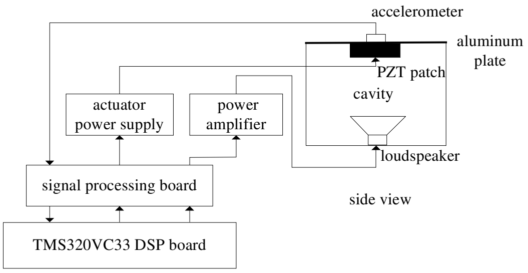


Fig. 4. The schematic diagram of the experimental setup

3.2 Experimental Results

The FEBPNN algorithm was used to suppress the primary disturbance. The number of neurons used in the neural network is 25-15-1. For comparative purposes, a 32-tap FIR filter controller adapted using the linear filtered-x LMS (FXLMS) algorithm was also implemented [4].

Case1: The first case investigated the nonlinear performance of the FXLMS and FEBPNN based controller. The primary disturbance signal was a 400HZ pure tone sinusoidal signal. The nonlinearity was introduced into the experimental arrangement by increasing the amplitude of the primary disturbance signal, and the preamplifier was driven to saturation, thus, the saturation nonlinearity was introduced into the AVC system [5]. The resultant error signal spectrum for nonlinear case is shown in Fig.5. From the results shown in Fig.5, it can be seen that the neural network controller performance is vastly superior to that of the linear controller. Not only was the primary tone level reduced, but also the harmonic signal in the spectrum was reduced.

Case 2: The second case investigated the tracking ability of the FEBPNN based controller. The primary disturbance signal was changed after the system had entered into steady-state phase. Fig.6 shows the error signal versus the number of iterations. Due to being Limited by the memory space of the DSP, it is not possible to record the entire change procedure of the error signal, so in Fig.6, the number of iterations is not continuous, the curve between 0 and 100 iterations shows the error signal when the AVC system turns off, the curve between 109500 and 110000 iterations shows the error signal when the AVC system turns on and has entered into steady-state phase, at this time, the primary disturbance signal is a 160HZ sinusoidal signal. When the number of iterations is equal to 110000, the primary disturbance signal was changed from a 160Hz sinusoidal signal to a 400Hz sinusoidal signal, the curve between 110000 and 110100 iterations shows the error signal during first 100 iterations after the primary disturbance signal was changed, the curve between 250000 and 250500 iterations shows the error signal after the system had entered into steady-state phase again. From the results shown in Fig.6, it can be seen that the system has better robust performance under change of the primary disturbance signal.

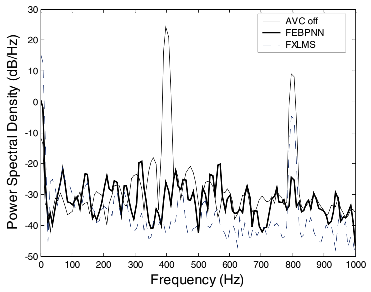


Fig. 5. The error signal spectrum for nonlinear case

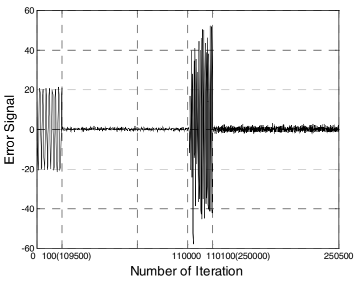


Fig. 6. The error signal versus number of iterations when the primary disturbance signal is changed

4 Conclusions

The neural network controller based on the FEBPNN algorithm has been developed for use in AVC system. Following this, the active vibration control of a three-dimensional cavity with a surface-bonded piezoelectric actuator and an accelerometer was studied experimentally. The experimental verification tests indicated that the neural network controller based on the FEBPNN algorithm was effective for a nonlinear system.

Acknowledgments

This research is supported by Science Funds of Educational Committee of Beijing City (KM200511232008) and Training Funds for Elitist of Beijing.

References

1. Lin, R.M., Nyang, K.M.: Analytical and Experimental Investigations on Vibrational Control Mechanisms for Flexible Active Structures. Smart Materials and Structures, **12** (2003) 500-506
2. Moheimani, S., Reza, O: A Survey of Recent Innovations in Vibration Damping and Control using Shunted Piezoelectric Transducers. IEEE Transactions On Control Systems Technology, **11** (2003) 482-494
3. Snyder, S.D., Tanaka, N.: Active Control of Vibration using a Neural Network. IEEE Trans On Neural Networks, **6** (1995) 819-828
4. Kuo, S.M., Morgan, D.R.: Active Noise Control Systems - Algorithms and DSP Implementations. New York (1996)
5. Elliott, S. J.: Signal Processing for Active Control. Academic Press, London (2001)

Compensating Modeling and Control for Friction Using RBF Adaptive Neural Networks

Yongfu Wang, Tianyou Chai, Lijie Zhao, and Ming Tie

Research Center of Automation, Northeastern University,
Shenyang, Liaoning 11004, China
Wyf3000@Sohu.com

Abstract. This paper presents an application of a radial basis functions adaptive neural networks for compensating the effects induced by the friction in mechanical system. An adaptive neural networks based on radial basis functions is employed, and a bound on the tracking error is derived from the analysis of the tracking error dynamics. The hybrid controller is a combination of a PD+G controller and a neural networks controller which compensates for nonlinear friction. The proposed scheme is simulated on a single link robot control system. The algorithm and simulations results are described.

1 Introduction

Friction is a commonly encountered phenomenon in mechanical systems. The effect of friction on machines is noticeable at very low velocities. At these speeds, motion tends to be intermittent. Commonly referred to as stick-up, intermittent motion can lead to overshoot and large-amplitude position or force limit cycling [1].

Various friction compensation techniques have been proposed for a servo mechanism to perform regulation tasks. These are high-gain PD control, friction model-based compensation, adaptive control [2],[3], etc. For regulation problems, high-gain PD control can reduce the steady-state position error, but often causes system instability. For friction mode-based compensation, it requires the exact friction model, which is difficult to obtain in practice.

Recently, [4] has developed a neural networks controller with special structure, in which a special class of nonsmooth activation functions is employed to compensate for friction, and a comparison with PD controller is presented in simulation. This paper combine the PD controller with standard RBF neural networks to solve the problem of a single link robot with friction concern. The rest of the paper is organized as follows: some basics for the friction model and RBF neural networks for controller design are reviewed in Section 2, and PD +G control based on radial basis functions neural networks is proposed in Section 3. In Section 4, the techniques mentioned above are applied to a single-link robot system, and simulation results are discussed. Finally, Section 5 gives the conclusions of the advocated method.

2 Preliminaries

Before getting into the problem of PD controller based on RBF neural networks compensation, we should explain friction model and describe RBF neural networks.

2.1 Friction Model

For the convenience of description, consider a 1-DOF robot system described by (1)

$$\begin{cases} J\ddot{q} + F + G = T_c \\ y = q \end{cases} . \tag{1}$$

where F denotes the friction , G denotes the gravity , and T_c is the joint torques.

Friction modeling has been studied by many researchers, but they don’t have uniform form. In paper [2], the friction model can be represented by the sum of the static torque T_{stick} , and the slip torque T_{slip} . A detailed friction model suitable for controller design is given in [3, 4, 5] as

$$F = [\alpha_0 + \alpha_1 e^{-\beta_1|\dot{q}|} + \alpha_2\left(1 - e^{-\beta_2|\dot{q}|}\right)]\operatorname{sgn}(\dot{q}). \tag{2}$$

where Coulomb friction is given by α_0, static friction is $(\alpha_0 + \alpha_1)$, and α_2 represents the viscous friction model. We can see that these friction models are highly nonlinear and discontinuous. This makes accurate friction compensation a difficult problem.

2.2 Radial Basis Functions Neural Networks

Radial basis functions neural networks belong to the class of multilayer feedforward neural networks [6],[7],[8]. They are characterized by a hidden layer made up of a collection of locally tuned processing units. Theses units can be seen as independent kernel nodes which compute the distance between their corresponding centroid and the input they received from the input units. The output of these kernel nodes is characterized by a nonlinear, radially symmetric activation function of that distance. Normally, the closer an input is to the center of the receptive field of one of these units which are fully connected to the units in the hidden layer. In other words, each output unit performs a weighted sum of the responses it receives from each hidden unit. RBF neural networks are modeled by the following relation.

$$y = \sum_{i=1}^{m} w_i \phi_i\left(\underline{x}\right). \tag{3}$$

where $\phi_i\left(\underline{x}\right)$ denotes the RBF, i.e.

$$\phi_i\left(\underline{x}\right) = \exp\left(-\frac{\left|\underline{x} - c_i\right|^2}{2\sigma_i^2}\right) / \sum_{j=1}^{m} \exp\left(-\frac{\left|\underline{x} - c_j\right|^2}{2\sigma_j^2}\right). \tag{4}$$

Eq.(3) can be rewritten by the following form

$$y = \underline{w}^T \phi(\underline{x}). \tag{5}$$

where $\underline{w} = [w_1, w_2, ..., w_m]^T$ is an adaptive parameter , and $\phi(\underline{x}) = [\phi_1(\underline{x}), ..., \phi_m(\underline{x})]^T$ is a regressive vector.

Some important results on RBF neural networks approximation indicate that RBF neural networks possess the so-called property of universal approximation. More precisely, for a given error tolerance $\in$ and an arbitrary nonlinear function $f(\cdot)$ continuous on a compact set $\Re$, there always exist a weight $\underline{w}$ and $\phi(\underline{x})$ such that

$$\max_{x\in\Re}\left|f(\cdot)-\underline{w}^T\phi(\underline{x})\right|<\in . \tag{6}$$

Theses results is the basis to justify the use of RBF neural networks in the nonlinear system approximation problem [8].

3 PD+G Control Based on RBF Neural Networks Compensation

If we know the friction and gravity terms in Eq. (1), the computed torque controller labeled T_c is given by the following equation [9]

$$T_c = K_p e + K_d \dot{e} + F + G . \tag{7}$$

where the e denotes the error of the position, and $\dot{e}$ denotes the error of the velocity.

If we use a RBF neural networks system to approximate friction model, the equation in (7) can rewritten in the form of

$$T_c = K_p e + K_d \dot{e} + T_\alpha + G . \tag{8}$$

where $T_\alpha = F = y = \underline{w}^T\phi(\underline{q}), \underline{x} = \underline{q} = [q \quad \dot{q}]^T$.

It is well known, the tracking errors in robot system is defined as follows:

$$e = q_d - q, \dot{e} = \dot{q}_d - \dot{q}, \ddot{e} = \ddot{q}_d - \ddot{q} . \tag{9}$$

where q_d , $\dot{q}_d$ and $\ddot{q}_d$ are the desired position, velocity and acceleration respectively.

Using Eq.(1), (8) and (9), the dynamics of the plant may be written as

$$\begin{cases} J\ddot{e} + K_d\dot{e} + K_p e = f - T_\alpha \\ f = J\ddot{q}_d + F \end{cases} . \tag{10}$$

Define $k_1 = J^{-1}K_d$, $k_2 = J^{-1}K_p$,The Equation (10) may be rewritten by

$$\ddot{e} + k_1\dot{e} + k_2 e = J^{-1}(f - T_\alpha). \tag{11}$$

where the nonlinear f is the function of friction. A general function f can be modeled by a RBF neural networks system as

$$f = \underline{w}^{*T}\phi(\underline{q}) + \varepsilon . \tag{12}$$

where the optimal approximation error ε is bounded by known constant ε_0 . The $\underline{w}^*$ is the following optimal parameter

$$\underline{w}^* = \operatorname{argmin}_{\underline{w}\in\Omega}\left[\sup_{\underline{w}\in\Omega_w}\left\|\hat{f}(\underline{q}\mid\underline{w}) - f(\underline{q})\right\|\right] . \tag{13}$$

The Equation (11) may be rewritten by

$$\ddot{e}+k_1\dot{e}+k_2 e=J^{-1}\left[\underline{w}^{*T}\phi(\underline{q})+\varepsilon-\underline{w}^T\phi(\underline{q})\right]. \tag{14}$$

Because of $\underline{\tilde{w}}=\underline{w}^{*}-\underline{w}$, we have

$$\ddot{e}+k_1\dot{e}+k_2 e=J^{-1}\left[\underline{\tilde{w}}^T\phi(\underline{q})+\varepsilon\right]. \tag{15}$$

Let $\underline{e}=\left[e,\dot{e}\right]^T$, then we can have the following state space equation

$$\dot{\underline{e}}=A\underline{e}+B\tilde{f}\ . \tag{16}$$

where, $A=\begin{pmatrix}0 & 1\\ -k_2 & -k_1\end{pmatrix}$, $B=\begin{pmatrix}0\\1\end{pmatrix}$, $\tilde{f}=J^{-1}\left[\underline{\tilde{w}}^T\phi(\underline{q})+\varepsilon\right]$.

If A is stable, there exists a positive-define symmetric matrix $P>0$ such that

$$A^T P+PA=-Q. \tag{17}$$

where Q a positive-define matrix.

Theorem 1: If we select the following PD+G controller based on adaptive RBF neural networks compensation in robot system described by eq.(18)and (19), the tracking error of the closed-loop system will be confined inside the compact set.

$$T_c=K_p e+K_d\dot{e}+T_\alpha+G. \tag{18}$$

$$\dot{\underline{w}}=\begin{cases}\gamma B^T P\,\underline{e}\,\phi(\underline{q})\ ,\ if\ \left(\|\underline{w}\|<M\right)\ or\left(\|\underline{w}\|=M\ \ and\ \gamma B^T P\,\underline{e}\,\phi(\underline{q})\geq 0\right)\\ P[\cdot]\ ,\ if\left(\|\underline{w}\|=M\,and\ \gamma B^T P\underline{e}\,\phi(\underline{q})<0\right)\end{cases}. \tag{19}$$

We omit the detail proof.

4 An Example of Simulation

Consider a single-link robot whose equations of motion are

$$\begin{cases}J\ddot{q}+d\dot{q}+mgl\cos(q)=\tau\\ y=q\end{cases}. \tag{20}$$

1) Select the system parameter

$m=1, l=1, J=4/3, \gamma=4/3$, $x(0)=[-0.5\quad 0]^T$, $w(0)=[-4\quad -3\quad -1\quad 2\quad 5\quad 1]^T$, $q_d=\sin(t), \dot{q}_d=\cos(t)$, $u\in[-12\quad +12]$.

2) Simulation results

The trajectories of the tracking error e is shown in Figs.1 in which the friction coefficient is 1.

The trajectories of the tracking error e is shown in Figs.2 in which the system using PD+G control causes instability when the friction coefficient is changed to 3.2.

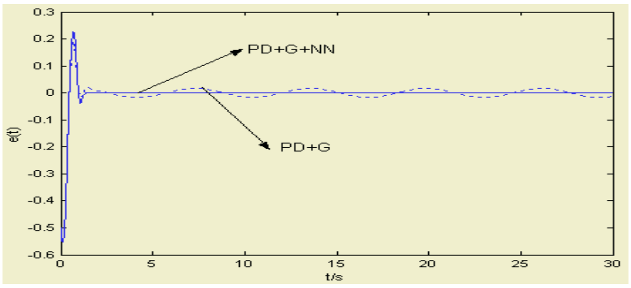


Fig. 1. Trajectory of tracking error *e* for *Case1*

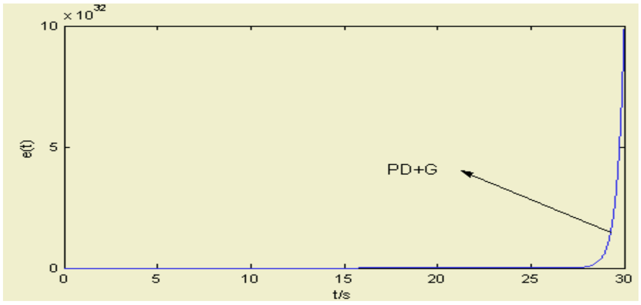


Fig. 2. Trajectory of tracking error *e* for *Case2*

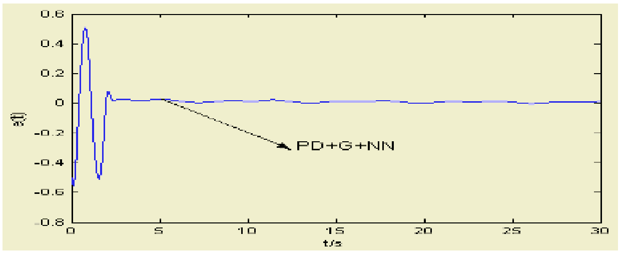


Fig. 3. Trajectory of tracking error *e* for *Case3*

The trajectories of the tracking error e is shown in Figs.3 in which the closed-loop system using PD+G controller based on RBF neural networks compensation still retain stability when the friction coefficient is changed to 3.2.

From the simulation results, it is seen that the PD+G controller based on RBF neural networks compensation has achieved the desired results.

We omit the detail comparison with [10] in simulation because of pages limits.

5 Conclusions

In this paper, we have developed a PD+G controller based on RBF neural networks which weights are tuned on-line, with no preliminary off-line training or learning phase, and derived a bound on the tracking error from the analysis of the tracking error dynamics. The technique presented can be extended to deal with other types of nonlinear friction.

Acknowledgments

The work is supported by the National Basic Research and Development of China (2002CB312200), the Hi-Tech Development Program of China (2004AA412030).

References

1. Kang, M.S.: Robust Digital Friction Compensation. Control Engineering Practice, **6** (1998) 359-367
2. Armstrong, B., Dupont P.C., Canudas De Wit.: A Survey of Models, Analysis Tools and Compensation Methods for Control of Machines With Friction. Automatica, **30** (1994) 1083-1138
3. Canudas de Wit, Noel, C.P., Aubin, A., Brogliato, B.: Adaptive Friction Compensation in Robot Manipulators: Low velocities. Int. J. Robot. Res., **10** (1991) 35-41
4. Selmic, R.R., Lewis, F.L.: Neural-network Approximation of Piecewise Continuous Functions: Application to Friction Compensation. IEEE Transactions on Neural Networks, **13** (2002) 745-750
5. Armstrong, B.: Friction: Experimental Determination, Modeling and Compensation. IEEE International Conference on Robotics and Automation, Philadelphia (1998) 1422-1427
6. Broomhead, D., Lowe., D.: Multivariable Interpolation and Adaptive Networks. Complex Systems, **2** (1988) 321-355
7. Sun, Y.L., Sarat, N.P.: Neuro-controller Design for Nonlinear Fighter Aircraft Maneuver Using Fully Tuned RBF Neural Networks. Automatica, **37** (2001) 1293-1301
8. Park, J., Sandberg, I.W.: Universal Approximation Using Radial-Basis-Function Neural Networks. Neural Computat., **3** (1990) 246-257
9. Fernando Reyes, Rafael Kelly. Experimental Evaluation of Model-Based Controllers on a Direct-Drive Robot Arm. Mechatronics, **11** (2001) 267-282
10. Wang, Y., Chai, T.: Compensating Modeling and Control for Friction Using Adaptive Fuzzy System. Proceedings of the IEEE CDC (2004) 5117-5121

Torque Control of Switched Reluctance Motors Based on Flexible Neural Network

Baoming Ge[1], Aníbal T. de Almeida[2], and Fernando J.T.E. Ferreira[2]

[1] School of Electrical Engineering, Beijing Jiaotong University, Beijing 100044, China
bm-ge@263.net

[2] Department of Electrical Engineering, University of Coimbra,3030 Coimbra, Portugal
adealmeida@isr.uc.pt

Abstract. Application of conventional neural network (NN) in modeling and control of switched reluctance motor (SRM) has been limited due to its structure of low degree of freedom, which results in a huge network with large numbers of neurons. In this paper, a flexible neural network (FNN), which uses flexible sigmoid function, is proposed to improve the learning ability of network, and the learning algorithm is derived. It greatly simplifies the network with fewer neurons and reduces iterative learning epochs. FNN based desired-current-waveform control for SRM, where FNN provides the inverse torque model, is presented. Simulation results verify the proposed method, and show that FNN gives better performances than conventional NN and the torque output of the control system has a very small ripple.

1 Introduction

Switched reluctance motor (SRM) has been applied to various kinds of variable-speed and servo-type drives. However, classical linear controllers could not deal properly with the high nonlinearity of SRM electromagnetic characteristics, so the ripple in the torque profile was still pretty high. Neural network (NN) with nonlinear functional approximation ability has been researched as a promising approach to solve SRM control and modeling problems [1]. However, the low flexible structure of conventional NN limits its learning ability and adaptation, which makes the constructed NN employ a vast numbers of neurons for an expected performance. Huge complex network increases computational burden in real time applications, especially for SRM with fast dynamic. Flexible neural network (FNN) in [2] and [3] improves the flexibility of network, but it is still limited due to only one parameter changeable in activation function. In this paper, a new FNN with the activation function of two parameters changeable, where the flexible neurons will flexibly change the shape of each unit to adapt its role in learning process, is proposed to overcome this difficulty. Application of the proposed FNN to torque control of SRM verifies the new method, and controller needs fewer units due to the improved learning ability, which reduces computational burden in real time and learning time.

2 Flexible Neural Network

2.1 Proposed Flexible Sigmoid Function

The sigmoid function with parameter changeable is called as flexible sigmoid function (FSF). In [2] and [3], FSF shown in (1) was proposed, which gives flexibility to neural network structure by adjusting parameter *a*.

$$f(x,a)=\frac{1-e^{-2xa}}{a(1+e^{-2xa})}. \tag{1}$$

However, its flexibility is limited because the slope of the function is nearly unchangeable within a large range of variable *x* near zero, although the size of the function is changed by selecting the parameter *a*. This property can be verified by

$$\frac{\partial f(x,a)}{\partial x}=1-a^2 f^2(x,a)=1-\left(\frac{1-e^{-2xa}}{1+e^{-2xa}}\right)^2. \tag{2}$$

Therefore, a new FSF in (3) is proposed to obtain high flexible structure of FNN. The proposed FSF has more flexibility than (1) since two parameters changeable are used. Not only the size of the function is changeable, but also it has a changeable slope that can be verified by using (4).

$$f(x,a,b)=\frac{1-e^{-2xa}}{b(1+e^{-2xa})}. \tag{3}$$

$$\frac{\partial f(x,a,b)}{\partial x}=\frac{a}{b}[1-b^2 f^2(x,a,b)]=\frac{a}{b}[1-\left(\frac{1-e^{-2xa}}{1+e^{-2xa}}\right)^2]. \tag{4}$$

Consequently, training two parameters according to the supervisor signals will achieve the flexible size and slope of the function for each unit, which is expected to improve the learning ability of network.

2.2 Learning Algorithm of FNN

The parameters *a* and *b* are updated together with connection weight simultaneously to minimize the cost function given by

$$J=1/2\sum_{i=1}^{L}(d_i-o_i^M)^2, \tag{5}$$

where d_i represents supervisor signal, *L* is the node number of output layer, and o_i^M is output of output layer *M*.

The increments of connection weights and FSF parameters can be obtained by employing the gradient descent method as following

$$\Delta w_{i,j}^{k,k+1}=-\eta_1\frac{\partial J}{\partial w_{i,j}^{k,k+1}}, \tag{6}$$

$$\Delta a_i^k = -\eta_2 \frac{\partial J}{\partial a_i^k} \ , \tag{7}$$

$$\Delta b_i^k = -\eta_3 \frac{\partial J}{\partial b_i^k} \ , \tag{8}$$

where $w_{i,j}^{k,k+1}$, a_i^k , and b_i^k represent connection weight and FSF parameters; η_1, η_2 , and η_3 are learning rates given by small positive constants.

The partial derivative of *J* with respect to the parameter a_i^k is

$$\frac{\partial J}{\partial a_i^k} = \frac{\partial J}{\partial o_i^k}\frac{\partial o_i^k}{\partial a_i^k} = -\sigma_i^k \frac{\partial f(i_i^k, a_i^k, b_i^k)}{\partial a_i^k} \ , \tag{9}$$

where i_i^k is the FSF input of the *i*th unit in the layer *k*. In the output layer, we have

$$\sigma_i^M = -\frac{\partial J}{\partial o_i^M} = d_i - o_i^M \ , \tag{10}$$

and in the hidden layer *k* it will be

$$\sigma_i^k = \sum_l \sigma_l^{k+1} \frac{\partial f(i_l^{k+1}, a_l^{k+1}, b_l^{k+1})}{\partial i_l^{k+1}} w_{i,l}^{k,k+1} \ . \tag{11}$$

From (7), and (9)-(11), the learning algorithm to update FSF parameter *a* of the *i*th unit in the layer *k* is deduced as

$$a_i^k(t+1) = a_i^k(t) + \eta_2 \sigma_i^k \frac{\partial f(i_i^k, a_i^k, b_i^k)}{\partial a_i^k} \ . \tag{12}$$

The function parameter *b* is changed in the same way as *a*, its updated algorithm is

$$b_i^k(t+1) = b_i^k(t) + \eta_3 \sigma_i^k \frac{\partial f(i_i^k, a_i^k, b_i^k)}{\partial b_i^k} \ . \tag{13}$$

Also the adjustment of connection weight can be summarized as

$$w_{i,j}^{k,k+1}(t+1) = w_{i,j}^{k,k+1}(t) + \eta_1 \sigma_j^{k+1} \frac{\partial f(i_j^{k+1}, a_j^{k+1}, b_j^{k+1})}{\partial i_j^{k+1}} o_i^k \ , \tag{14}$$

where o_i^k denotes the output of the *i*th unit in the layer *k*.

3 FNN Based Control for SRM

FNN based torque control system for SRM is shown in Fig. 1, in which FNN model to determine the nonlinear inverse model of torque calculates the desired value of phase current that produces a desired phase torque. Torque sharing function (TSF) distributes the total torque demand among the phases and generates the required phase torque profiles. The current closed-loop controller with hysteresis band is used to regulate the actual phase current tracking the desired current waveform.

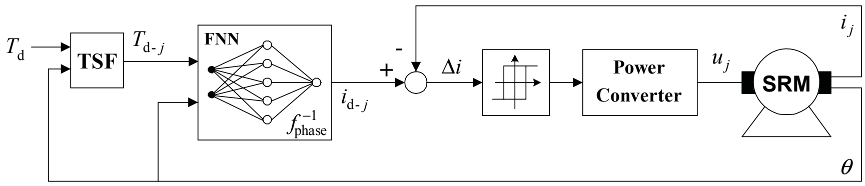


Fig. 1. FNN based torque control of SRM

3.1 FNN Based Inverse Model of Torque

For a typical m-phase SRM, its total torque is expressed as

$$T_e = \sum_{j=1}^{m} T_j(\theta, i_j) = \sum_{j=1}^{m} f_{\text{phase}}(\theta_j,\ i_j)\ , \tag{15}$$

$$\theta_j = \theta - (j-1)\frac{2\pi}{mN_r}\ , \tag{16}$$

where $T_j(\theta, i_j)$ and i_j are the phase torque and current of phase j, N_r is the number of rotor poles, θ is the angular position of the rotor. $f_{\text{phase}}(\theta_j, i_j)$ represents an unknown nonlinear function of phase torque, then its inverse model is written as

$$i_j = f_{\text{phase}}^{-1}(\theta_j, T_j)\ . \tag{17}$$

FNN model used to determine the nonlinear inverse model of torque has three layers, including a hidden layer with 12 neurons. The input vectors of two nodes are the rotor position and the phase torque. One node of output is the phase current. The proposed FSF is used in hidden layer, and a linear function is enough for output layer. Connection weight, FSF parameters a and b are initialized randomly. The measured data from a four-phase SRM of 7.5kW are used as sample space. The input data are obtained by using increment of 1N.m for phase torque and increment of 2° for rotor position from 0° to 30°, but the position points of 0° and 30° are not included in training process due to their singularity.

3.2 Simulation Results

Simulation results are illustrated in Figs. 2 and 3. Fig.2 gives the sum-squared network error of the FNN based torque inverse model, which shows that the high learning ability of FNN makes the network error reduce to 0.0001 after 415 epochs. Fig. 3 shows the response of total torque, where SRM with the load torque of 30N.m operates at the speed of 30rad/s. It can be seen that the torque ripple keeps very small, which is mainly due to the accurate FNN model. In our research, a conventional NN based torque inverse model is constructed to compare with the proposed FNN, where the same data are used to train the network. As a result, 70 neurons of hidden layer are necessary for a three-layer network to achieve the same network error. Moreover, huge network structure slows down its convergence rate and makes it take 3089 epochs for the same precise, as shown in Fig. 2. The same accurate torque responses are

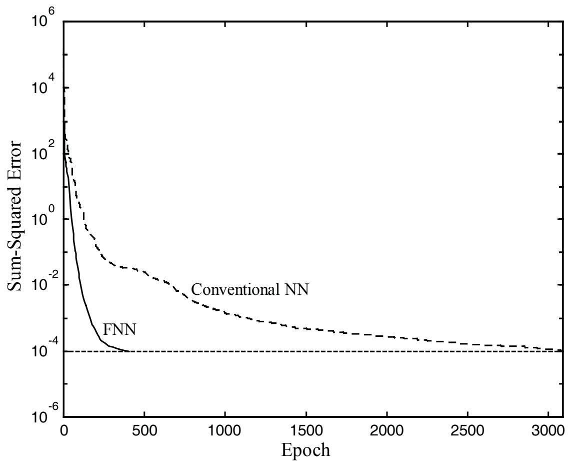


Fig. 2. Error during the learning process

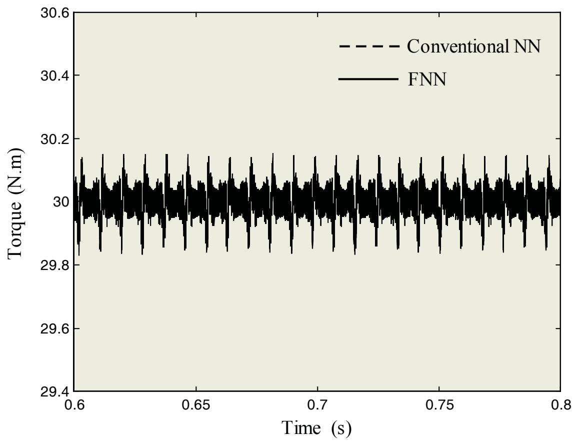


Fig. 3. Actual torque produced by SRM

produced in Fig.3 when both the proposed FNN and conventional NN models are respectively applied to torque control of SRM, because two network models have the same precision by offline learning. But the conventional NN based system shows its computational burden with time consuming due to huge complex network. However, The flexible structure of proposed FNN with three degrees of freedom enhances its learning ability with fewer neurons.

4 Conclusion

A new flexible neural network was proposed to improve the adaptive ability to complicated problems and learning algorithm was derived. The proposed FNN was applied to modeling and control of SRM with high nonlinear characteristic. The simulation results verified the validity of proposed method, and showed that FNN gave better characteristic than the conventional NN and the torque-ripple minimization of SRM control system was ideally achieved.

Acknowledgments

This work was supported in part by the Key Project of Chinese Ministry of Education (NO.2004104051), and in part by the Special Science Foundation of Beijing Jiaotong University under Grant 2003SM013. The authors would like to thank them for the financial supports.

References

1. Rahman, K. M., Gopalakrishnan, S., Fahimi, B., Velayutham Rajarathnam, A., Ehsani, M.: Optimized Torque Control of Switched Reluctance Motor at All Operational Regimes using Neural Network. IEEE Transactions on Industry Applications, **37** (2001) 904-913
2. Teshnehlab, M., Watanabe, K.: Intelligent Control Based on Flexible Neural Networks. Kluwer Publishers (1999)
3. Bevrani, H.: A Novel Approach for Power System Load Frequency Controller Design. Transmission and Distribution Conference and Exhibition 2002, Asia Pacific. IEEE/PES. **1** (2002) 184-189

Position Control for PM Synchronous Motor Using Fuzzy Neural Network

Jun Wang[1,2], Hong Peng[2], and Xiao Jian[1]

[1] College of Electrical Engineering,
Southwest Jiaotong University, Chengdu, Sichuan 610031, China
jiayu@mail.sc.cninfo.net
[2] Xihua University, Chengdu, Sichuan 610039, China

Abstract. A robust position controller with linear-quadratic (LQ) method and fuzzy neural-network (FNN) technique for permanent magnet synchronous motor (PMSM) is presented in this paper. An FNN controller is implemented to preserve a favorable model-following characteristics under various operating conditions. The performance of the proposed drive is investigated both in experiment and simulation at different condition.

1 Introduction

Linear state feedback control is theoretically an attractive method for controlling a linear plant represented by a state-space model. The method has the full flexibility of shaping the dynamics of the closed-loop system to meet the desired specification. Techniques such as a pole placement or linear-quadratic (LQ) method can be used to achieve the designed goals. However, these have the main problem that the desired response may not be obtained once the external disturbance and/or the parameters uncertainty exists.In recent years, much research has been on the FNN system and its application, which combines the capability of reasoning in handling uncertain information and the capability of neural networks in learning from processes [1]-[3]. For instance, Y.C. chen and C.C. Teng [1] proposed a model reference control structure using an FNN controller, which is trained on-line using an FNN identifier; Faa-Jeng Lin [2] described FNN sliding-mode position controller for induction servo motor drive; R.-J.Wai, F.-J.Lin [3] introduced integral-proportion controller with FNN controller to control a nonlinear objects.In this study, it is designed by combined the LQ method and the FNN method. LQ method is used to decide the demand feedback again to shape the dynamics and to meet the requirement of the performance index at the robustness for the PMSM position control system. Finally, simulation experimental results due to periodic step command were provided to verify the effectiveness of the proposed control system.

2 The Position Controller Design for PMSM by LQ

The basic principle in controlling a PMSM drive is based on field orientation through decoupling the quadrate axis (q axis) and direct axis (d axis). For a permanent magnet machine, if d axis current i_d is at zero, PMSM becomes a reduced-order linear time-invariant system as follows.

$$pi_q = (v_q - R_s i_q - \omega_r K_b)/L_q \tag{1}$$

$$p\omega_r = P(T_e - T_l)/2J - B\omega_r/J \tag{2}$$

$$T_e = K_T i_q \tag{3}$$

where $K_T = 3P.\lambda_M/4$ and $K_b = \lambda_M.v_d$ and v_q are the d, q-axis stator voltages, i_d and i_q are the d-q-axis stator currents, L_q is the q-axis inductance, T_e is the electromagnetic torque, R_s and ω_r are the stator resistance and the rotor speed, respectively. P is the pole number of the motor, p is the differential operator, T_l is the load torque. In (3), since the generated motor torque is linearly proportional to the q-axis current, the maximum torque per ampere can be achieved.

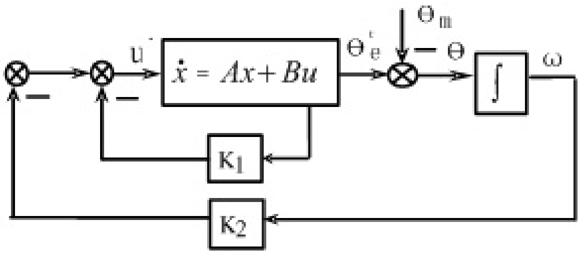


Fig. 1. The block diagram of a state variable feedback control system

To design a desired controller using the LQ method, the system must first be expressed in the state-space form. (2) can be expressed as

$$\begin{pmatrix} \dot{\theta}_e \\ \dot{\omega} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 0 & -\frac{B}{J} \end{pmatrix} \begin{pmatrix} \theta_e \\ \omega \end{pmatrix} + \begin{pmatrix} 0 \\ \frac{P}{2J} \end{pmatrix} T_e - \begin{pmatrix} 0 \\ \frac{P}{2J} \end{pmatrix} T_l \tag{4}$$

where $\theta_e = \theta_m - \theta$, θ_m and θ are the desired position and the actual rotor position, respectively. There, we redefine the new state-space equation assumption the disturbance $T_1 = 0$ as

$$\dot{x} = Ax + Bu \tag{5}$$

where $A = \begin{pmatrix} 0 & 1 \\ 0 & -\frac{B}{J} \end{pmatrix}$, $x^T = [\theta_e, \omega_r]^T$, $b = [0, P/2J]^T$. Using the LQ method, it is to find an optimal control, u^*, minimizing the performance index J

$$J = \int_0^\infty (x^T Q x + u^T R u) dt \tag{6}$$

associated with the system (5). In (6), matrix R is position defined, and Q is

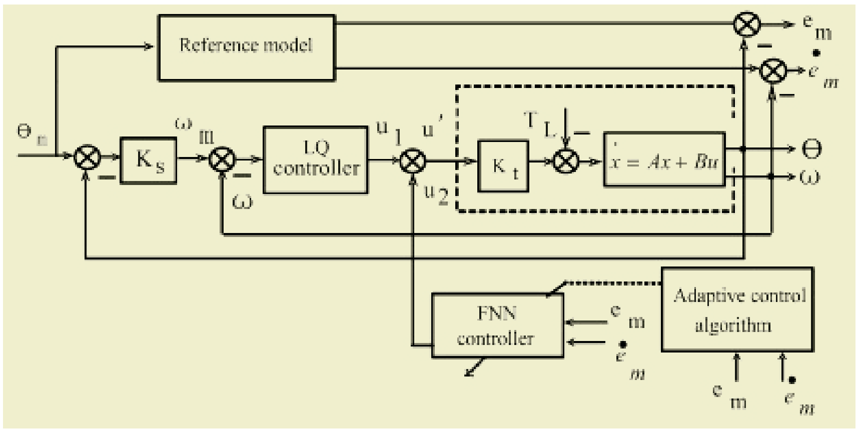


Fig. 2. The closed-loop drive system with FNN controller

nonnegative define. To find the optimal control law, u^*, the following Riccati equation

$$A^T P + PA - PbR^{-1}b^T P + Q = 0 \tag{7}$$

is first solved. Let P' be the solution of (7) and be nonnegative. Thus, to yield a minimum index of (6), the control u^* is defined as follows:

$$u^* = -K^T x = -(R^{-1}b^T P')^T x \tag{8}$$

and the feedback gain $K^T = [k_1, k_2]^T$ is expressed as $K = R^{-1}b^T P'$.when system (5) is under the control of (8), the resultant closed-loop dynamic are given by

$$\dot{x} = [A - bK^T]x = A_c x \tag{9}$$

The block diagram of a state variable feedback control system is shown in Fig.1.

3 The Proposed FNN Controller

The closed-loop drive system with the proposed closed-loop drive system with FNN controller is shown in Fig.2, which consists of a PMSM, LQ controller, FNN controller and adaptive reference model. The proposed FNN controller is augmented to the LQ control system to preserve the desired command tracking response under the occurrence of uncertainties. The control law is expressed as

$$u^* = u_1 + u_2 \tag{10}$$

where u_1 is the output of the LQ position controller, and u_2 is the output of the FNN controller. Since the system shown in Fig.1 is three-order system to meet the demanded system performance, the desired specifications of position step command tracking usually are: (1) no steady-state error; (2) no overshoot;

(3) desired risen time, t_{re}. So, the designed transfer function of reference model is

$$\frac{\theta(s)}{\theta^*(s)}|_{T_L(x)=0} = \frac{K_1}{s^3 + a_2 s^2 + a_1 s + a_0} \tag{11}$$

where the parameters K_1, a_0, a_1 and a_2 are constant determined by the desired specifications.

The input signals of the controller are chosen as error e_m between the reference position θ_m and actual position θ and the error $\dot{e}_m$ between the reference speed ω_m and actual speed ω which is described as

$$e_m = \theta_m - \theta, \dot{e}_m = \omega_m - \omega \tag{12}$$

A four-layer FNN as shown in Fig.3 is adopted to implement the proposed controller. Node numbers in the input, a membership, a rule and output layers are two, six, nine and one respectively.

Layer 1: input layer. The input nodes are written as

$$o_1^1 = e_m, o_2^1 = \dot{e}_m \tag{13}$$

Layer 2: membership layer. The connecting weight is $w_{ij}^2 (i = 1, \ldots, 4, j = 1, 2)$, the overall output is o_i^2.

$$net_i^2 = f(x_i^2), o_i^2 = f(net_i^2) \tag{14}$$

The Gaussian function is adopted as the exciting function expressed as

$$f(x) = -\frac{(x - m_{ij})^2}{(\sigma_{ij})^2} \tag{15}$$

where m_{ij} and σ_{ij} are the mean and the standard deviation of the Gaussian function. The weights between the input and membership layer are assumed to be unity.

Layer 3: rule layer. The output of the kth node is expressed as

$$net_k^3 = \prod_j w_{ij}^3 x_j, o_k^3 = net_k^3 \tag{16}$$

Layer 4: output layer. In this layer, the connecting weight is w_k^4, the overall output is O^4.

$$o^4 = \sum_k w_k^4 o_k^3 \tag{17}$$

To describe the on-line learning algorithm of the proposed FNN controller using the supervised gradient decent method, the performance index function J is defined as

$$J = \frac{1}{2}(\theta_m - \theta)^2 = \frac{1}{2} e_m^2 \tag{18}$$

Due to the real-time control system, adjusting the weight of FNN are adopted by the gradient decent algorithm. Since the gradient decent algorithm can be found easily in many references, it needn't be introduced here.

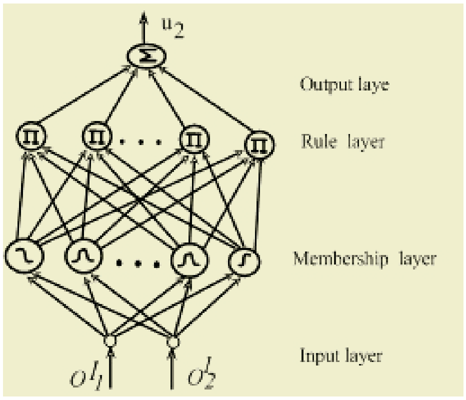


Fig. 3. Schematic diagram of the FNN controller

4 Simulation Results

The parameters of the LQ controller are designed when no load, nominal condition and the desired specifications are met. The reference model is chosen under desire specification. The nominal parameters of PMSM are: Rated power is 700w, rated speed is 1500rpm, number of poles is 4, magnetic flux is 0.175wb; Stator resistance is 2.875,stator induction is 0.085H. Desired rising time is $t_{re} = 0.3s$ and the sample time is $0.2ms$. The reference model is designed as

$$\frac{\theta(s)}{\theta^*(s)}|_{T_L(x)=0} = \frac{13800}{s^3 + 85s^2 + 1890s + 13800} \tag{19}$$

The efficacy of the proposed scheme is verified by computer simulation based on MATLAB/SIMULINK. The step command is a period step signal with magnitude of $2\pi rad$ and period of $3s$. To demonstrate the control performance of the proposed controller, the simulation results are compared under the same conditions for typical LQ controller and the proposed controller. The simulation conditions are the following two cases [4]. Fig.4 shows different position responses of typical LQ controller when no load, B and J are unchanged (case 1) shown in Fig.4(a) and a load of $5N.m$ is applied to the shaft, the inertia J and damping constant B of the rotor are increase by 50% (case 2) shown in Fig.4 (b). Fig.4 shows different position responses of the proposed FNN controller under the condition of case 1 in Fig.4 (c) and case 2 in Fig.c (d).

Case 1. $T_L = 0N.m, B' = B, J' = J$. Case 2. $T_L = 5N.m, B' = 5B, J' = 5J$.

From the simulation results, compared the typical LQ controller, the propose algorithm has smaller overshoot and shorter transient time. It can be already seen under the control of the proposed FNN controller that the degenerated and smooth responses under inertia variations and load disturbances are much improved with the augmentation of the proposed FNN controller.

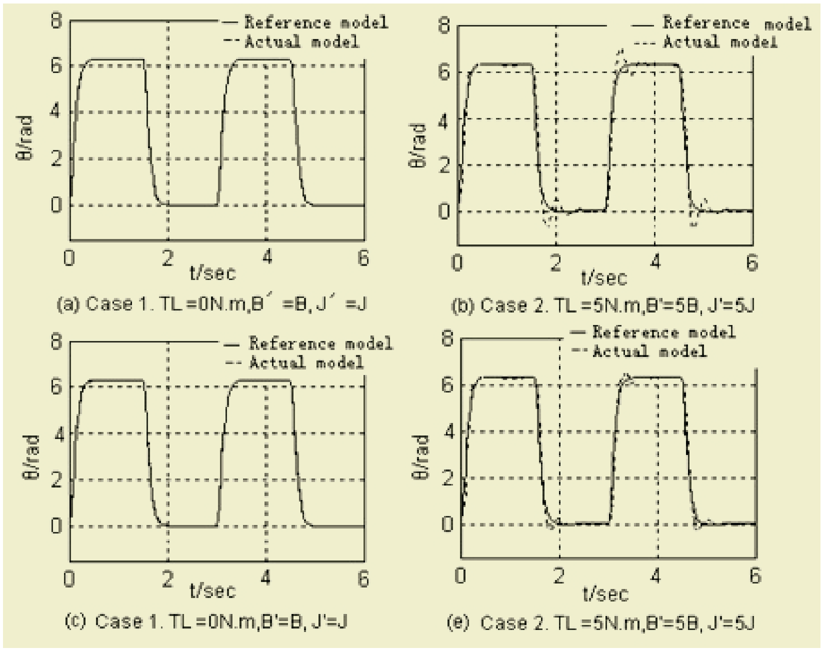


Fig. 4. The simulation position responses

5 Conclusion

A controller based LQ and FNN techniques for PMSM are theoretically analyzed and simulated in this paper. Major characters of the proposed controller are described on the following: (1) it makes system stronger robustness. (2) It has onliner learning capability that can deal with a larger range of parameter variations, and external disturbances. (3) More smooth responses can be obtained.

Acknowledgements

This work is supported by the doctor fund of Ministry of Education of the People's Republic of China (No. 20040613013) and the important project fund of the education department of Sichuan province, China (No. 0229957).

References

1. Chen, Y.C., Teng, C.C.: A Model Reference Control Structure Using a Fuzzy Neural Network. Fuzzy Sets and Systems, **73** (1995) 291-312
2. Lin, F.J., Wai, R.J., Chen, H.P.: A PM Synchronous Servo Motor Drive With an On-Line Trained Fuzzy Neural Network Controller. IEEE Trans. on Energy Conversion, **13** (1998) 319-325
3. Wai, R.J., Lin, F.J.: Fuzzy Neural Network Sliding-Mode Position Controller for Induction Serve Motor Drive, IEEE Proc. -Electr. Power Appl., **146** (1999) 297-308

SVM Based Lateral Control for Autonomous Vehicle

Hanqing Zhao, Tao Wu, Daxue Liu, Yang Chen, and Hangen He

Research Institute of Automation, National University of Defense Technology
Changsha, Hunan 410073, China
zhaohanqing@yahoo.com.cn, wutao.nudt@263.net, hehangen@yahoo.com

Abstract. SVMs have been very successful in pattern recognition and function approximation problems. In this paper, by collecting the human driving data, a data-driven lateral controller for autonomous vehicle based on SVMs is presented. Furthermore, according to the model of the vehicle, a simple method to improve the performance of this controller is introduced. The simulation results and experiments on the real vehicle show that the performance of the controller is satisfactory.

1 Introduction

The lateral control of autonomous vehicle has been widely studied for more than thirty years. The design methodology of it can be approached in two kinds: one is based on vehicle dynamic models and control theory, the other is based on learning human driving strategy and machine learning theories. Both of them have some advantages and disadvantages. The former approach can be easily analyzed in mathematics, but precise modeling and control are difficult due to the system nonlinearity and complexity. The latter one mostly relies on experiment data or some *a priori* human control knowledge so it's easy to be realized, but it is hard to reach higher control accuracy.

In this paper, a frame to build a data-driven lateral controller of autonomous vehicle based on support vector machines (SVMs) is proposed. In the frame, the controller is initialized by imitating a driver. After that, if the controller is failed to track the trace, it will be improved by adding some new support vectors. Simulations and real experiments show that the satisfactory controller can be designed in this frame.

2 SVM Based Human Control Learning

It is undeniable that a skilled driver is a good controller because he can drive a car in many complex environments. Thus, the controller can be initialized by approximating human control decisions or outputs.

Many methods of function approximation were used to solve learning control problem, such as neural networks [1],[2],[4],[8], fuzzy logic [3],[7], decision tree [5],[6] . The performance evaluation and optimization of learning are also studied [10]. At the same time, SVMs have been very successful in pattern recognition and function estimation. It is naturally to use SVMs to learn a control strategy. In [13], we have presented a method to design a controller based on SVMs. In this section, we will introduce this method in short.

2.1 Vehicle Model

In general, any dynamic system can be approximated through the difference equation [11]. That means:

$$u(k+1) = \Gamma[u(k),\cdots,u(k-n_u+1),x(k),\cdots,x(k-n_x+1),z(k)]. \tag{1}$$

Here, $\Gamma(\cdot)$ is some (possible nonlinear) map, $u(k)$ is the control vector, x(k) is the system state vector, z(k) is a vector describing the external environment at time step k.

The order of the dynamic system is given by the constants n_u and n_x, which may be infinite. Thus, a static model can abstract a dynamic system. For the case of the driving controller, the model will only require current state information. Thus the static map can be built as follows:

$$u(k+1) = \Gamma(u(k),x(k),z(k)). \tag{2}$$

To learn the driver's strategy, two features are chosen according to the preview control: one is the lateral error y_e, the other is the direction error ϕ_e. Fig.1 illustrates this model.

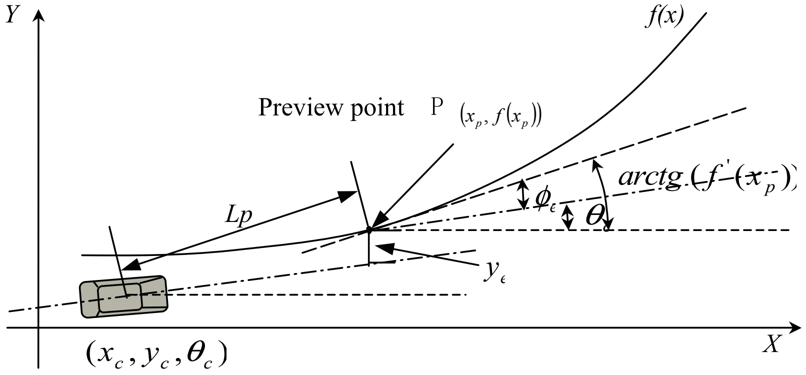


Fig. 1. Preview control model

In the Fig.1, $f(x)$ is a reference trajectory; Lp is the preview distance which is decided by the velocity V and preview time Tp: Lp=Tp*V.

The lateral error y_e and direction error ϕ_e in the vehicle coordinate can be calculated by the following equations:

$$y_e = (x_v - x_p)\cdot\sin\theta_c - (y_v - y_p)\cdot\cos\theta_c. \tag{3}$$

$$\phi_e = arctg\left(f'(x_p)\right) - \theta_c. \tag{4}$$

Where, θ_e is the heading angle. Thus, the control strategy of driving vehicle can be modeled as Eq. (5) instead of Eq. (2):

$$u = g(y_e, \phi_e). \tag{5}$$

Next, how to use support vector machines to approximate $g(y_e,\ \phi_e)$ will be introduced.

2.2 SVM Based Controller Model

In order to approximate the map (Eq.5), support vector machine (SVM) is chosen. Unlike other methods, which implement the empirical risk minimization principle, SVMs implement the sum of structural risk and empirical risk, so SVMs may result in better generalization performance than other methods. SVMs can be described as follows when they are used to fit the data:

$$f(x) = \sum_i (\alpha_i - \alpha_i^*) k(x, x_i) + b \cdot$$

Here, $\alpha_i, \alpha_i^* \ (i = 1, \cdots, n)$ is the optimal solution of the following quadratic optimization problem:

$$\begin{aligned} \max W(\alpha, \alpha^*; \xi, \xi^*) &= -\frac{1}{2}\sum_{i,j}(\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j)k(x_i, x_j) \\ &\quad + \sum_i y_i(\alpha_i^* - \alpha_i) - \varepsilon \sum_i (\alpha_i + \alpha_i^*) \\ s.t. \quad & \sum_i (\alpha_i^* - \alpha_i) = 0, \quad 0 \le \alpha, \alpha_i^* \le C, i = 1, \cdots, n \end{aligned}$$

Where, $k(\cdot,\cdot)$ is kernel function and it is chosen as RBF kernel in this paper:

$$k(x, y) = \exp(-\|x - y\|^2 / \sigma^2) \cdot$$

3 Data Selection for SVM Training and Improvement

3.1 Data Selection for SVM Training

As SVM is a data driven method to represent the knowledge, only “good” data can be built to be a good SVM. So it is very important to choose data.

First, to imitate the driver, we should choose the data. As human is not an accurate controller, the outputs (steer angles) cannot be the same when the input data are the same. It is reasonable to choose the mean value of outputs for the same inputs. Besides, in order to get a good controller, we should drive the car in many situations, which lead to so many samples to be collected. It is not a good idea to train SVM with all of the data because it is possible to build a SVM with so many support vectors. As we know that too many support vectors will cost so much computing time. So it is necessary to select a part of data as training data.

In this paper, K-mean clustering is used to select the training data. The value of K is decided by the computing performance of the controller. The data is selected as the training data if this data is nearest to the clustering center.

3.2 Data Selection for Improvement of SVM

Although a SVM has a good ability of generalization, it can not be proved that the controller based on this SVM can drive the car on any road with any degree of precision. So it is necessary to present a way to improve the performance of the controller. Traditional control (such as adaptive control) theory presents many methods to adjust some parameters of models to improve the performance of the controller. As the controller presented here is a data-driven controller, so the performance can be improved by adjusting the support vectors.

There are two ways to adjust support vectors: one is to add the new support vectors, the other is to adjust the old support vectors. In this paper, we will focus on the former.

If the vehicle is failed to track the road, the most probable reason is that some new situations occur and support vectors are too far to them. In this case, adding these new situations seems to be useful to improve the controller's performance if the vehicle faces to the same situations next time.

There are two problems to be solved when adding a support vector:

- Which one should be added as a new support vector?
- How to decide the output of the new support vector?

We start with the support vector set $M^{(0)}$ ($M^{(0)}$ is the initial support vectors learned from a driver). Suppose $M^{(k)}$ is the support vectors set at k step. At the k-th step we do two things:

First, given k-th support vector set $M^{(k)}$, we attempt to choose the point x(k) where the change ratio of tracking error is maximum.

Secondly, find the value u(k) that makes the maximum tracking error(e(k)) to be less than the threshold (E). u(k) can be decided according to the steer angle (▯) as follows:

```
Step1: if (|e(k)|>E & e(k)>0)
          if (  >0) u(k) = u(k)/2;
          else      u(k)=(u(k)-45)/2;
else if (|e(k)|>E & e(k)<= 0)
          if (  >0)  u(k)=(u(k)+45)/2;
          else       u(k) = u(k)/2;
else go to step 3.
Step 2: use u(k) to train a new controller and drive the car
with this controller, go to Step 1.
Step 3: The end.
```

```
Note:
  e(k)>0 means that the car is to the right of the road, and
  vice versa.
    >0 means that the car is turning right and vice versa.
```

4 Experiments

To verify the method presented in this paper, the controller is used to drive a real autonomous vehicle running on a rough road with some blind bends. The road is about 1800 meters long. The road map can be seen in Fig.2.

The autonomous vehicle was equipped with some sensors, including inertial measure unit (IMU) and differential global position system (DGPS), steering angle potentiometer, so that the position (longitude, latitude and height) and state (yaw, pitch, and roll) of the vehicle can be obtained.

First, a driver controls the vehicle running from point B to C and from C to B. During this time, 452 samples have been recorded.

Then, 10 training data were chosen from 452 samples by k-mean clustering method. Fig.3 is the tracking error with all the 452 training samples and 10 training data. It was shown that the SVM based on these chosen data can approximate that based on all the data very well.

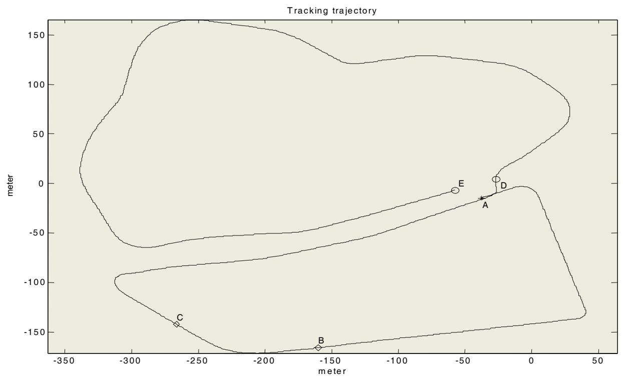


Fig. 2. Road map of a real test track

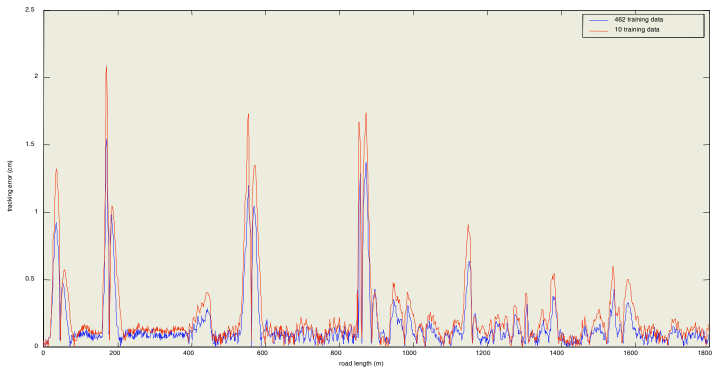
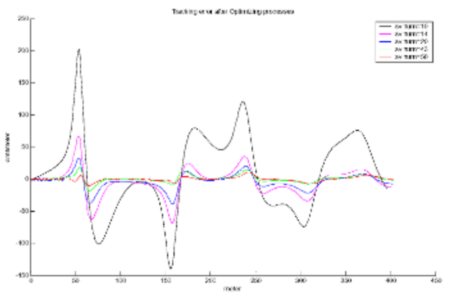


Fig. 3. Tracking errors of 452 training data and 10 training data

The experiment result also shows that the controller has a great ability of generalization. Although the training data was just from a section of the road (from B to C), the vehicle completed all the running. The tracking errors are shown in Fig. 6. There are three curves in this figure: the blue one is the tracking errors of PD controller; the red one is the tracking errors of SVM controller based on the 10 training data; the green one is that of an improved SVM controller. Obviously, the controller based on these 10 training data has less tracking error than that of PD controller.

However, the results need to be improved because there is so much error at each blind bend. Since the controller enjoyed its ability of generalization, we built a simulation environment to improve the controller. In the simulation environment, the road to be tracked is designed as Fig.4. Using the method introduced in section 3.2, the support vectors are turned to be more and more, but the tracking errors come to be less and less. The results are shown in Fig.5 and Table 1.



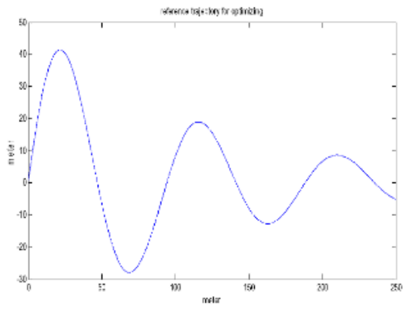


Fig. 4. Road map in simulation **Fig. 5.** Tracking errors and the SV number

Table 1. Max tracking errors and the number of support vectors

learning times	Tracking error range (cm)	Error limit (cm)	SV number
1	[-140.2237 , 202.0194]		10
2	[-68.0892 , 67.3015]	83	14
3	[-18.3467 , 18.6945]	20	28
4	[-11.1944 , 9.2833]	13	43
5	[-9.8585 , 8.7843]	10	56

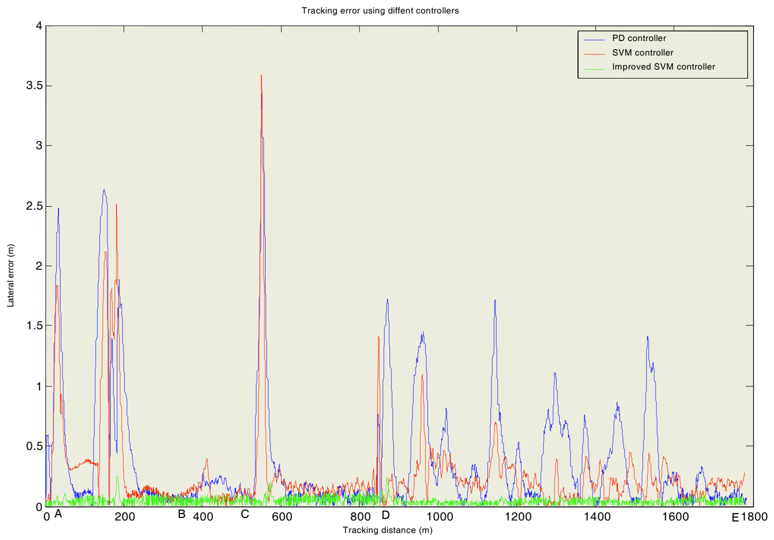


Fig. 6. SVM tracking errors and improved SVM tracking errors

Finally, the controller with 56 support vectors trained in this simulation environment is used to control the real vehicle on that rough road again and the results can be found in fig.6. The tracking error (improved) has been reduced to less than 0.25m. The performance of this controller is very satisfactory.

5 Conclusion

In this paper, we present a simple method to design and improve the lateral controller of autonomous vehicle based on SVMs. It is a general framework to design a data-driven controller for some control problems. Since this controller is data-driven, the performance of controllers can be improved easily by adding some support vectors. The experiments on the real vehicle indicated that it was very easy to get a satisfactory controller with this method.

Acknowledgment

This work is supported by National Nature Science Foundation (No.60234030).

References

1. Pomerleau, D.A.: Neural Network Perception for Mobile Robot Guidance. Kluwer Academic Publishers, Boston (1993)
2. Asada, H., Liu, S.: Transfer of Human Skills to Neural Net Robot Controller. Proceedings of the IEEE International Conference on Robotics and Automation (1991) 2442-2447
3. Jang, J.S.R., Sun, C.T.: Neuro-fuzzy Modeling and Control. The Proceedings of the IEEE, **83** (1995) 378-406
4. Nechyba, M.C.: Learning and Validation of Human Control Strategies. Ph.D. Thesis, The Robotics Institute, Vehiclenegie Mellon University (1998)
5. Sammut, C., Hurst, S., Kedzier, D.: Learning to Fly. Proceedings of the Ninth International Conference on Machine Learning, Aberdeen: Morgan Kaufmann (1992)
6. Urbancic, T.: Reconstructing Human Skill with Machine Learning. Proceedings of the 11th European Conference on Aritfical Intelligence (1994)
7. Kramer, U.: On the Application of Fuzzy Sets to the Analysis of the System Driver-vehicle-environment. Automatica, **21** (1985) 101-107
8. Veelenturf, L.P.J.: Analysis and Application of Artificial Neural Networks. Prentice Hall International
9. Velthuis, W.J.R.: Learning Feed-forward Control -Theory, Designed and Applications. Ph.D. Thesis, University of Twente, Enschede, Netherlands (2000)
10. Xu, Y.S., Song, J.Y., Nechyba, M.C., Yam, Y.: Performance Evaluation and Optimization of Human Control Strategy. Robotics and Autonomous Systems, **39** (2002) 19-36
11. Narendra, K.S., Parthasarathy, K.: Identification and Control of Dynamical Systems Using Neural Networks. IEEE Trans. on Neural Networks, **1** (1990) 4-27
12. Liu, J., Lan, H.: An Algorithm Using Predictive Control Theory for Automatic Drive. International Conference on Machine Learning and Cybernetics, Beijing, China (2002) 1601-1604
13. Wu, T., Zhao, H., He, H.: SVM Based Human Control Learning for ALV. International Conference on Robotics and Biomimetics, Shenyang, China (2004)

Control of Reusable Launch Vehicle Using Neuro-adaptive Approach*

Yong D. Song[1], Xiao H. Liao[1], M.D. Gheorghiu[1], Ran Zhang[1], and Yao Li[2]

[1] Department of Electrical Engineering, North Carolina A&T State University,
Greensboro, NC 27411, USA
songyd@ncat.edu

[2] Department of Electrical and Computer Engineering, University of Maryland,
College Park, MD 20742, USA
yaoli@umd.edu

Abstract. This work explores neural networks (NN) based control approach to nonlinear flight systems in the presence of disturbances and uncertainties. Neuro-adaptive control incorporating with two neural network (NN) units is proposed to cope with NN reconstruction error and other lumped system uncertainties. It is shown that with the proposed strategy, the angles of attack, sideslip and body-axis roll of the vehicle are effectively controlled. The method does not involve analytical estimation of the upper bounds on ideal weights, reconstruction error, or nonlinear functions. Stable on-line weight-tuning algorithms are derived based on Lyapunov's stability theory. Analyses and simulations confirm the effectiveness of the proposed control scheme.

1 Introduction

Reusable Launch Vehicle (RLV) system, as conceptually illustrated in Figure 1, holds great promise in future space exploration. The performance RLV is directly linked to its effective guidance and control - this must be included throughout design, including detailed aerodynamics and propulsion. Advanced control techniques make it possible to develop highly reliable and adaptive RLV trajectory control schemes. One of the most challenging issues related to the design of RLV flight control system lies in the wide range of flight conditions under which the vehicle operates. The resultant dynamic system of the vehicle is highly nonlinear and strongly coupled with uncertainties and disturbances, which calls for advanced control methodologies.

In this paper we present a method that incorporates two neural network (NN) units with adaptive control to achieve high performance trajectory tracking in angle of attack α, sideslip β and body-axis roll Φ of the RLV. Unlike most existing NN-based control methods, the proposed approach does not involve analytical estimation of the upper bounds on ideal weights, reconstruction error, or nonlinear functions, making the design and implementation procedures simpler and more feasible.

2 RLV Dynamic Model

The nonlinear RLV flight dynamic equations can be derived using the method similar to [1] as

* This work was supported in part by NSF under the HBCU-RISE program (Award number HRD-0450203).

$$\begin{bmatrix}\dot\alpha\\ \dot\beta\\ \dot\phi\end{bmatrix}=\frac{1}{mV}\begin{bmatrix}-\dfrac{\sin\alpha}{\cos\beta}[T+C_x(\alpha)\overline{q}S]+\dfrac{\cos\alpha}{\cos\beta}C_z(\alpha,\beta)\overline{q}S\\ -\cos\alpha\sin\beta[T+C_x(\alpha)\overline{q}S+\cos\beta C_y(\beta)\overline{q}S-\sin\alpha\sin\beta C_z(\alpha,\beta)\overline{q}S]\\ 0\end{bmatrix}+$$

$$+\frac{g}{V}\begin{bmatrix}\dfrac{1}{\cos\beta}(\sin\alpha\sin\theta+\cos\alpha\cos\phi\cos\theta)\\ \cos\alpha\sin\beta\sin\theta+\cos\beta\cos\theta\sin\phi-\sin\alpha\sin\beta\cos\phi\cos\theta\\ 0\end{bmatrix}+\begin{bmatrix}-\cos\alpha\tan\beta & 1 & -\sin\alpha\tan\beta\\ \sin\alpha & 0 & -\cos\alpha\\ 1 & \sin\phi\tan\theta & \cos\phi\tan\theta\end{bmatrix}\begin{bmatrix}p\\ q\\ r\end{bmatrix}$$

$$+\frac{\rho S}{4m}\begin{bmatrix}0 & -\dfrac{\sin\alpha}{\cos\beta}C_{xq}(\alpha)\overline{c}+\dfrac{\cos\alpha}{\cos\beta}C_q(\alpha)\overline{c} & 0\\ \cos\beta C_{yp}(\alpha)b & -\cos\alpha\sin\beta C_{xq}(\alpha)\overline{c}-\sin\alpha\sin\beta C_{zq}(\alpha)\overline{c} & \cos\beta C_{yr}(\alpha)b\\ 0 & 0 & 0\end{bmatrix}\begin{bmatrix}p\\ q\\ r\end{bmatrix} \tag{1}$$

$$+\frac{\rho VS}{2m}\begin{bmatrix}-\dfrac{\sin\alpha}{\cos\beta}C_{x\delta_e}(\alpha)+\dfrac{\cos\alpha}{\cos\beta}C_{z\delta_e}(\alpha,\beta) & 0 & 0\\ -\cos\alpha\sin\beta C_{x\delta_e}(\alpha)-\sin\alpha\sin\beta C_{ze}(\alpha,\beta) & \cos\beta C_{y\delta_a}(\beta) & \cos\beta C_{y\delta_r}(\beta)\\ 0 & 0 & 0\end{bmatrix}\begin{bmatrix}\delta_e\\ \delta_a\\ \delta_r\end{bmatrix}$$

$$\begin{bmatrix}\dot p\\ \dot q\\ \dot r\end{bmatrix}=\begin{bmatrix}I_2pq+I_1qr\\ I_5pr-I_6(p^2-r^2)\\ -I_2qr+I_8pq\end{bmatrix}+\begin{bmatrix}I_3C_l(\alpha,\beta)\overline{q}Sb+I_4C_n(\alpha,\beta)\overline{q}Sb\\ I_7C_m(\alpha)\overline{q}S\overline{c}\\ I_4C_l(\alpha,\beta)\overline{q}Sb+I_9C_n(\alpha,\beta)\overline{q}Sb\end{bmatrix}$$

$$+\frac{\rho VS}{4}\begin{bmatrix}I_3C_{lp}(\alpha)b+I_4C_{np}(\alpha)b & 0 & I_3C_{lr}(\alpha)b+I_4C_{nr}(\alpha)b\\ 0 & I_7C_{mq}(\alpha)\overline{c} & 0\\ I_4C_{lp}(\alpha)b+I_9C_{np}(\alpha)b & 0 & I_4C_{lr}(\alpha)b+I_9C_{nr}(\alpha)b\end{bmatrix}\begin{bmatrix}p\\ q\\ r\end{bmatrix} \tag{2}$$

$$+\overline{q}S\begin{bmatrix}0 & I_3C_{l\delta_a}(\alpha,\beta)b+I_4C_{n\delta_a}(\alpha,\beta)b & I_3C_{l\delta_r}(\alpha,\beta)b+I_4C_{n\delta_r}(\alpha,\beta)b\\ I_7C_{m\delta_e}(\alpha)\overline{c} & 0 & 0\\ 0 & I_4C_{l\delta_a}(\alpha,\beta)b+I_9C_{n\delta_a}(\alpha,\beta)b & I_4C_{l\delta_r}(\alpha,\beta)b+I_9C_{n\delta_r}(\alpha,\beta)b\end{bmatrix}\begin{bmatrix}\delta_e\\ \delta_a\\ \delta_r\end{bmatrix}$$

which can be expressed in terms of $x=\begin{bmatrix}\alpha & \beta & \phi\end{bmatrix}^T$ and $y=\begin{bmatrix}p & q & r\end{bmatrix}^T$ as follows,

$$\dot x = A(\alpha,\beta,\phi,\theta)+B(\alpha,\beta,\phi,\theta)y(p,q,r)+H(\alpha,\beta)u \tag{3}$$

$$\dot y = C(\alpha,\beta,p,q,r)+D(\alpha)y(p,q,r)+E(\alpha,\beta)u \tag{4}$$

with *A, B, C, D,* and *E* and *H* being defined accordingly. Extensive simulation studies show that the effect of the control gain matrix *H* in (3) is negligible, which leads to

$$\ddot x=\frac{d}{dt}A(.)+BC+[\frac{d}{dt}B(.)+BD]y+BEu \tag{5}$$

It is interesting to note that with proper selection of control surface both matrix *E* and *B* can be made invertible. This fact will be used in control design in next section.

3 Control Design

For the problem on hand, the most important design specification is to achieve asymptotic tracking of a known reference trajectory with the strongest possible form of stability. The control problem can be stated as follows: *design a control u such that the state variable x tracks the desired trajectory x^* as time tends to infinity.*

As the first step, we define the trajectory tracking error by $e = x - x^*$. Also we introduce a filtered variable $s = \dot{e} + k_0 e$, here $k_0 > 0$ is a design constant. Now rewriting (5) in terms of s to get

$$\begin{aligned} \dot{s} &= \frac{d}{dt}A(.) + BC + [\frac{d}{dt}B(.) + BD]y + BEu - \ddot{x}^* + k_0\dot{e} \\ &= f_0 + \Delta f(.) + Ru \end{aligned} \tag{6}$$

where

$$f_0 = BC + BDy - \ddot{x}^* + k_0\dot{e}\ ,\ \ \Delta f = \frac{d}{dt}A(.) + \frac{d}{dt}B(.)y\ ,\text{ and }\ R = BE$$

3.1 Neuro-adaptive Control

To design neuro-adaptive control scheme for the system we reconstruct f via NN units

$$f_{NN} = f_0 + W_I^T \psi_I \tag{7}$$

where f_0 was previously defined as the available part of f, $W_I \in R^{mxm}$ represents the optimal weight of the NN, $\psi_I \in R^m$ is the base function of NN whose selection typically include linear function, sigmoid function, hyperbolic tangent function, radial basis function, etc. Let the discrepancy between f and f_{NN} be denoted by: $\varepsilon = f - f_{NN}$, where ε is the reconstruction error. With the support of the approximate theory of multilayer neural networks, such an error can be made arbitrarily small provided that certain conditions (e.g., sufficiently large numbers of neurons, sufficient smoothness of L etc.) are satisfied. This has been the basis for many NN based control laws.

In reality, however, it should be noted that one can only use a network with a finite number of neurons and that the nonlinear function being reconstructed is not smooth enough. Thus, the approximation capabilities of the network can be guaranteed only on a subset of the entire plant state space [2]. In other words, during system operations, the reconstruction error sometimes is bounded by a constant, but at some other times it may not necessarily be confined by that number. For this reason, the NN reconstruction error should be treated carefully in control design because simply assuming that $||\varepsilon||$ is uniformly bounded by a constant does not warrant the success of the control scheme.

In the proposed control scheme, two NN units are included, with the first NN unit compensating the lumped nonlinearities and uncertainties, and the second NN unit attenuating the effect of the NN reconstruction error and other resulting uncertainties. The control scheme is given by:

$$u = R^{-1}(-k_0 s - \hat{f}_{NN} - u_c) \tag{8}$$

where the estimate neural network of the uncertain part $\hat{f}_{NN} = f_0 + \hat{W}_I^T \psi_I$. The terms used in the controller and estimation of the first neural network are: u_c a compensating signal to be specified. $k_0 > 0$ a constant chosen by the designer, $\hat{W}_I$ is the estimate of the ideal weight values. The control's action of the second NN unit is given by [3]:

$$u_c = \frac{\sum_{i=1}^{N} \psi_{II_i} \hat{W}_{II}^T \psi_{II}}{\sum_{i=1}^{N} \psi_{II_i} \|s\| + n} s \tag{9}$$

where $n > 0$ is a small number chosen by designer, $\hat{W}_{II}$ is the estimate of the ideal weight values, ψ_{II} is a basis function.

3.2 Stability Analysis

For all control systems, and for adaptive control systems in particular, stability is the primary requirement. In this section Lyapunov stability theory is used to prove that the proposed control scheme is able to achieve the control objective. From (6) we have

$$\dot{s} = -k_0 s + \tilde{W}_I^T \psi_I - u_c + \varepsilon \tag{10}$$

where $\tilde{W}_I = W_I - \hat{W}_I$ is the weight estimated error, and ε is the error to be attenuated by the second NN unit. The first step is to achieve the tuning algorithms for $\hat{W}_I$ and $\hat{W}_{II}$ by considering the following Lyapunov candidate function:

$$U = U_0 + U_1 + U_2 \tag{11}$$

with

$$\begin{aligned} U_0 &= \frac{1}{2} s^T s, \quad U_1 = \frac{1}{2g_I} tr(W_I - \hat{W}_I)^T (W_I - \hat{W}_I), \\ U_2 &= \frac{1}{2g_{II}} (W_{II} - \hat{W}_{II})^T (W_{II} - \hat{W}_{II}) \end{aligned} \tag{12}$$

where $g_I > 0$ and $g_{II} > 0$ are free design parameters. It can easily be seen that all the terms are positive. Using Equation (10), the derivative of the Lyapunov function will hold:

$$\dot{U} = -k_0 s^T s + tr[(W_I - \hat{W}_I)^T (\psi_I s^T - \frac{1}{g_I} \dot{\hat{W}}_I)] + Q \tag{13}$$

where the expression for Q is

$$Q = -s^T u_c + s^T e - \frac{1}{g_{II}} (W_{II} - \hat{W}_{II}) \dot{\hat{W}}_{II} \tag{14}$$

Before tuning the weights, one can focus on Q. After using some simple computations and the value of u_c given in (9), together with the following weight-tuning algorithms

$$\dot{\hat{W}}_I = -\sigma_1 \hat{W}_I + g_I \psi_I s^T \,,\; \dot{\hat{W}}_{II} = -\sigma_2 \hat{W}_{II} + g_{II} \frac{\sum_{i=1}^{N} \psi_{II_i} \left\| s^T s \right\| \psi_{II}}{\sum_{i=1}^{N} \psi_{II_i} \left\| s \right\| + n}$$

where $\sigma_1 > 0$ and $\sigma_2 > 0$ are design parameters chosen to prevent weight drift [3], we can show that

$$\dot{U} \le -k_0 s^T s + \frac{\sigma_1}{g_I} tr[(W_I - \hat{W}_I)^T \hat{W}_I] + \frac{\sigma_2}{g_{II}} (W_{II} - \hat{W}_{II})^T \hat{W}_{II} + nW_{II_m} \tag{15}$$

After some easy computations, the Equation (15) can further be rewritten as:

$$\begin{aligned} \dot{U} &\le -k_0 s^T s - \frac{\sigma_1}{2g_I} tr(W_I - \hat{W}_I)^T (W_I - \hat{W}_I) - \frac{\sigma_2}{2g_{II}} (W_{II} - \hat{W}_{II})^T (W_{II} - \hat{W}_{II}) + \xi \\ &= -\lambda_0 V_0 - \lambda_1 V_1 - \lambda_2 V_2 + \xi \le -\lambda_{\min} U + \xi \end{aligned} \tag{16}$$

where the new terms are

$$\xi = nW_{II_m} + \frac{\sigma_1}{2g_I} tr(W_I^T W_I) + \frac{\sigma_{II}}{2g_{II}} W_{II}^T W_{II} \,,\; \lambda_0 = 2k_0 \;\; \lambda_1 = \sigma_1$$
$$\lambda_2 = \sigma_2 \;\; \lambda_{\min} = \min(\lambda_0 \lambda_1 \lambda_2)$$

are constants. From Equation (16) it can be seen that $U \to \frac{\xi}{\lambda_{\min}}$ as $t \to \infty$, which implies that $\|s\|, \|\hat{W}_I\|$ *and* $\|\hat{W}_{II}\|$ are semi-globally uniformly ultimately bounded [3].

4 Simulation Study

In this section, a series of simulations are presented to illustrate the effectiveness of the proposed neuro-adaptive control in tracking a desired angle of attack α, sideslip β and body-axis roll Φ of the reusable launch vehicle. The aerodynamic coefficients used for simulation were collected from a generic RLV model as listed in the appendix. The basis functions for the first NN unit are chosen as:

$$\psi_I(c_{1i}, X, \gamma) = \frac{1 - e^{-\lambda(\|X\| - c_{1i})^2}}{1 + e^{-\lambda(\|X\| - c_{1i})^2}} \,, \quad (c_{1i} = 0.2, 0.4, \ldots, \; \|X\| = \begin{bmatrix} \alpha \\ \beta \\ \phi \end{bmatrix})$$

where λ = *0.5*. The basis functions for the second NN unit are selected as:

$$\psi_{II}(c_{2i}, X, \lambda) = \frac{c_{2i} - c_{2i} e^{-\lambda(\|X\| - c_{2i})^2}}{\mu + e^{-\lambda(\|X\| - c_{2i})^2}} \,, \quad (c_{2i} = 0.3, 0.7, \ldots, \; \|X\| = \begin{bmatrix} \alpha \\ \beta \\ \phi \end{bmatrix})$$

where μ = *0.5*. Fifteen NN basis functions were used for each of the NN units: $\psi \in R^j$ $j = 15$ .All the weights are initialized at zero in the simulation. The following command values of α, β and Φ are applied:

$$\alpha^* = 0.8 + 0.1\sin\frac{t}{6} \,, \beta^* = 0.1 + 0.2e^{-t} \,,\; \phi^* = 0.8 + 0.01\cos t$$

The actual and desired paths are shown in Figure 3.3. The tracking error is presented in Figure 3.4 and the action of the control in Figure 3.5. The estimated weights for the first NN are presented in Figure 3.6. It can be seen that the tracking is achieved with small error and all the internal signals are smooth and bounded.

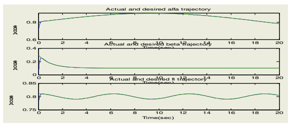


Fig. 1. Actual and desired trajectories

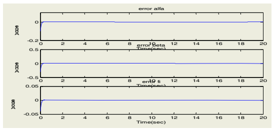


Fig. 2. Trajectory errors

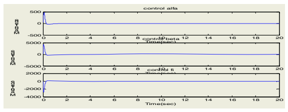


Fig. 3. Action of the controls

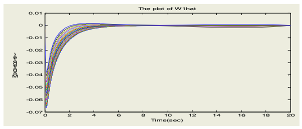


Fig. 4. Estimated weights of the first NN ($\hat{W}_I$)

5 Conclusion

This work proposed and analyzed a new approach to path tracking control of RLV. Compared with most existing methods, the proposed one does not involve manual estimation of the bounds on ideal weights, reconstruction error, or the nonlinear function being approximated. This means that there is no need to redesign or reprogram the control scheme for varying flight conditions. The salient feature of the proposed method also lies in its flexibility and simplicity in design and implementation.

References

1. Lee, T., and Kim, Y.: Nonlinear Adaptive Flight Control Using Backstepping and Neural Networks Controller. AIAA, **24** (2001)
2. Morelli, E.A.: Global Nonlinear Parametric Modeling with Application to F-16 Aerodynamics. Proceedings of the 1998 American Control Conference, IEEE Publications, Piscataway, NJ (1998) 997-1001
3. Song, Y.D.: Neuro-adaptive Control With Application To Robotic Systems. Journal of Robotics Systems , **14** (1997) 433-447
4. Song, Y.D., Xue, X.: Fault-tolerant control of composite systems with faulty dynamics. UNSW, Sydney, Australia, July (2002)
5. Kim, B. S., and Calise, A. J., "Nonlinear Flight Control Using Neural Networks. Journal of Guidance, Control and Dynamics, **20** (1997) 26-33
6. Peter, F. G., and Anthony J. C. Optimization of Launch Vehicle Ascent Trajectories with Path Constraints and Coast Arcs. Journal of Guidance, Control and Dynamics, **24** (2001)
7. Matthew, C. T., Daclan, G. B., and Ian, P.: Input/Output Conditioning of Robust Integrated Flight and Propulsion Controller. AIAA, **24** (2001)

A Neural Network Based on Biological Vision Learning and Its Application on Robot*

Ying Gao, Xiaodan Lu, and Liming Zhang

Dept. Electronic Engineering, Fudan University, Shanghai 200433, China
{gaoying,lmzhang}@fudan.edu.cn

Abstract. This paper proposes a neural network called "Hierarchical Overlapping Sensory Mapping (HOSM)", motivated by the structure of receptive fields in biological vision. To extract the features from these receptive fields, a method called Candid covariance-free Incremental Principal Component Analysis (CCIPCA) is used to automatically develop a set of orthogonal filters. An application of HOSM on a robot with eyes shows that the HOSM algorithm can pay attention to different targets and get its cognition for different environments in real time.

1 Introduction

As early as 1962, Hebel and Wiesel found out that each neuron on biological visual pathways only connects its neighbor region called receptive field. They can extract features from their corresponding receptive field. The receptive fields of adjacent neurons are overlapped and the neurons can cover the whole vision field. The visual pathway is consisted of multi-layer neurons. Their receptive fields of each layer have the same size. From lower layer to higher layer, the size of the receptive fields is becoming larger[1][2]. This structure can detect object with any size or in any location.

Motivated by the model, N Zhang and J Weng proposed a vision network called Staggered Hierarchical Mapping (SHM) in 2002[3]. As shown in Fig. 1, it is consisted of 9 layers of network. Layer 0 is the input image. Each square denotes a neuron. Each neuron only gets its inputs from a restricted region in the previous layer. When the layer is higher, the size of the receptive field is larger. SHM well simulates the architecture of visual pathway, but its flaw is that the extracted features vary with different receptive fields. If the location or size of the target changes, the responses of the corresponding receptive fields are completely different. Such output can hardly be used for pattern recognition. In [3], SHM manages only face recognition with simple occlusion. After supervised learning, the recognition rate reaches only 92.86%. But the detection for target with any size or in any location claimed in [3], can not be put into practice. Also motivated by the [1]biological visual pathway, we propose a neural network of global sensory mapping called HOSM, in which the input image is reduced in scale each layer and the receptive fields can cover any position inside the visual field with any size. All the receptive fields have the same group of filters that are obtained by learning of all the

* This research is supported by the National Natural Science Foundation of China (NSF 60171036) and the Shanghai Science and Technology Committee (No. 045115020).

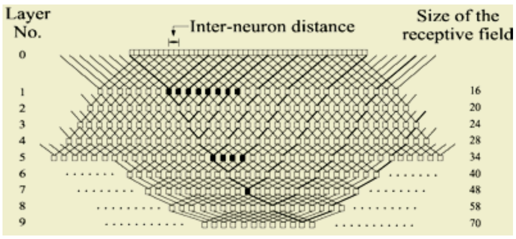


Fig. 1. The architecture of SHM

receptive fields' inputs. The network can extract *k* local features from the inputs. Thus wherever the target's location is, it could be detected and recognized.

In section 2 and section 3 the architecture of HOSM and its developmental algorithm are presented. Section 4 shows its application on robot. Finally we give the results.

2 Hierarchical Overlapping Sensory Mapping

To improve SHM [3], we propose a sensory mapping network, in which the input image is divided into lots of sub-blocks with identical size. Each block is transformed into a vector $X^l \in R^m$, $l = 1, 2, \ldots n$. As shown in Fig. 2, the block marked upper left is consisted of 5 overlapping receptive fields(sub-blocks). The size of the receptive fields and the overlapped distance are determined by the resolution of the camera and input images' scale. The neurons of higher layer should cover a bigger range than the previous one, so the input image should be scaled down for higher layer, in which the overlapping sub-blocks still have the same size. The sub-blocks in the higher level are forming into vectors $X^{l_1} \in R^m, l_1 = 1, 2, \cdots n_1$. The same job is done to the level of three, four… until the input image is scaled down to almost the same size of the receptive field.

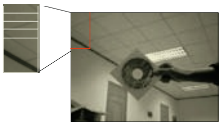


Fig. 2. The overlapping receptive fields

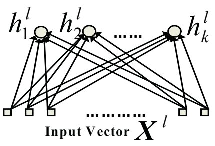


Fig. 3. Neural networks for weights' learning

Fig. 3 shows a neural network. Its inputs are sub-blocks of input image with a dimension of *m* and its outputs are *k* neurons. Fig. 4 shows all the possible sub-blocks in 4 levels: their position is kept organized and their receipted fields grow incrementally over each level until the entire image is approximately covered with one sub-block.

Suppose the overlapping sub-blocks of each level are organized as a part of the training set $\{X^1, X^2 \cdots X^n \cdots X^{n_1} \cdots X^{n_2} \cdots\}$. For video sequence, the training set is shown as $\{X^1(t), X^2(t) \cdots X^n(t) \cdots X^{n_1}(t) \cdots X^{n_2}(t) \cdots\}$ $t = 1, 2, \cdots$.Here *t* is the serial number of the frame. They are inputted to the neural network of Fig.3 sequentially in the training phase. We can obtain *k* eigenvectors of covariance matrix for these training samples as the weight vectors of neural network by CCIPCA, which will be introduced in section 3. In the testing period all the sub-blocks shown in Fig.4 are inputted to their corresponding

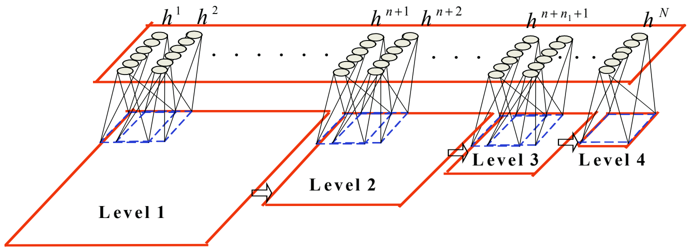


Fig. 4. The neural network of the Hierarchical Overlapping Sensory Mapping

networks with the same weights. After the projection of their individual network, we get *N·k* features corresponding to the output of *N* neural networks if the total number of the sub-blocks is *N*. Obviously, the flaw that "the extracted features vary with different receptive fields" of SHM[3] does not exist in our network. When an image is sensed, the *N·k* responses of the organized sub-blocks can be used for identifying any interested targets, whatever the position or size of the target is. For arbitrarily input region at any position with any size, these networks can find a receptive field approximately covers the region and get the same *k* responses.

In our experiments, the resolution of the input image is 640×480. The basic size of the receptive field is *m*=80×80 and the sub-block shifts 20 pixels each time. Then the total number amounts to *n*=29×21=609. To cover a larger range in the next level, we scale down the image to 480×360, then we use the same size *m* to cover the transformed image as input. The total sub-blocks in level 2 is n_1=315. Similar job is done to the third level and the fourth level. Finally there are 4 levels and the total sub-blocks for one input is *N*=1056. In the training phase, the *N* sub-blocks of each input image are transported to a network (shown in Fig. 3) producing *k* visual filters. The learning algorithm is CCIPCA[4], which is fast enough for online development. In the testing phase, the receptive fields of the entire input image are loaded in the network, producing responses as the final features.

3 Candid Covariance-Free Incremental Principal Component Analysis (CCIPCA)

In the training phase of the neural network, we apply an incremental PCA method called CCIPCA [4], which could extract visual eigenvectors incrementally from very long online real-time video stream. Principal component analysis (PCA) is a well-known technique but the computational approach to PCA needs computing the samples' covariance matrix, which requires that all the training images should be available before. It is very difficult for real time application. On the contrary, the CCIPCA algorithm could compute the principal components of a sequence of samples incrementally without estimating the covariance matrix.

For a sub-block from the input image, it is transformed into a column vector $\boldsymbol{X}^l$, denoted as $\boldsymbol{u}'(j), j=1,2,\cdots$, here *j* is the index of the sub-blocks. When the *p*th block appears, the mean of $\boldsymbol{u}'(j)$ is $\boldsymbol{m}_2(p)=\frac{1}{p}\sum_{i=1}^{p}\boldsymbol{u}'(j)$, and the covariance matrix is

$$A(p)=\frac{1}{p}\sum_{j=1}^{p}[\boldsymbol{u}'(j)-\boldsymbol{m}(p)][\boldsymbol{u}'(j)-\boldsymbol{m}(p)]^{\mathrm{T}}=\frac{1}{p}\sum_{j=1}^{p}\boldsymbol{u}(j)\boldsymbol{u}(j)^{\mathrm{T}} \quad (1)$$

Here, $\boldsymbol{u}(j)=(\boldsymbol{u}'(j)-\boldsymbol{m}(j))$. The ith eigenvector of $\boldsymbol{u}(j)$ satisfies $\lambda_i \boldsymbol{y}_i(p)$ = $A(p)\boldsymbol{y}_i(p)$. $\boldsymbol{y}_i(p)$ is the ith eigenvector and λ_i is the corresponding eigenvalue. To speed up the computation, the variable used for iteration in CCIPCA is $\lambda_i \boldsymbol{y}_i$. If the pth input blocks of image appears, we have

$$\boldsymbol{v}_i(p)=\lambda_i \boldsymbol{y}_i(p)=A(p)\boldsymbol{y}_i(p)\cdot \quad (2)$$

from (1) and (2), it could be deduced that

$$\boldsymbol{v}_i(p)=\frac{1}{p}\sum_{j=1}^{p}\boldsymbol{u}(j)\boldsymbol{u}(j)^{\mathrm{T}}\boldsymbol{y}_i(p)\cdot \quad (3)$$

Since all the eigenvectors are normalized, if we can estimate V_i, λ_i and $\boldsymbol{y}_i$ will be calculated as $\lambda_i=\|\boldsymbol{v}_i\|$ and $\boldsymbol{y}_i=\boldsymbol{v}_i/\|\boldsymbol{v}_i\|$. Change Eq. (3) to iteration form. We choose $\boldsymbol{y}_i(p)$ as $\boldsymbol{v}_i(p-1)/\|\boldsymbol{v}_i(p-1)\|$, which leads to the following incremental expression as the basic iteration formula of CCIPCA,

$$\boldsymbol{v}_i(p)=\frac{p-1}{p}\boldsymbol{v}_i(p-1)+\frac{1}{p}\boldsymbol{u}(p)\boldsymbol{u}(p)^{\mathrm{T}}\frac{\boldsymbol{v}_i(p-1)}{\|\boldsymbol{v}_i(p-1)\|}\cdot \quad (4)$$

where (p-1)/p is the weight for the last estimate and 1/p is the weight for the new data $\boldsymbol{u}(p)$. It is proven that, with the algorithm given by Eq. (4), $v_i \rightarrow \lambda_i \boldsymbol{y}_i$ when $p \rightarrow \infty$, where λ_i is the ith largest eigenvalue of the covariance matrix of { $\boldsymbol{u}(j)$ } and $\boldsymbol{y}_i$ is the corresponding eigenvector[4]. Eq. (4) can apply to all the eigenvectors. For example, to compute the eigenvector corresponding to the second maxima eigenvalue, we subtract the projection on the first order eigenvector from input as shown in Eq.(5),

$$\boldsymbol{u}_2(p)=\boldsymbol{u}_1(p)-\boldsymbol{u}_1^{\mathrm{T}}(p)\frac{\boldsymbol{v}_1(p)}{\|\boldsymbol{v}_1(p)\|}\frac{\boldsymbol{v}_1(p)}{\|\boldsymbol{v}_1(p)\|}\cdot \quad (5)$$

Where $\boldsymbol{u}_1(p)=\boldsymbol{u}(p)$. The residual $u_2(p)$, which is in the complementary space of $\boldsymbol{u}_1(p)$, serves as the input data to the Eq.(4). So the orthogonality is always guaranteed when the convergence is reached. Besides, since the real mean of the image data is unknown, the sample mean $\hat{\boldsymbol{m}}(p)$ could be incrementally estimated by

$$\hat{\boldsymbol{m}}(p)=\frac{p-1}{p}\boldsymbol{m}(p-1)+\frac{1}{p}\boldsymbol{u}'(p)\cdot \quad (6)$$

It has been proved that CCIPCA could quickly estimate the eigenvectors and converge very fast for high-dimensional image vectors[4]. Compared with the available IPCA algorithm, CCIPCA is faster and converges better. In our experiments for online estimation of the 10 eigenvectors of the input images with the resolution of 160×120, CCIPCA could handle it more than 60 frame/s.

4 Application on Robot

HOSM is a sensory mapping method, not a classification method. Classification in our experiment is performed by HDR [5], which use output of HOSM as input.

Hierarchical Discriminant Regression (HDR) is a new subspace regression tree for high-dimensional learning, which puts classification, regression and discrimination in one unit. No matter how high the dimension of the input data would be, it could classify and discriminate quickly. For more details, see paper [5].

The diagram in Fig. 5 shows the vision model applied on robot. With the sensory mapping done by HOSM, HDR will built a cognition network for classification and recognition and also gives its feedback for sensory mapping as attention selection. The HDR tree is like a network of knowledge which is learned autonomously by robot. When the tree is getting mature, the robot is getting more intelligent like human.

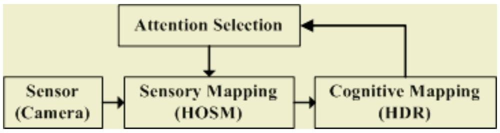


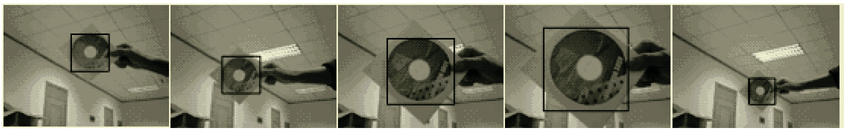
Fig. 5. The proposed Vision Model for Autonomous Developing Robot

5 Experiments

To test how practical the vision model can be, we organized experiments on a robot. The hardware platform is XEON P4 3.6 G, 2 GB memory，200 GB HD. In the training phase, video frames are presented to the system and CCIPCA is applied to develop the 30 weight vectors for all the receptive fields. During the experiments, object needed to be attentive will be put in front the camera, moving or stay still, while the receptive fields which fall onto the area of the target are marked for classification. In the testing phase, the same target is put in front of the camera again, moving randomly or stay still. Both the training and testing period are 30 seconds and the frame rate is 2 frame/s.

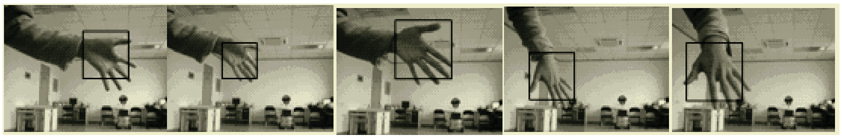
As depicted in Section 2, the resolution of the video frame is 640×480, the basic size of the receptive field is 80×80 and each block shifts 20 pixel. The resolution of the second, third and forth level is 480×360, 320×240 and 160×120. The extracted features are organized as groups and then sent to HDR engine as input. After building a cognitive tree in the training phase, the machine “recognizes” the target. In the testing phase, wherever the target moves , the machine could locate the target autonomously. The target we choose for experiments is a CD disk and a hand. As shown in Fig. 7 and 8, for each target, the recognition rate of the 60 frames within 30s is 96.6% and 98.3%.

It is proved in our experiments that the proposed visual neural network performs well for target recognition, however it changes its location. The model we proposed isn’t based on the movement detection of the target, so for the moving or still target, the robot can find it and locate it. Obviously, if the overlapped size between the adjacent receptive fields is larger and the ratio of reduced scale between levels is lower, the accuracy of the target location would be higher. But the computation complicity will increase highly. Considering the on-line intelligence development of a robot, the system should be able to deal with the sequential video frames. In our vision model, simulation shows that the system can train input video in real time with a speed of 2 frame/s, which is enough for simple object detection and robot control.



(a) the 10th frame(b) the 20th frame(c) the 30th frame(d) the 40th frame(e) the 50th frame

Fig. 6. CD disk Tracking in our Experiments



(a) the 10th frame (b) the 20th frame (c) the 30th frame (d) the 40th frame (e) the 50th frame

Fig. 7. Hand Tracking in our Experiments

6 Conclusion

In this paper we propose a neural network based on receptive fields, which could autonomously develop weights of orthogonal eigenvectors for input samples and extract features from any location and any size within the vision field of a robot. The structure of the vision sensory mapping is based on feature extraction and cognition, not on motion detection and not for special object. Experiment results show that our robot can see and locate target after letter time training, and can autonomously develop cognition targets of any kinds. The aim of our current research is building a practical system in which the robot could autonomously develop the cognition of the environment and any targets. We believe that will be a great progress in robot intelligence development.

References

1. Hubel, D., Wiesel, T.: Receptive Fields, Binocular Interaction and Functional Architecture in the Cat's Visual Cortex. J. of Physiology, **160** (1962) 106-154
2. Shou, T.: Brian Mechanisms of Visual Information Processing. Shanghai Science and Education Publishing House (1977)
3. Zhang, N., Weng, J., Zhang, Z.: A Developing Sensory Mapping for Robots. Development and Learning, 2002. Proceedings. The 2nd International Conference, (12-15 June 2002) 13-20
4. Weng, J., Zhang, Y., Hwang, W.S.: Candid Covariance-free Incremental Principal Component Analysis. IEEE Trans. Pattern Analysis and Machine Intelligence, **25** (2003) 1034-1040
5. Hwang, W.-S., Weng, J.: Hierarchical Discriminant Regression. IEEE Trans. on Pattern Analysis and Machine Intelligence, **22** (2000) 1277-1293

Discrete-Time Adaptive Controller Design for Robotic Manipulators via Neuro-fuzzy Dynamic Inversion*

Fuchun Sun, Yuangang Tang, Lee Li, and Zhonghang Yin

Dept. of Computer Science and Technology,
State Key Lab of Intelligent Technology & Systems, Tsinghua University,
Beijing 100084, China
chunsheng@tsinghua.edu.cn, tyg02@mails.tsinghua.edu.cn

Abstract: A stable discrete-time adaptive tracking controller using neuro–fuzzy (NF) dynamic inversion is proposed for a robotic manipulator with its dynamics approximated by a dynamic T-S fuzzy model. NF dynamic inversion is used to compensate for the robot inverse dynamics. By assigning the dynamics of the Dynamic NF (DNF) system, the dynamic performance of the robot control system can be guaranteed in the initial control stage. The discrete-time adaptive control composed of NF dynamic inversion and NF variable structure control (NF-VSC) is developed to stabilize the closed-loop system and ensure the high-quality tracking. The system stability and the convergence of tracking errors are guaranteed and effectiveness of the proposed control approach. is verified.

1 Introduction

Recently, stable NF adaptive control of nonlinear systems [1] has attracted intense research in the past decade. However, most researches in this field almost focus on those using static NF networks in the form of T-S fuzzy model [2], and there are few researches on stable adaptive control based on DNF models.

Dynamic inversion was first developed in aerospace field [3] and its application in neuro-adaptive control can be found in [4]. However, these approaches are for static neural networks (NNs), some problems subject to existing NF control still remain. To solve these problems, Sun *et al.* proposed stable adaptive control based on dynamic inversion using dynamic NNs [5]. This approach can exclude the assumption that the system state should be within the compact set and the system performance is guaranteed in the initial control stage. As the extension of [5], this paper is concerned with the DNF adaptive control for a robot modeled by a dynamic T-S fuzzy model. Firstly, the sector nonlinearity approach is used to model the robot dynamics[1]. The introduction of dynamic inversion makes the state of the DNF system strictly to be the interpolation between the initial state and the desired trajectory such that the system

* This work was jointly supported by the National Excellent Doctoral Dissertation Foundation (No:200041), National Natural Science Foundation of China (No: 60474025, 90405017) and National Key Project for Basic Research of China (No: G2002cb312205)

performance in the initial control stage can be improved by assigning parameters of the DNF system. The NF-VSC proposed by Sun [6] is introduced in the control structure to further improve the DNF learning convergence and stability.

2 NF Adaptive Control Based on Dynamic Inversion

2.1 Robot Fuzzy Modeling

The discrete-time DNF model for a robot [5] can be represented by

Rule i: If $z_1(k)$ is F_{i1}, $z_2(k)$ is F_{i2}, $\cdots, z_p(k)$ is F_{ip}

$$\text{Then } \boldsymbol{M}_i\dot{\boldsymbol{q}}(k+1)=\boldsymbol{A}_i\boldsymbol{x}(k)+\boldsymbol{b}_i+\boldsymbol{u}(k),\ i=1,\cdots,m\ . \tag{1}$$

where $F_{ij}(i=1,2,\cdots,m, j=1,2,\cdots,p)$ are fuzzy sets described by membership function μ_{ij}, $\boldsymbol{x}(k)=\left(\boldsymbol{q}^{\mathrm{T}}(k),\dot{\boldsymbol{q}}^{\mathrm{T}}(k)\right)^{\mathrm{T}}\in R^{2n}$ is the state vector with $\boldsymbol{q}(k)$ and $\dot{\boldsymbol{q}}(k)$ vectors of generalized coordinates and velocities, $\boldsymbol{u}(k)\in R^n$ is the input vector. m is the number of rules, and $z_1(k)\sim z_p(k)$ are measurable variables or nonlinear function terms.

2.2 NF Adaptive Control Algorithm

The following tracking error metric is defined for (1)

$$\boldsymbol{S}(k)=\boldsymbol{C}(\boldsymbol{x}(k)-\boldsymbol{x}_d(k))=\boldsymbol{\Lambda q}(k)+\dot{\boldsymbol{q}}(k)-\boldsymbol{C}\boldsymbol{x}_d(k)\ . \tag{2}$$

where $\boldsymbol{S}(k)=[s_1(k),\cdots,s_n(k)]^{\mathrm{T}}$, $\boldsymbol{x}(k)=[\boldsymbol{q}^{\mathrm{T}}(k),\dot{\boldsymbol{q}}^{\mathrm{T}}(k)]^{\mathrm{T}}$, $\boldsymbol{x}_d(k)=[\boldsymbol{q}_d^{\mathrm{T}}(k),\dot{\boldsymbol{q}}_d^{\mathrm{T}}(k)]^{\mathrm{T}}$ is the desired state trajectory to be tracked, and $\boldsymbol{C}=[\boldsymbol{\Lambda},\boldsymbol{I}]\in R^{n\times 2n}$, $\boldsymbol{\Lambda}=\boldsymbol{\Lambda}^{\mathrm{T}}>\boldsymbol{0}$. By using center-average defuzzifier, product inference, and singleton fuzzifier, the dynamic fuzzy system (1) can be written as below

$$\sum_{i=1}^{m}\theta_i(\boldsymbol{z}(k))\boldsymbol{M}_i\dot{\boldsymbol{q}}(k+1)=\sum_{i=1}^{m}\theta_i(\boldsymbol{z}(k))\left(\boldsymbol{A}_i\boldsymbol{x}(k)+\boldsymbol{b}_i\right)+\boldsymbol{u}(k)\ . \tag{3}$$

It is evident that $\boldsymbol{M}(\overline{\boldsymbol{q}}(k))=\sum_{i=1}^{m}\theta_i(\boldsymbol{z}(k))\boldsymbol{M}_i$. Substituting $\dot{\boldsymbol{q}}(k+1)=\boldsymbol{S}(k+1)-\boldsymbol{\Lambda q}(k+1)$ $+\boldsymbol{C}\boldsymbol{x}_d(k+1)$ into (3) gives

$$\boldsymbol{M}(\overline{\boldsymbol{q}}(k))\boldsymbol{S}(k+1)=\boldsymbol{M}(\overline{\boldsymbol{q}}(k))\left(\boldsymbol{\Lambda q}(k+1)-\boldsymbol{C}\boldsymbol{x}_d(k+1)\right)+\sum_{i=1}^{m}\theta_i(\boldsymbol{z}(k))\left(\boldsymbol{A}_i\boldsymbol{x}(k)+\boldsymbol{b}_i\right)+\boldsymbol{u}(k)\ . \tag{4}$$

Using the approximation relation $\boldsymbol{q}(k+1)\cong\overline{\boldsymbol{q}}(k)=\boldsymbol{q}(k)+a(k)\delta\dot{\boldsymbol{q}}(k)$ [5], we obtain the tracking error dynamics as follows

$$\boldsymbol{M}(\overline{\boldsymbol{q}}(k))\tilde{\boldsymbol{S}}(k+1)=\sum_{i=1}^{m}\theta_i(\boldsymbol{z}(k))\boldsymbol{M}_i\boldsymbol{h}(k)+\sum_{i=1}^{m}\theta_i(\boldsymbol{z}(k))\left(\boldsymbol{A}_i\boldsymbol{x}(k)+\boldsymbol{b}_i\right)+\boldsymbol{u}(k)\ . \tag{5}$$

where $\tilde{\boldsymbol{S}}(k+1)=\boldsymbol{S}(k+1)-\boldsymbol{r}\boldsymbol{S}(k)$,

$$\boldsymbol{h}(k)=(a(k)\boldsymbol{\Lambda}\delta-\boldsymbol{I})\dot{\boldsymbol{q}}(k)+\bar{\boldsymbol{r}}\boldsymbol{S}(k)+\boldsymbol{C}\left(\boldsymbol{x}_d(k)-\boldsymbol{x}_d(k+1)\right). \tag{6}$$

and $\boldsymbol{r}=\boldsymbol{r}^{\mathrm{T}}>\boldsymbol{0}$ is a design parameter to assign the dynamics of $\boldsymbol{S}(k)$, and $\bar{\boldsymbol{r}}=\boldsymbol{I}-\boldsymbol{r}$.

The model constructed by sector nonlinearity approach is only a rough approximation to the robot dynamics because of parameter uncertainty and disturbances. Therefore, a stable DNF adaptive control approach will be developed. The DNF system with the same antecedents as that in (1) is constructed by

Rule i: If $z_1(k)$ is F_{i1}, $z_2(k)$ is F_{i2}, $\cdots, z_p(k)$ is F_{ip}

$$\text{Then } \hat{\boldsymbol{M}}_i(k)\dot{\hat{\boldsymbol{q}}}(k+1)=\hat{\boldsymbol{A}}_i(k)\hat{\boldsymbol{x}}(k)+\hat{\boldsymbol{b}}_i(k)+\boldsymbol{u}(k)-\bar{\boldsymbol{u}}(k),\quad i=1,\cdots,m. \tag{7}$$

where $\hat{\boldsymbol{x}}(k)=\left(\hat{\boldsymbol{q}}^{\mathrm{T}}(k),\dot{\hat{\boldsymbol{q}}}^{\mathrm{T}}(k)\right)^{\mathrm{T}}\in R^{2n}$ is the state of the DNF system. $\bar{\boldsymbol{u}}(k)$ is a robust control component, and as $\hat{\boldsymbol{x}}(k)\to\boldsymbol{x}_d(k)$, $\bar{\boldsymbol{u}}(k)\to\boldsymbol{0}$. $\hat{\boldsymbol{M}}_i(k)$, $\hat{\boldsymbol{A}}_i(k)$ and $\hat{\boldsymbol{b}}_i(k)$ $(i=1,\cdots,m)$ denote the estimates of $\boldsymbol{M}_i,\boldsymbol{A}_i$ and $\boldsymbol{b}_i$ and can be represented as below

$$\hat{\boldsymbol{M}}_i(k)=\boldsymbol{M}_{i0}+\Delta\hat{\boldsymbol{M}}_i(k),\ \hat{\boldsymbol{A}}_i(k)=\boldsymbol{A}_{i0}+\Delta\hat{\boldsymbol{A}}_i(k),\ \hat{\boldsymbol{b}}_i(k)=\boldsymbol{b}_{i0}+\Delta\hat{\boldsymbol{b}}_i(k). \tag{8}$$

where $\boldsymbol{M}_{i0}$, $\boldsymbol{A}_{i0}$, $\boldsymbol{b}_{i0}$ $(i=1,\cdots,m)$ are known consequent parameters which are determined by sector nonlinearity approach, $\Delta\hat{\boldsymbol{M}}_i(k)$, $\Delta\hat{\boldsymbol{A}}_i(k)$, $\Delta\hat{\boldsymbol{b}}_i(k)$ are estimate errors. If no priori knowledge about $\boldsymbol{M}_{i0}$, $\boldsymbol{A}_{i0}$, $\boldsymbol{b}_{i0}$ $(i=1,\cdots,m)$ are used, we have

$$\boldsymbol{M}_{i0}=\boldsymbol{0},\ \ \boldsymbol{A}_{i0}=\boldsymbol{0}\ \text{ and }\ \boldsymbol{b}_{i0}=\boldsymbol{0}. \tag{9}$$

The tracking error metric for the DNF system (7) can be defined as

$$\boldsymbol{S}_0(k)=\boldsymbol{C}(\hat{\boldsymbol{x}}(k)-\boldsymbol{x}_d(k))=\boldsymbol{\Lambda}\hat{\boldsymbol{q}}(k)+\dot{\hat{\boldsymbol{q}}}(k)-\boldsymbol{C}\boldsymbol{x}_d(k). \tag{10}$$

$\boldsymbol{S}_0(k)$ dynamics in the whole state space can be written as

$$\hat{\boldsymbol{M}}_i(k)\tilde{\boldsymbol{S}}_0(k+1)=\sum_{i=1}^{m}\theta_i(\boldsymbol{z}(k))\left(\hat{\boldsymbol{M}}_i\hat{\boldsymbol{h}}(k)+\hat{\boldsymbol{A}}_i\hat{\boldsymbol{x}}(k)+\hat{\boldsymbol{b}}_i\right)+\boldsymbol{u}(k)-\bar{\boldsymbol{u}}(k). \tag{11}$$

where $\hat{\boldsymbol{M}}(k)=\sum_{i=1}^{m}\theta_i(\boldsymbol{z}(k))\hat{\boldsymbol{M}}_i$, $\tilde{\boldsymbol{S}}_0(k+1)=\boldsymbol{S}_0(k+1)-\boldsymbol{r}_0\boldsymbol{S}_0(k)$,

$$\hat{\boldsymbol{h}}(k)=(a(k)\boldsymbol{\Lambda}\delta-\boldsymbol{I})\dot{\hat{\boldsymbol{q}}}(k)+\bar{\boldsymbol{r}}_0\boldsymbol{S}_0(k)+\boldsymbol{C}(\boldsymbol{x}_d(k)-\boldsymbol{x}_d(k+1)). \tag{12}$$

and $\boldsymbol{r}_0>\boldsymbol{0}$ will be designed to assign the dynamics of $\boldsymbol{S}_0(k)$.

For a DNF system (8), if the desired dynamics is chosen as

$$\boldsymbol{S}_0(k+1)=\boldsymbol{r}_0\boldsymbol{S}_0(k). \tag{13}$$

then the dynamic inversion of the DNF system [5] is obtained from (11) as below

$$\boldsymbol{u}_I(k)=\boldsymbol{u}(k)-\bar{\boldsymbol{u}}(k)=-\sum_{i=1}^{m}\theta_i(\boldsymbol{z}(k))\left(\hat{\boldsymbol{M}}_i\hat{\boldsymbol{h}}(k)+\hat{\boldsymbol{A}}_i\hat{\boldsymbol{x}}(k)+\hat{\boldsymbol{b}}_i\right). \tag{14}$$

The following control law is considered for the robot trajectory tracking

$$\boldsymbol{u}(k)=\boldsymbol{u}_I(k)+\bar{\boldsymbol{u}}(k). \tag{15}$$

where $\bar{\boldsymbol{u}}(k)$ includes an compensation term $\boldsymbol{u}_p(k)$ and the NF-VSC $\boldsymbol{u}_n(k)$. $\boldsymbol{u}_p(k)$ is used to compensate for the model uncertainty (see (18)) , while $\boldsymbol{u}_n(k)$ in the feed-back loop is used to enhance the stability and robustness of the robot system.

Define the state deflection metric of the robot from the DNF system as

$$\boldsymbol{S}_e(k) = \boldsymbol{C}\tilde{\boldsymbol{x}}(k) . \tag{16}$$

with $\tilde{\boldsymbol{x}}(k) = \boldsymbol{x}(k) - \hat{\boldsymbol{x}}(k) = \left(\boldsymbol{q}_e^{\mathrm{T}}(k), \dot{\boldsymbol{q}}_e^{\mathrm{T}}(k)\right)^{\mathrm{T}}$ and $\mathbf{C}$ defined as before. Then, we have

$$\begin{aligned}\boldsymbol{M}(\bar{\boldsymbol{q}}(k))\tilde{\boldsymbol{S}}_e(k+1) = \sum_{i=1}^{m}\theta_i(\boldsymbol{z}(k))\Big((\Delta\boldsymbol{M}_i - \Delta\hat{\boldsymbol{M}}_i(k))\bar{\boldsymbol{h}}(k) + (\Delta\boldsymbol{A}_i - \Delta\hat{\boldsymbol{A}}_i(k))\boldsymbol{x}(k) + \\ \Delta\boldsymbol{b}_i - \Delta\hat{\boldsymbol{b}}_i(k)\Big) + \sum_{i=1}^{m}\theta_i(\boldsymbol{z}(k))\Big(\boldsymbol{M}_i\big((a(k)\boldsymbol{A}\delta - \boldsymbol{I})\dot{\boldsymbol{q}}_e(k) + \bar{\boldsymbol{r}}\boldsymbol{C}\tilde{\boldsymbol{x}}(k)\big) + \hat{\boldsymbol{A}}_i\tilde{\boldsymbol{x}}(k)\Big) .\end{aligned} \tag{17}$$

where $\tilde{\boldsymbol{S}}_e(k+1) = \boldsymbol{S}_e(k+1) - \boldsymbol{r}\boldsymbol{S}_{\mathrm{e}}(k)$ $\bar{\boldsymbol{h}}(k) = \boldsymbol{h}(k) + (\boldsymbol{r} - \boldsymbol{r}_0)\boldsymbol{S}_0(k)$ and $\bar{\boldsymbol{u}}(k) = \boldsymbol{u}_p(k) + \boldsymbol{u}_n(k)$. The compensation term is assumed to be

$$\boldsymbol{u}_p(k) = -\sum_{i=1}^{m}\theta_i(\boldsymbol{z}(k))\Big(\hat{\boldsymbol{M}}_i(k)\big((a(k)\boldsymbol{A}\delta - \boldsymbol{I})\dot{\boldsymbol{q}}_e(k) + \bar{\boldsymbol{r}}\boldsymbol{C}\tilde{\boldsymbol{x}}(k)\big) + \hat{\boldsymbol{A}}_i(k)\tilde{\boldsymbol{x}}(k)\Big) . \tag{18}$$

Substituting (18) into (17) leads to

$$\boldsymbol{M}(k)\tilde{\boldsymbol{S}}_e(k+1) = \tilde{\boldsymbol{W}}(k)\boldsymbol{Y}(k) + \boldsymbol{u}_n(k) . \tag{19}$$

where $\tilde{\boldsymbol{W}}(k) = \boldsymbol{W} - \hat{\boldsymbol{W}}(k)$, $\boldsymbol{W} = \left(\Delta\boldsymbol{M}_1, \cdots, \Delta\boldsymbol{M}_m, \Delta\boldsymbol{A}_1, \cdots, \Delta\boldsymbol{A}_m, \Delta\boldsymbol{b}_1, \cdots, \Delta\boldsymbol{b}_m\right) \in R^{n\times n_r}$

$$\begin{aligned}&\hat{\boldsymbol{W}}(k) = \left(\Delta\hat{\boldsymbol{M}}_1(k), \cdots, \Delta\hat{\boldsymbol{M}}_m(k), \Delta\hat{\boldsymbol{A}}_1(k), \cdots, \Delta\hat{\boldsymbol{A}}_m(k), \Delta\hat{\boldsymbol{b}}_1, \cdots, \Delta\hat{\boldsymbol{b}}_m(k)\right) \in R^{n\times n_r} \\ &\boldsymbol{Y}(k) = \left(\boldsymbol{\theta}(\boldsymbol{z}(k)) \otimes \bar{\boldsymbol{h}}^{\mathrm{T}}(k), \boldsymbol{\theta}(\boldsymbol{z}(k)) \otimes \boldsymbol{x}^{\mathrm{T}}(k), \boldsymbol{\theta}(\boldsymbol{z}(k))\right)^{\mathrm{T}} \\ &\quad = \left(y_1(k), \cdots, y_{n_r}(k)\right)^{\mathrm{T}} \in R^{n_r} ,\ n_r = (3m+1)n .\end{aligned} \tag{20}$$

Finally, the nonlinear control component is defined as

$$\boldsymbol{u}_n(k) = -p_a(k)\bar{\boldsymbol{G}}sat(\boldsymbol{S}_{eg}(k)) . \tag{21}$$

where $\mathrm{S}_e(k) = \left(s_{eg}^1(k), \cdots, s_{eg}^n(k)\right)^{\mathrm{T}}$,$s_{eg}^l(k) = s_e^l(k)/A_l(k)$ with $s_e^l(k)$ being the i-th component of $\boldsymbol{S}_e(k)$ and $A_l(k)$ being the size of a sector, $sat(\mathrm{S}_{eg}(k)) = \big(sat(s_{eg}^1(k)), \cdots, sat(s_{eg}^n(k))\big)$ $\bar{\boldsymbol{G}} = \mathrm{diag}(\bar{g}_1, \cdots, \bar{g}_n) > \boldsymbol{0}$ is a constant matrix and $sat(.)$ is defined by

$$\begin{aligned}&|y| \le 1 \Rightarrow sat(y) = y;\ \ |y| > 1 \Rightarrow sat(y) = \mathrm{sgn}(y) \\ &p_a(k) = \left(\sum_{j=1}^{n_r}\left|y_j(k)\right|\right),\ \ A_l(k) = \alpha_M(\bar{p} + \bar{g}_l)^2 p_a(k)/2r_l\bar{g}_l\ ,\ \bar{p} = \max_{i,j,k}\left|\theta_{ij}(k)\right| .\end{aligned} \tag{22}$$

Theorem: If the system (1) is digitally-controlled by the control law (15), and the DNF system has the following adaptive learning algorithm:

$$\Delta\hat{\boldsymbol{W}}(k) = \boldsymbol{\eta r}\left(\boldsymbol{S}_e(k)\boldsymbol{Y}^{\mathrm{T}}(k) + \sigma(\boldsymbol{W}_0 - \hat{\boldsymbol{W}}(k))\right), \tag{23a}$$

$$\text{and } \left\|\Delta\boldsymbol{M}(k)\right\| \le (1 - r_M^2)/\alpha_M + r_m^2\eta_m\left\|\boldsymbol{Y}(k)\right\|. \tag{23b}$$

then, the robot state deflection metric will enter the sector defined as below

$$\Omega = \bigcup_{i=1}^{n}\left(s_{ei}(k)\middle|\left|s_e^l(k)\right| \le A_l(k)\right). \tag{24}$$

where $\boldsymbol{\eta} = \mathrm{diag}(\eta_1,\cdots,\eta_n) > \boldsymbol{0}$ is learning rate matrix, $\sigma > 0$ and $\boldsymbol{W}_0 = (W_{01},\cdots,$ $W_{0n_r})$ are design parameters. r_m and η_m are defined as the minimum eigenvalues of $\boldsymbol{r}$ and $\boldsymbol{\eta}$, respectively, and r_M the maximum eigenvalue of $\boldsymbol{r}$.

Proof: The Lyapunov function is chosen as

$$V(k) = \tfrac{1}{2}\boldsymbol{S}_e^{\mathrm{T}}(k)\boldsymbol{M}(k-1)\boldsymbol{S}_e(k) + \tfrac{1}{2}\boldsymbol{S}_0^{\mathrm{T}}(k)\boldsymbol{S}_0(k) + \tfrac{1}{2}tr\left(\tilde{\boldsymbol{W}}^{\mathrm{T}}(k-1)\boldsymbol{\eta}^{-1}\tilde{\boldsymbol{W}}(k-1)\right)$$

the other can follow the inference given in [5].

3 Application Example

In this section, the above control approach is employed in the control of a 2-link manipulator. Desired trajectories and initial conditions are the same as those in [5].

The discrete-time fuzzy dynamics model of a 2-link rigid robot is derived using sector nonlinearity approach [1], and nonlinear terms are chosen as

$$\begin{aligned} &z_1 = m_2 r_1' r_2' \cos(\phi),\ z_2 = m_2 r_1' r_2' \sin(\phi)\dot{\phi},\ z_3 = m_2 r_1' r_2' \sin(\phi)\dot{\theta} \\ &z_4 = -m_2 r_2' \cos(\theta + \phi),\quad z_5 = -(m_1 + m_2) r_1' \cos(\theta). \end{aligned} \tag{25}$$

If the maximum and minimum values of z_i are denoted as $z_i^{\max}$ and $z_i^{\min}$, then we have

$$z_i = E_i^1 z_i^{\max} + E_i^2 z_i^{\min},\quad i = 1,\cdots,5. \tag{26}$$

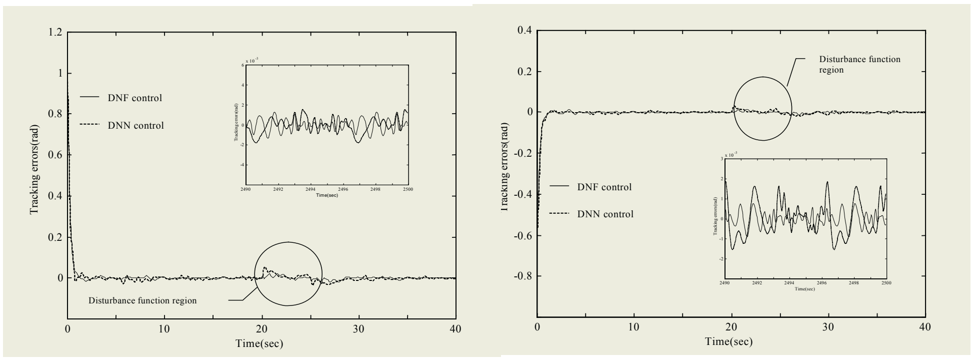
where membership functions are calculated as

$$\mu_{i1} = (z_i - z_i^{\min})/(z_i^{\max} - z_i^{\min}),\ \mu_{i2} = (z_i^{\max} - z_i)/(z_i^{\max} - z_i^{\min})(i = 1,\cdots,5). \tag{27}$$

Substituting (26) into the robot discrete model in [5] leads the fuzzy model shown in (1). p and h are determined by (32) in [6] with $\alpha_0 = 0.999$, $p^{\min} = 0.02$, $p^{\max} = 2.5$, $h^{\min} = 2.5$, $h^{\max} = 16.5$. The design parameters are chosen as

$$\boldsymbol{C} = \begin{bmatrix}\boldsymbol{\Lambda} & \boldsymbol{I}\end{bmatrix},\ \boldsymbol{\Lambda} = \mathrm{diag}(6\quad 6.5),\ \alpha_M = 0.01. \tag{28}$$

The learning rates for DNF adaptive control are $\eta_1 = \eta_2 = 36.4$, $\sigma = 2.5\times10^{-6}$ and the learning rates for DNN adaptive control are the same as those in [5]. $\bar{r}_0 =$ $\mathrm{diag}(0.091, 0.091)$ for both approaches.



(a) Joint one (b) Joint two

Fig. 1. Angle tracking error responses for robot using DNF and DNN control

Figs.1 (a) and (b) present the angle tracking errors for two joints during the first 40 seconds, where disturbance control torques of 250.0 (N·m) for joint one and 125.0 (N·m) for joint two are added during $t \in [20, 25]$. It has been shown that the DNF controller results in a better control performance in terms of convergence of tracking errors and stable precision for robot trajectory tracking.

References

1. Tanaka, K., Wang, H. O.: Fuzzy Control Systems Design and Analysis – A Linear Matrix Inequality Approach. A wiley-Interscience Publication John Wiley & Sons, Inc, (2001)
2. Jagannathan, S.: Adaptive Fuzzy Logic Control of Feedback Linearizable Discrete-time Dynamical Systems under Persistence of Excitation. Automatica, **34** (1998) 1295-1310
3. Ngo, A.D., Doman, D.B.: Dynamic Inversion-based Adaptive/Reconfigurable Control of the X-33 on Ascent. IEEE Proceedings of Aerospace Conference, **6** (2001) 2683-2697
4. Leitner, J., Calise, A., Prasad, J.V.R.: Analysis of Adaptive Neural Networks for Helicopter Flight Control. Journal of Guidance, Control, and Dynamics, 20 (1997) 972-979
5. Sun, F.C., Li, H.X., Li, L.: Robot Discrete Adaptive Control Based on Dynamic Inversion Using Dynamical Neural Networks. Automatica, **38** (2002) 1977-1983
6. Sun, F.C., Sun, Z.Q., Woo, P.Y.: Stable Neural Network-based Adaptive Control for Sampled-data Nonlinear Systems. IEEE Trans. on Neural Networks, **9** (1998) 956-968

General Underactuated Cooperating Manipulators and Their Control by Neural Network

S. Murat Yeşiloğlu and Hakan Temeltaş

Department of Electrical Engineering,
Istanbul Technical University, 34469 Istanbul Turkey
smy@ieee.org, temeltas@elk.itu.edu.tr

Abstract. Underactuation may arise when a manipulator includes at least one passive joint which is either a free joint where torque propagating along the axis of rotation is only due to friction, or a flexible joint which has passive elements introducing stiffness and damping. In this regard, free flying platform can be considered as linked to a fixed base by six DOF free joints with no friction. Dynamics of underactuated cooperating manipulators become a bigger challenge when underactuation is encountered by the combination of free, flexible and active joints based on a free-flying platform in no-gravity setting. We call this class of systems as "general underactuated systems". This paper demonstrates the theoretical background of an *order of N* algorithm for the dynamical modeling and neural network based control of general underactuated manipulators cooperating to perform a common task on a free-flying space platform.

1 Introduction

Underactuated manipulators are those possessing fewer actuators than degrees of freedom (DOF). Manipulators with flexible joints to which passive elements such as torsional spring and rotational dashpot pair, if revolute, are attached, free-flying robots in space where there is no gravity, manipulators with one or more of the actuators failed, and those that include "free joints" by design or structure, as in the case of hyper-redundant manipulators, are some of the popular examples that fall under this research area. It is a challenging problem to come up with a methodology for computationally high performance algorithm dealing with these manipulators when two or more of the same type manipulators work together in a cooperated manner to perform a common task that is to move a rigid object. Obviously, it is more challenging when all three, namely free joints, flexible joints and free-flying in space problems, coexist in the same model which is particularly what is considered in this paper. We call that "general underactuated systems."

Modeling and control of underactuated mechanical systems has been the focus area of some of the researchers in the robot dynamics field for more than a decade. Among the work done, there are a few of them addressing the dynamic modeling of underactuated closed-kinematic-chain systems. Out of those, [1] drives dynamical modeling based on the technique given by [2]. Although the algorithm is claimed to be computationally efficient, the method they use is still *order of* N^3. Multibody dynamical algorithm with *order of N* performance has been a hot research area and is being improved by researchers, such as [3], since it was first introduced by Armstrong [4] in 1979. The roots of the algorithm presented in this paper can be found in [5] which presents the dynamics of underactuated open chain manipulator.

This paper is organized as follows: The dynamical modeling algorithm is given in section 2. How to incorporate this algorithm into neural network controller is then demonstrated in section 3. Finally, section 4 is the conclusion of the paper.

2 Dynamical Modeling

Order of N algorithm for the dynamical modeling of general underactuated cooperating manipulators will be given in this chapter.

The algorithm presented here utilizes a basis-free vectorial representation. ${}^ih_k$ is a unit vector parallel to the axis of rotation of the joint at the kth link of the ith manipulator. ${}^iH_k$ and ${}^i\theta_k$ are the spatial axis of motion and the joint angle for the aforementioned joint, respectively. ${}^i\ell_{k,k+1}$ is the link vector associated with link k. Spatial velocity and the spatial acceleration of the link are denoted as ${}^iV_k$ and ${}^i\alpha_k$ and they are defined as $\left({}^i\omega_k^T \; {}^iv_k^T\right)^T$ and $\left({}^i\dot{\omega}_k^T \; {}^i\dot{v}_k^T\right)^T$,respectively, where ^{T} is the transpose operator. ${}^im_k$ is the mass and ${}^i\mathcal{I}_k$ is the inertia tensor at point ${}^iO_k$.${}^i\ell_{k,c}$ is the vector from ${}^iO_k$ to the links center of mass. Link forces and torques acting at ${}^iO_k$ are denoted as ${}^if_k$ and ${}^i\tau_k$. Link spatial force vector ${}^iF_k$ is defined as $\left({}^i\tau_k^T \; {}^if_k^T\right)^T$. Φ is the propagation matrix.

Based on the definitions given above (for more detailed explanation, please refer to [6]) the link velocities are obtained as follows:

$$V = \phi H\dot{\theta} \tag{1}$$

Defining tip propagation operator, $\sigma_t = \left[\,{}_60 \cdots {}^i\phi_{n,n-1}\right]$ yields Jacobian by premultiplying both sides of (1)

$$\mathcal{J} \overset{\triangle}{=} \sigma_t \phi H$$

Spatial forces are calculated from tip to base. Therefore, ϕ^T, in this case, is used as the propagation operator.

$$f = \phi^T(M\alpha + b + \sigma_t^T F_t) \tag{2}$$

where F_t is the tip forces and b is the stacked up spatial bias forces. The definition of spatial bias forces acting on link k is given as $b_k = \left[\,(\widehat{\omega}_k \mathcal{I}_k \omega_k)^T \; (m_k \widehat{\omega}_k^2 \ell_{k,c})^T\,\right]^T$. We obtain the applied torques if we project (2) onto the axis of motion. $\mathcal{T} = H^T f$. Finally, we get the equation of motion as

$$\mathcal{T} = \mathcal{M}\ddot{\theta} + C + \mathcal{J}^T F_t \tag{3}$$

where $\mathcal{M} \overset{\triangle}{=} H^T\phi^T M\phi H$ and $C \overset{\triangle}{=} H^T\phi^T M\phi a + H^T\phi^T b$. A linear operator S is constructed by reordering the rows of an identity matrix to rearrange the joint space into four subspaces; base, actuated joints, free joints and flexible joints. When we apply this operator to (3)

$$S\mathcal{M}S^{-1}S\ddot{\theta} + SC + S\mathcal{J}^T F_t = S\mathcal{T} \tag{4}$$

$$\begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{bmatrix} + \begin{bmatrix} C_1 \\ C_2 \end{bmatrix} + \begin{bmatrix} \mathcal{J}_1^T \\ \mathcal{J}_2^T \end{bmatrix} F_t = \begin{bmatrix} \mathcal{T}_1 \\ \mathcal{T}_2 \end{bmatrix} \tag{5}$$

Let θ_1 include base, actuated and free joint angles and θ_2 represent flexible joint angles. Then, $\mathcal{T}_2$ is the torques (or forces) generated by the joint flexibility.

$$\mathcal{T}_2 = -d\dot{\theta}_2 - k\theta_2 \tag{6}$$

where d and k are diagonal matrices representing spring and damper characteristics, respectively. Substituting (6) in (5) we get,

$$m_1\ddot{\theta}_1 + m_2\ddot{\theta}_2 + C_1 + \mathcal{J}_1^T F_t = \mathcal{T}_1 \tag{7}$$

$$m_3\ddot{\theta}_1 + m_4\ddot{\theta}_2 + d\dot{\theta}_2 + k\theta_2 + C_2 + \mathcal{J}_2^T F_t = 0 \tag{8}$$

$\ddot{\theta}_1$ can be solved from (7).

$$\ddot{\theta}_1 = m_1^{-1}(\mathcal{T}_1 - m_2\ddot{\theta}_2 - C_1 - \mathcal{J}_1^T F_t) \tag{9}$$

When (9) is substituted in (8), we get

$$\bar{\mathcal{M}}\ddot{\theta}_2 + d\dot{\theta}_2 + k\theta_2 + \bar{C} + \bar{\mathcal{J}}^T F_t = B\mathcal{T}_1 \tag{10}$$

where $\bar{\mathcal{M}} \triangleq m_4 - m_3 m_1^{-1} m_2$, $\bar{C} \triangleq C_2 - m_3 m_1^{-1} C_1$, $\bar{\mathcal{J}} \triangleq \mathcal{J}_2 - \mathcal{J}_1 {m_1^{-1}}^T m_3^T$, $B \triangleq -m_3 m_1^{-1}$. The system can be reduced from second order to first order differential equation as follows.

$$\mathcal{M}_s\dot{W} + D_s W + C_s + \mathcal{J}_s^T F_t = B_s\mathcal{T}_1 \tag{11}$$

where

$$W = \begin{bmatrix} \theta_2 \\ \dot{\theta}_2 \end{bmatrix} \quad \mathcal{M}_s = \begin{bmatrix} I & 0 \\ 0 & \bar{\mathcal{M}} \end{bmatrix} \quad D_s = \begin{bmatrix} 0 & -I \\ k & d \end{bmatrix}$$

$$C_s = \begin{bmatrix} 0 \\ \bar{C} \end{bmatrix} \quad \mathcal{J}_s = \begin{bmatrix} 0 \\ \bar{\mathcal{J}} \end{bmatrix} \quad B_s = \begin{bmatrix} 0 \\ B \end{bmatrix}$$

Defining

$$\mathcal{T}_s \triangleq B_s\mathcal{T}_1 - C_s \tag{12}$$

equation (11) becomes

$$\mathcal{M}_s\dot{W} + D_s W + \mathcal{J}_s^T F_t = \mathcal{T}_s \tag{13}$$

2.1 Dynamics of General Underactuated Cooperating Manipulators

Joint accelerations can be written as the sum of so called *free accelerations* and *correction accelerations*.

$$\ddot{\theta}(\bar{\mathcal{T}}, F_s) = \underbrace{\ddot{\theta}(\bar{\mathcal{T}}, 0)}_{\ddot{\theta}^f} + \underbrace{\ddot{\theta}(0, F_s)}_{\ddot{\theta}^\delta} \tag{14}$$

Free accelerations, $\ddot{\theta}^f$, are nothing more than joint accelerations when the closed loops are cut. For such case, equation (13) becomes

$$\mathcal{M}_s \dot{W}^f + D_s W^f = \mathcal{T}_s \tag{15}$$

This approximately yields the solution as

$$\begin{aligned} \dot{W}^f &= \mathcal{M}_s^{-1} D_s e^{-\mathcal{M}_s^{-1} D_s (t-t_o)} D_s^{-1} \mathcal{M}_s \dot{W}_o \\ W^f &= -e^{-\mathcal{M}_s^{-1} D_s (t-t_o)} D_s^{-1} \mathcal{M}_s \dot{W}_o + D_s^{-1} \mathcal{T}_s \end{aligned} \tag{16}$$

where

$$D_s^{-1} = \begin{bmatrix} k^{-1}d & k^{-1} \\ -I & 0 \end{bmatrix}$$

$$\ddot{\theta}_2^f = \begin{bmatrix} 0 & I \end{bmatrix} \dot{W}^f \tag{17}$$

$$\ddot{\theta}_1^f = m_1^{-1}(\mathcal{T}_1 - m_2 \ddot{\theta}_2^f - C_1) \tag{18}$$

$$\ddot{\theta}^f = S^{-1} \begin{bmatrix} \ddot{\theta}_1^f \\ \ddot{\theta}_2^f \end{bmatrix} \tag{19}$$

3 Neural Network Controller

A three-layer feedforward Neural Network (NN) with a sigmoid function and bias units is considered. Equation (20) defines the input output relationship of the NN

$$h(x) = U_2^T \sigma(U_1^T x) \tag{20}$$

where x is the input vector, $\sigma(\cdot)$ is the sigmoid function, U_1 is the weight matrix between input and hidden layers, and U_2 is the weight matrix between hidden and output layers.

The NN will be used as the approximation of the inverse dynamics model of the system which is cooperating manipulators. We need to start with the kinematic constraints of the system to determine its dependent joint variables.

3.1 Dependent Joint Variables

As introduced in Equation (**??**), A_c is the constraint matrix in the form of a link Jacobian propagating the velocity of the center of mass of the common load to the tip velocities of the cooperating manipulators . Therefore we can write

$$A_c V_c = \mathcal{J}\dot{\theta} \qquad \Rightarrow \qquad \widetilde{A_c}\mathcal{J}\dot{\theta} = 0 \tag{21}$$

where $\widetilde{A_c}$ is the annihilator of A_c. Equation (21) implies that $\dot{\theta}$ has to lie in the null space of $\widetilde{A_c}\mathcal{J}$. Let S_c be a linear operator rearranging the basis of the joint space so that dependent joints in terms of kinematics can be separated from the independent ones.

$$\widetilde{A_c}\mathcal{J}S_c^{-1} = \begin{bmatrix} E^{ind} & E^{dep} \end{bmatrix} \qquad S_c\dot{\theta} = \begin{bmatrix} \dot{\theta}^{ind} \\ \dot{\theta}^{dep} \end{bmatrix} \tag{22}$$

The correct choice of S_c can be checked that E^{dep} should be full rank. Consequently, the dependent joint rates can be written in terms of the independent ones.

$$\dot{\theta} = L\,\dot{\theta}^{ind} \quad \text{where} \quad L = S_c^{-1}\begin{bmatrix} I \\ -{E^{dep}}^{-1}E^{ind} \end{bmatrix} \tag{23}$$

3.2 NN in Control

The methodology provided here is based on [7] in general. Let the independent joint position tracking error e, filtered position tracking error s, and the reference trajectory θ_r be defined as

$$\begin{aligned} e &\triangleq \theta^{ind} - \theta_d^{ind} \\ s &\triangleq \dot{e} + \Lambda e \\ \theta_r &\triangleq \theta_d^{ind} - \Lambda e \end{aligned}$$

where θ_d is the desired trajectory in the joint space and $\Lambda > 0$. Now we will define the input vector for the NN.

$$x = \left[e^T \; \dot{e}^T \; {\theta_d^{ind}}^T \; {\dot{\theta}_d^{ind}}^T \; {\ddot{\theta}_d^{ind}}^T \right]^T$$

The output function of the NN is written as

$$h(x) = \mathcal{M}L\ddot{\theta}_r + \mathcal{C}_1\dot{\theta}_r \tag{24}$$

where $\mathcal{C}_1 = \mathcal{C}L + \mathcal{M}\dot{L}$.

Weight update strategy and discussion about robustness can be found in [7]. Here it will be provided just for the convenience.

$$\dot{\hat{U}}_1 = -K_1 x \hat{U}_2^T \hat{\sigma}'(Ls)^T - k_v K_1 ||Ls|| \hat{U}_1 \tag{25}$$

$$\dot{\hat{U}}_2 = -K_2(\hat{\sigma} - \hat{\sigma}'\hat{U}_1^T x)(Ls)^T - k_v K_2 ||Ls|| \hat{U}_2 \tag{26}$$

where $\hat{\sigma} = \sigma(\hat{U}_1^T x)$, K_1 and K_2 are positive definite and symmetric matrices. The initial values of the weights are chosen as zero. The control signal u is

$$u = h(x) - k_m Ls + \nu \tag{27}$$

where k_m is a positive definite constant and ν is the term that guarantees robustness.

$$\nu(t) = -k_z(||Z||_F + Z_M)s \tag{28}$$

where $Z = diag(U_1, U_2)$.

4 Conclusion

Neural network based control with a computationally high performance algorithm for the dynamical modeling of general underactuated systems has been studied. A robust neural network is used as the controller. The main contribution of this paper is to demonstrate how the dynamical algorithm can be incorporated with this controller. Because the tracking error is always guaranteed to stay bounded, the closed loop system remains stable through out on-line learning phase with weights starting from zero initial condition. Due to page limitations, we limited our focus to demonstrate just the theoretical part of this study.

References

1. Menini, L., Tornambè, A., Zaccarian, L.: Modelling and Control of an Under-actuated Closed Kinematic Chain. IEE Proceedings on Control Theory and Applications, **145** (1998) 1–8
2. Nakamura, Y., Ghodoussi, M.: Dynamics Computation of Closed-link Robot Mechanisms with Nonredundant and Redundant Actuators. IEEE Trans. on Robotics and Automation, **5** (1989) 294–302
3. Anderson, K.S., Critchley, J.H.: Improved Order-n Performance Algorithm for the Simulation of Constrained Multi-rigid-body Systems. Multibody Systems Dynamics, **9** (2003) 185–212
4. Armstrong, W.W.: Recursive Solutions to the Equations of Motion of n n-Link Manipulator. In Proceedings of the Fifth World Congress on the Theory of Machines and Mechanisms. Volume 2., Montreal (1979) 1342–1346
5. Jain, A., Rodriguez, G.: An Analysis of the Kinematics and Dynamics of Under-actuated Manipulators. IEEE Trans. on Robotics and Automation, **9** (1993) 411–422
6. Yesiloglu, S.M., Temeltas, H.: High Performance Algorithm on the Dynamics of General Underactuated Cooperating Manipulators. In Proc. of the IEEE International Conference on Mechatronics & Robotics, Aachen, Germany (2004) 531–536
7. Kwan, C.M., Yesildirek, A., Lewis, F.L.: Robust Force/Motion Control of Constrained Robots Using Neural Network. Journal of Robotic Systems, **16** (1999) 697–714

Intelligent Fuzzy Q-Learning Control of Humanoid Robots

Meng Joo Er and Yi Zhou

Intelligent Systems Center
50 Nanyang Drive, 7th Storey, Research TechnoPlaza
Boarder X Block, Singapore, 637533
emjer@ntu.edu.sg

Abstract. In this paper, a design methodology for enhancing the stability of humanoid robots is presented. Fuzzy Q-Learning (FQL) is applied to improve the Zero Moment Point (ZMP) performance by intelligent control of the trunk of a humanoid robot. With the fuzzy evaluation signal and the neural networks of FQL, biped robots are dynamically balanced in situations of uneven terrains. At the mean time, expert knowledge can be embedded to reduce the training time. Simulation studies show that the FQL controller is able to improve the stability as the actual ZMP trajectories become close to the ideal case.

1 Introduction

From the literature, the concept of Zero Moment Point (ZMP) has been actively used to ensure dynamic stability of a biped robot [1],[2]. The ZMP is defined as the point on the ground about which the sum of all the moments of the active forces is equal to zero. If the ZMP is within the convex hull of all contact points between the feet and ground, the biped robot is possible to walk. Hereafter, this convex hull of all contact points is called the stable region. As off-line predefined trajectories are not suitable for uncertain environment and uneven terrains, dynamic online control is requested. For humanoid robots, design of appropriate control laws in unstructured environments with uncertainties is very important. A number of researchers have been paying attention to intelligent methodologies, such as Artificial Neural Networks [3], Fuzzy Systems [4] and other AI algorithms, e.g. GA [5]. In [5], the author utilized fuzzy reinforcement learning based on Generalized Approximate Reasoning for Intelligent Control (GARIC) and achieved good dynamic balance control of a biped robot. Compared with the Actor-Critic method applied in [5], Q-Learning of [6] is a type of off-policy reinforcement learning and it is a method that learns action-value functions and determines a policy exclusively from the estimated values. Q-learning is one of the most important breakthroughs in Reinforcement Learning (RL) and it dramatically simplifies the analysis of the algorithm and enables early convergence proofs. In this paper, Fuzzy Q-learning (FQL) has been proposed to improve humanoid robot's stability and robustness of walking through uncertain terrains by intelligent trunk control.

2 Gait Synthesis for a Humanoid Robot

There is no unique solution for the biped dynamic balance control problem. Any trajectory through the state space (with constraints) that does not result in a failure is acceptable. To synthesize the biped walking motion, it is required to take a workspace variable p based on the ZMP criterion and the motion trajectory for $p(t)$ can be obtained by solving a dynamic optimization problem. Therefore, we can view the biped motion control problem as minimizing the following performance index

$$\int_{t_i}^{t_f} \|P_{zmp}(t) - P_{zmp}^d\|^2 dt \tag{1}$$

subject to the boundary conditions of both $p(t)$ and $\dot{p}(t)$ at initial time t_i and final time t_f , where P_{zmp} is the actual ZMP and P_{zmp}^d is the desired ZMP position. One control objective pertaining to gait synthesis for the biped dynamic balance can be described as follows:

$$P_{zmp} = (x_{zmp}, y_{zmp}, 0) \in S \tag{2}$$

where $(x_{zmp}, y_{zmp}, 0)$is the coordinate of ZMP with respect to $O - XYZ$ and S is the domain of the supporting area. In order to make the biped dynamic balancing problem a tractable one, dynamic balancing in the sagittal and frontal planes is considered to be independent. Instead of using scalar critical signal $r(t)$, a fuzzy evaluation $R(t)$ is considered for this reinforcement learning [5].

3 Fuzzy Q-Learning

3.1 FIS Structure of Q-Learning

FQL is an extension of the original Q-learning proposed in [6] that tunes FIS consequents. This method is essentially based on a state evaluation function that associates a value with each state, indicating the state quality with respect to the task. The architecture of the FQL is based on extended ellipsoidal basis function (EBF) neural networks, which are functionally equivalent to TSK fuzzy systems [7]. If the output linguistic variables of a Multi-Input Multi-Output (MIMO) rule are independent, a MIMO FIS can be represented as a collection of Multi-Input Single-Output (MISO) FISs by decomposing the above rule into m sub-rules with G_k, $k = 1, 2, ...m$ as the single consequent of the kth sub-rule [8]. The structure of the FQL following our previous work [9] is shown in Figure 1.

Let n denotes the number of inputs. Each input variable x_i $(i = 1, 2, ..., n)$ has l membership functions. Layer one transmits values of the input linguistic variable x_i, $i = 1, 2, ...n$ to the next layer directly. At the mean time, layer two evaluates membership functions (MF) of the input variables. The MF is chosen as a Gaussian function of the following form:

$$\mu_{ij}(x_i) = exp[-\frac{(x_i - c_{ij})^2}{\sigma_{ij}^2}] \quad i = 1, 2...n, j = 1, 2..., l \tag{3}$$

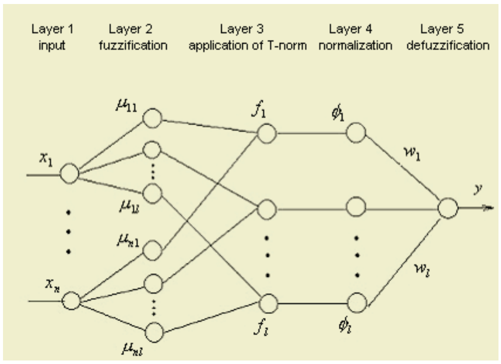


Fig. 1. Structure of fuzzy Q-learning

where μ_{ij} is the jth membership function of x_i and c_{ij} and σ_{ij} are the center and width of the jth Gaussian membership function of x_i respectively. Layer three is a rule layer. The number of nodes in this layer indicates the number of fuzzy rules. If the T-norm operator used to compute each rule's firing strength is multiplication, the output of the jth rule $R_j(j = 1, 2, ...l)$ in layer 3 is given by

$$f_j(x_1, x_2, ..., x_n) = exp[-\sum_{i=1}^{n} \frac{(x_i - c_{ij})^2}{\sigma_{ij}^2}] \quad j = 1, 2, ..., l \tag{4}$$

Normalization takes place in layer 4 and we have

$$\alpha_j = \frac{f_j}{\sum_{i=1}^{l} f_i} \quad j = 1, 2, ..., l \tag{5}$$

Lastly, nodes of layer five define output variables. If the Center-Of-Gravity (COG) method is performed for defuzzification, the output variable, as a weighted summation of the incoming signals, is given by

$$y = \sum_{j=1}^{l} \alpha_j \omega_j \tag{6}$$

where y is the value of the output variable and ω_j is the consequent parameter of the jth rule which is defined as a real-valued constant.

Due to page limitation, details of FQL will not be presented here. Instead, readers may refer to [9] for details. The neural network approximates the optimal evaluation function corresponding to the current state and FIS by using the optimal local action quality defined at time step t. Then, it computes the TD error and uses it to tune the parameter vector q based on the eligibility trace. It elects local actions based on the new vector q_{t+1} and computes the global action $U_{t+1}(X_{t+1})$ according to the new FIS. After that, it estimates the new evaluation function for the current state with the new vector q_{t+1} and the actions are effectively elected. Finally, it updates the eligibility trace, which will be used for parameter update at the next time step. Eligibility trace values need to be reset to zeros at the end of each episode.

4 Trunk Intelligent Control

As described in the section before, the neural network has 5 layers. There are two inputs in Layer 1 and 10 units in Layer 2 (there are five antecedent labels, negative medium (NM), negative small (NS), zero (ZE), positive small (PS), and positive medium (PM) for each input). There are 25 units in Layer 3 (the number of balancing rules), 7 units in Layer 4 (there are 7 consequent labels, negative big (NB), negative medium (NM), negative small (NS), zero (ZE), positive small (PS), positive medium (PM), and positive big (PB) for the output). For Layer 5, there is only one output unit to compute the desired control action. The learning algorithms described before are used to update the values of the parameters. With initial expert knowledge embedded, the Q-value of the selected action a is initialized to a fix value k_q, while all the other values are given random values according to a uniform distribution in $[0, k_q/2]$. On the other hand, without any prior knowledge, Q-values of all actions are initialized to zero or some random values. The terms ΔP_{zmp} and $\Delta \dot{P}_{zmp}$ that refer to the distance between P_{zmp} and P^d_{zmp} and the rate of distance change correspondingly are to be applied as inputs of the controller here. For FQL, intuitive balancing knowledge based on 25 fuzzy rules is presented in Table 1.

Table 1. Fuzzy rules for biped dynamic balancing

		$\Delta \dot{P}_{zmp}$				
		NM	NS	ZE	PS	PM
ΔP_{zmp}	NM	PB	PB	PM	PS	ZE
	NS	PB	PM	PS	ZE	NS
	ZE	PM	PS	ZE	NS	NM
	PS	PS	ZE	NS	NM	NB

5 Simulation Results

In this section, the FQL-based biped dynamic balance control method is applied to the simulated humanoid robot as shown in Figure 2.

We set the walking step length, $D_s = 40$cm, $H_{ao} = 16$cm,$L_{ao} = 25$cm, and walking cycle=$2 * T_c = 1.8s$ (a walking cycle consists of two steps), and the double-support phase=$T_d = 0.15$s. The details in parameter setting are listed in Table 2.

In order to demonstrate the ability of biped walking on uneven terrains, white noise of zero mean is injected into the ground. An ideal ZMP trajectory was chosen as the reference so as to show improvement due to the FQL controller. Simulation results

Table 2. Robot's parameters

	Lhip	Lthigh	Lshank	Lan	Laf	Lab
Length(cm)	40	40	50	5	10	10

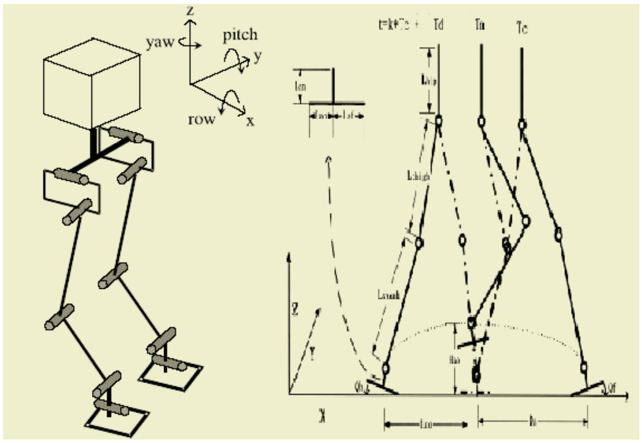


Fig. 2. The link structure of the humanoid robot

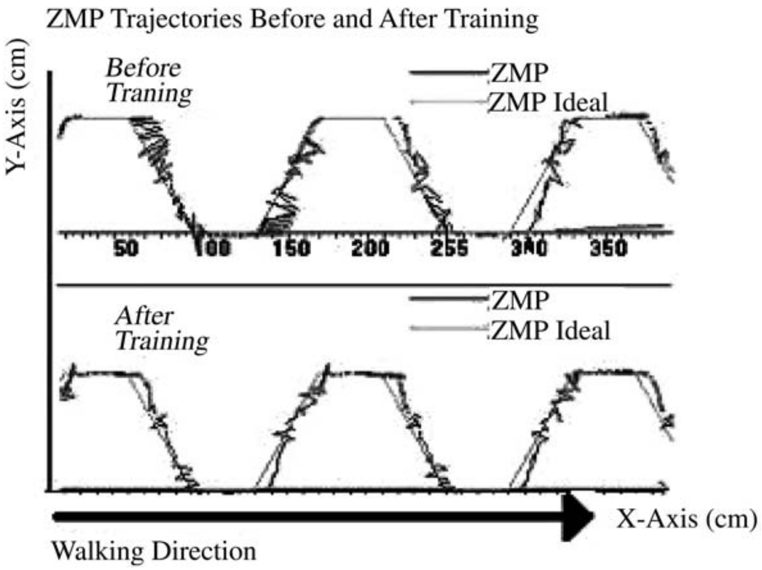


Fig. 3. Simulation results of ZMP performance before and after training

before and after reinforcement learning are shown in Figure 3. The results show that the stability of humanoid robots can be improved as the ZMP trajectories become very close to the ideal case.

6 Conclusion

Dynamic biped balance control using FQL is proposed in this paper. This FQL agent can form the initial gait from fuzzy rules obtained from expert knowledge, and then accumulate on-line reinforcement learning. Simulation results show that with the proposed FQL, much better ZMP stability can be achieved. This FQL controller can take the advantage of both human being's knowledge and training capability. Thus, the proposed intelligent control approach achieves very good performance for humanoid robots walking on uneven terrains.

Acknowledgements

We would like to thank C. Zhou and Z. Tang at the Advanced Robotics Intelligent Control Center (ARICC) of Singapore Polytechnic for their support in biped hardware and the development of simulation software.

References

1. Huang, Q., Yokoi, K., Kajita, S., Kaneko,S., Rrai, H.,Koyachi, N.,Tanie,K.: Planning Walking Patterns for A Biped Robot. IEEE Trans. Robotics and Automation, **17** (2001) 280-289
2. Vukobratovic,M.: Zero-Moment Point-Thirty Five Yeas of Its Life. International Jounal of Humanoid Robotics, **1** (2001) 157-173
3. Juang,J.G.: Fuzzy Neural Network Approaches for Robotic Gait Synthesis. IEEE Trans on Systems, Man and Cybernetics, Part B: Cybernetics, **30** (2000) 594-601
4. Ogino,M., Katoh,Y., Aono, M., Asada, M., Hosoda, K: Reinforcement Learning of Humanoid Rhythmic Walking Parameters Based on Visual Information. Advanced Robotics, **18** (2004) 677-697.
5. Zhou,C.: Robot Learning with GA-based Fuzzy Reinforcement Learning Agents. Information Science, **145** (2002) 45-68
6. Watkins C. J. C. H.: Learning from Delayed Rewards.PhD Thesis, Cambridge University (1989)
7. Jang,J. S. R.:ANFIS: Adaptive-network-based Fuzzy Inference System. IEEE Trans. System, Man and Cybernetics, **23** (1993) 665-684
8. Wang,L. X.: A Course in Fuzzy Systems and Control, New Jersey, Prentice Hall (1997)
9. Er,M. J., Deng,C.: Online Tuning of Fuzzy Inference Systems Using Dynamic Fuzzy Q-Learning. IEEE Trans on Systems, Man and Cybernetics, Part B, **34** (2004) 1478-1489

Performance Analysis of Neural Network-Based Uncalibrated Hand-Eye Coordination

Jianbo Su

Department of Automation, Shanghai Jiao Tong University
Shanghai 200030, China
jbsu@sjtu.edu.cn

Abstract. Performance of a neural network-based control scheme is investigated for uncalibrated robotic hand-eye coordination system. Since the conditions for offline modelling with neural network are normally different from those for online control, unmodeled dynamics is inevitable and should be compensated by controller design. We analyze the system's tracking error and stability with a discrete system model under a PI visual servoing controller, taking account of robot dynamics and image processing delays. The internal model principle is adopted to arrive at a feedforward compensator to enhance system performance.

1 Introduction

The Image Jacobian Matrix (IJM) model has widely been adopted in the study of uncalibrated robotic hand-eye coordination [1]. It is used to directly relate system errors observed in image plane to system control in robot coordinate system [2], irrespective of the globally nonlinear hand-eye relations that are practically very difficult to obtain. Researches show that performance of the hand-eye coordination system is mainly determined by how fast and how accurate the IJM is estimated online [3]. Among all approaches, the neural network has been shown an effective tool to implement IJM model in terms of reducing online computation expenses with offline training [4][10]. However, the conditions under which the neural network is trained offline are normally different from those that the neural network is used for online control[5][6]. There exit modelling errors, or modelling uncertainties that may not be negligible for achieving superb performance.

Researches in uncalibrated robotic visual servoing have been focusing on coordination strategy and controller design [7]. Less attention is paid for systematic analysis of performance of an uncalibrated hand-eye coordination system [8]. This paper highlights performance analysis of the uncalibrated robotic hand-eye coordination system in terms of system modelling errors from neural network realizations of IJM. System tracking errors and system stability are elaborated with the help of its root locus plot. Then, the well-known internal model principle is incorporated to design a compensatory feedforward controller. In [9], the internal model controller is used to deal with the tracking problem on a two-degree-of-freedom control system with feedback and feedforward controller. In this paper, we will show its applications in robotic visual servoing problem together with PI visual controller to deal with system's uncertain dynamics from a neural network model.

A discrete system model is adopted for analysis in Section 2, by taking account of robotic dynamics and delays from image processing. The static tracking error and the system stability are investigated in Section 3 and Section 4. A feed-forward controller is consequently suggested in Section 5 with internal model principle. Simulations are provided in Section 6 to demonstrate effectiveness of the proposed method.

2 The Discrete System Model

For the eye-in-hand configuration, motions of the camera and the robot hand are coupled, which arises great difficulty for IJM estimation. If the IJM is realized by neural network, an offline training procedure is required. Since the object motions are random and not easy to measure, training data are always accumulated under the condition that the object is static. Therefore, no matter how accurate the neural network is converged after training, there exist modelling errors when used in online control.

Fig.1 shows the structure diagram of the whole coordination system to track a 2-D moving object. System error is obtained from visual feedback by comparing the desired positions with the true positions of the robot hand in image plane, and then divided by motion planner to robotic control periods to serve as inputs to the trained neural network, by which the control signals to robot system are generated. The immeasurable object motion is treated as the external disturbance to the system control.

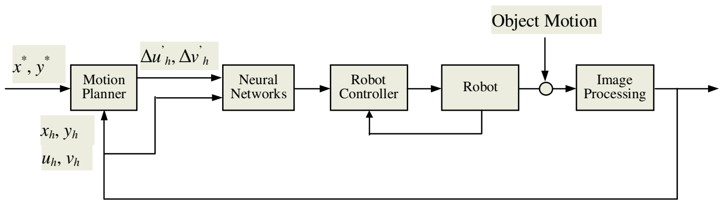


Fig. 1. Control structure of the hand-eye coordination system with neural network realization of IJM.

Since the control of the whole system is in discrete form in the sense of visual sampling and robot control, we analyze the system performance with a discrete model. Fig. 2 illustrates control diagram of the system with discrete models, in which, $D(z)$ denotes visual controller to generate velocity control to the robot from position errors of robot hand and object in image plane. $I(z) = z/(z-1)$ is an integral function to transform velocities to positions. $M(z)$ is the combined effect of the neural network, robot dynamics and camera imaging model. Without loss of generality, we only take control in the u direction of image plane as an example to analyze the system model. Then $D(z)$ can be expressed as

$$D(z) = K_p + K_i \frac{z}{z-1} = (K_p + K_i) \frac{z - K_p/(K_p + K_i)}{z-1}, \quad (1)$$

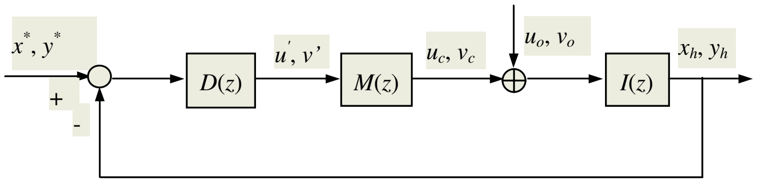


Fig. 2. System diagram of the uncalibrated robotic hand-eye system.

if a PI control law is adopted for visual controller. After a conventional system identification by the step response[5],[6], $M(z)$ can be described by a second-order system

$$M(z) = (1 + e_x)\frac{(1-a)(1-b)z^{-d+1}}{(z-a)(z-b)}, \tag{2}$$

where e_x denotes the extent of the neural network to be trained. If the neural network is sufficiently trained, e_x could be the Gaussian white noise. d is the time delay from image processing.

3 Tracking Error Analysis

It is easy to see that if the object is static, then the static error of the system is always zero, no matter whether the neural network is sufficiently trained or not.

If the object is moving, i.e., u_o is not zero, the tracking error is

$$E(z) = \frac{u_o(z)I(z)}{1 + D(z)M(z)I(z)}, \tag{3}$$

where $u_o(z)$ is the z transformation of the object motion in u direction in the image plane. If the object is moving linearly, i.e.,

$$u_o(z) = K_o\frac{z}{z-1}, \tag{4}$$

then the static tracking error is

$$e(\infty) = \lim_{z\to 1}(z-1)E(z) = 0. \tag{5}$$

If the object is moving with an acceleration, i.e.,

$$u_o(z) = K_o\frac{z}{(z-1)^2}, \tag{6}$$

then the static tracking error is

$$e(\infty) = \frac{K_o}{(1+e_x)K_i} \neq 0. \tag{7}$$

The error is closely related to the integration parameter K_i, as well as the object moving parameter K_o. The larger the K_i is, or the smaller the K_o is, the smaller the tracking error will be. This agrees with commonsense of physical observations and controller design.

4 System Stability

We analyze the system performance via root locus of the system in terms of system gain K ($K = K_p + K_i$), which is shown is Fig. 3 . It is clear in Section 3 that the system error can be reduced by increasing integration gain of the PI controller. However, if the system gain is too large, the system may have poles outside the unit circle in z plane, which means the system is unstable. Moreover, it is well known that the system may have a transient process for error convergence with large oscillations when the system poles are near unit circle in z plane. Fig. 4 shows this phenomena by the examples of tracking a linearly moving object (shown in Fig. 4(a)) and a sinusoidally moving object (shown in Fig. 4(b)).

A PID controller can also be adopted for system controller. But research shows that the system error is not decreased dramatically by adding differentiation part, yet the system stable region is enlarged. System features under other controllers are still under investigations.

5 A Compensatory Feedforward Controller

To decrease the system's static error and keep the system stable, we design a feed-forward controller by using acceleration signal. This is motivated by [9], in which the

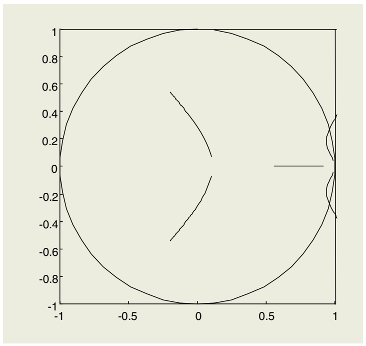
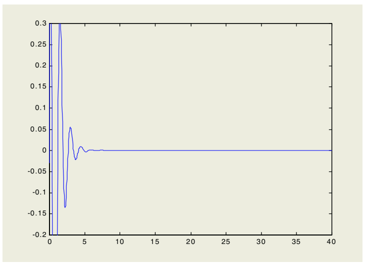
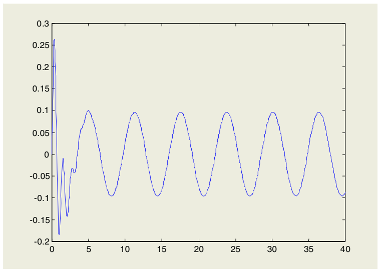


Fig. 3. Root locus under PI controller in terms of controller gain.



(a) Error for tracking a linearly moving object.



(b) Error for tracking a sinusoidally moving object

Fig. 4. Tracking errors for the system with PI controller.

internal model controller was utilized for a dynamic tracking problem in association with a feedback controller.

The controller diagram is shown in Fig. 5, in which $D_F(z)$ is the feedforward controller based on object acceleration in the image plane. Since the dynamics of the object is not known, the acceleration of the object can only be estimated online. Practically, an ARMAX model is acknowledged with a least-square iterative algorithm.

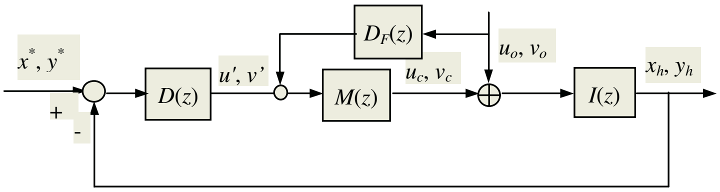


Fig. 5. System diagram with feed-forward controller.

The feedforward controller is implemented by an AR model with the following form:

$$D_F(z) = K_F \frac{1}{A(z^{-1})}, \tag{8}$$

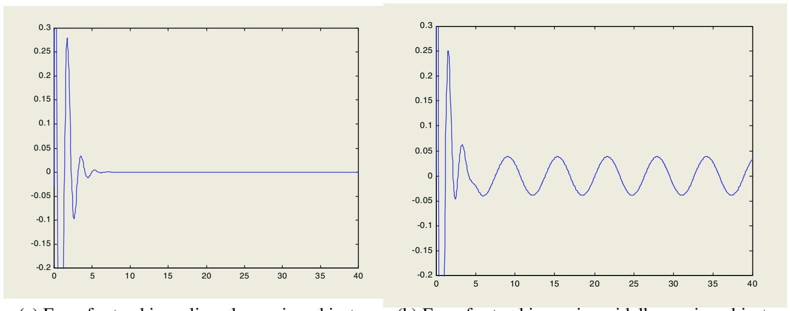
$$A(z^{-1})\Delta u_o(k-1) = [u(k) - u'(k-1)] + \xi(k), \tag{9}$$

where $A(z^{-1}) = 1+a_1z^{-1}+\cdots+a_mz^{-m}$, and $\xi(k)$ is the system noise. A least-square iterative procedure is also incorporated here to solve the above problem.

Simulation results with the feedforward controller of (8)~(9) are illustrated in Fig. 6, in which Fig.6(a) is the tracking error for a linearly moving object and Fig. 6(b) is for a sinusoidally moving object. Here, we choose $m = 6$ for the AR model. Comparing Fig. 6 with Fig. 4, it is easy to learn that the transient tracking process with the feedforward controller is smoothed with smaller overshoot at the beginning. In addition, the system tracking errors have been suppressed by half at the stable tracking stages with faster adjustment time, which exhibits the effectiveness of the proposed controller.

6 Conclusions

This paper investigates the performance of a calibration-free robotic hand-eye coordination system with neural network to realize the image Jacobian matrix model. The eye-in-hand system configuration is considered and a 2-D tracking task is instantiated to show the system control under the PI control law. The system performance is explored with the discrete model. The root locus of the system is exhibited, on which the system stability is discussed in details in terms of the gain of the PI controller. Based on the internal model principle, a feedforward controller is proposed to suppress effects of the system modelling uncertainties on system control from the neural network realization of IJM model. With the proposed feedforward controller, the system's tracking errors are reduced and the system stability is retained, which primarily demonstrates



(a) Error for tracking a linearly moving object. (b) Error for tracking a sinusoidally moving object.

Fig. 6. Tracking errors for the system with feedforward controller as compensations.

potentials of the internal model principle in the field of visual servoing control for an uncalibrated robotic hand-eye coordination system.

Future work lies in that the proposed feedforward controller should be verified in experiments. Moreover, how to utilize the internal model controller to compensate other kinds of visual servoing controller in dealing with system modelling uncertainties, and thus demonstrate extensive applications of it in visual servoing problem is absolutely a more exciting topic.

References

1. Scheering, C., Kersting, B.: Uncalibrated hand-eye coordination with a Redundant Camera System. Proc. IEEE Inter. Conf. on Robot. Automa. (1998) 2953-2958
2. Yoshimi, B.H., Allen, P.K.: Alignment Using an Uncalibrated Camera System. IEEE Trans. Robot. Automat. **11**, **4** (1995) 516–521
3. Papanikolopoulos, N.P., Khosla, P.K.: Adaptive Robotic Visual Tracking: Theory and Experiments. IEEE Trans. Robot. Automat. **38** (1993) 429-445
4. Hashimoto, H., Kubota, T., Sato, M., Harashima, F.: Visual Control of Robotic Manipulator Based on Neural Networks. IEEE Trans. on Industrial Electronics. **39** (1992) 490-496
5. Su, J., Xi, Y., Hanebeck, U., Schmidt, G.: Nonlinear Visual Mapping Model for 3-D Visual Tracking with Uncalibrated Eye-in-hand Robotic System. IEEE Trans. System, Man and Cybernetics, Part B. **34** (2004) 652-659
6. Su, J.: Dynamic Coordination of Uncalibrated Hand/eye Robotics System Based on Neural Network. Journal of System Engineering and Electronics. **12** (2001) 45-50
7. Corke, P., Good, M.: Dynamic Effects in Visual Closed-loop Systems. IEEE Trans. Robot. Automat. **12** (1996) 671-683
8. Su, J., et al.: Calibration-free Robotic Eye-hand Coordination Based on an Auto Disturbance-rejection Controller. IEEE Trans. on Robotics. **20** (2004) 899–907
9. Sugie,T., Vidyasagar, M.: Further Results on the Robust Tracking Problem in Two-degree-of-freedom Control Systems. Systems and Control Letters. **13** (1989) 101-108
10. Su, J., Pan, Q., Luo, Z.: Full-DOF Calibration-free Robotic Hand-eye Coordination Based on Fuzzy Neural Network. Lecture Notes in Computer Science. **3174** (2004) 7-12

Formation Control for a Multiple Robotic System Using Adaptive Neural Network

Yangmin Li and Xin Chen

Department of Electromechanical Engineering, Faculty of Science and Technology, University of Macau,
Av. Padre Tomas Pereira S.J., Taipa, Macao SAR, China
{ymli,ya27407}@umac.mo

Abstract. Due to the limitations of sensors, each member of a decentralized system can only deal with local information respectively. A description of local relationship within formation pattern is proposed in this paper. Furthermore, a NN control approach with robust term is proposed to control individual motion. By using such individual control method, all robots will finally form a unique formation. Based on properties of such control strategy, we propose a modified artificial potential approach to realize obstacle avoidance.

1 Introduction

Formation task achieved by multiple robots is a hot research topic in recent years. Given a predetermined relationship among robots, the issue of how to make all robots form a certain order in parade is investigated. Regarding to related work, supposed that there was a predetermined leader for coordination, a control law was provided to make a group of nonholonomic robots keep a certain formation [1]. A behavior-based approach was presented to formation maneuvers for groups of mobile robots [2]. To use a quantitative value to estimate the performance of formation, LFS(leader-to-formation stability) was proposed [3]. Usually keeping a formation can be regarded as a kind of tracking trajectory that robot should track a moving referenced point determined by other robots. We deal with a practical situation that with parameter uncertainty and bounded perturbations, how to use NN control strategy [4] to make robots form regular formation. Based on the properties of control law, we provide a modified artificial potential approach to realize obstacle avoidance.

2 Description of Formation

The key issue on formation task is how to describe the formation pattern. We propose following definitions:

Leaders-Follower Cell: If robot i follows a reference point which results from positions of a set of robots, this set is defined as leaders set of robot i, denoted by Ω_i. Then the combination of robot i and all robots in Ω_i are called as leaders-follower cells. The reference point can be expressed as $p_i^d = \sum_{j \in \Omega_i} S_{ij}(p_j + D_{ij}^d)$, where $p = [x \quad y]^T$,

D_{ij}^d represents the ideal distance between robot and each leader, $S = \{S_{ij}\}$ is a projection vector. It holds that $\sum_{j \in \Omega_i} S_{ij} = 1$.

Formation: A formation is defined as a combination of a group of leaders-follower cells. Leaders of one cell can be followers of other cells. The robot that only plays the role of leader is the leader of the formation.

Leader-to-Follower Link: In a leaders-follower cell, the relationship between a leader and the follower is defined as a leader-to-follower link whose direction is from leader to follower.

Link Between Two Robots: Between two arbitrary robots, if there exist a set of leader-to-follower links that form one and only one directional link connecting these two robots, we call it as a link between two robots. The number of leader-to-follower links contained in the link is defined as the size of the link.

Fig. 1 shows an example of formation. There are three leaders-follower cells and four leader-to-follower links. For example, robot 4 has two leaders, robot 2 and 3. Then robot 2, robot 3 and robot 4 form a leaders-follower cell.

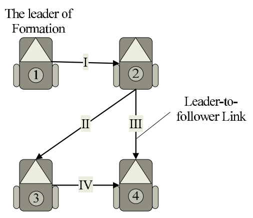


Fig. 1. Sample of formation pattern.

3 Individual Control Strategy

3.1 Dynamic Description for Individual Robot

Robot used in the formation is a kind of car-like robot with nonholonomic constraints $\dot{x}\sin\theta - \dot{y}\cos\theta + d\dot{\theta} = 0$. Since we just concern the position of robot denoted by $p = [x \quad y]^T$, the dynamics can be expressed as:

$$M_0 T\ddot{p} + M_0\dot{T}\dot{p} = S^T B\tau - \bar{\tau}_d. \tag{1}$$

where $T = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$, $M_0 = \begin{bmatrix} m & 0 \\ 0 & \frac{I}{d} \end{bmatrix}$, $S(q) = \begin{bmatrix} \cos\theta & -d\sin\theta \\ \sin\theta & d\cos\theta \\ 0 & 1 \end{bmatrix}$, $\bar{\tau}_d$ represents unmodeled structure of dynamics. d represents the distance between mass center and central point between two driving wheels.

3.2 Neural Network Controller

We adopt a modified reference point:

$$p_i^d = \lambda \sum_{j \in \Omega_i} S_{ij}(p_j + D_{ij}^d) + (1 - \lambda)p_i, \tag{2}$$

where λ is a positive scaling factor. p_i is coordinate of robot i's leader.

We define the relative position error as $e_i = p_i - p_i^d = \lambda\big[p_i - \sum_{j\in\Omega_i} S_{ij}(p_j + D_{ij}^d)\big]$. A filtered error is $z_i = \dot{e}_i + \Lambda e_i$. If define $\dot{p}_i^r = \dot{p}_i^d - \Lambda e_i$, then, $z_i = \dot{p}_i - \dot{p}_i^r$. We define a new error $\tilde{z}_i = T_i z_i$. Then equation (1) is transformed to

$$M_0 \dot{\tilde{z}}_i = S^T B \tau_i - \bar{M}_i \ddot{p}_i^r - \bar{V}_i \dot{p}_i^r - \bar{\tau}_{id}. \tag{3}$$

Because it is difficult to get $\bar{M}_i$, $\bar{V}_i$, $\bar{\tau}_{id}$ directly, we use an adaptive neural network to model $\bar{M}_i \ddot{p}_i^r + \bar{V}_i \dot{p}_i^r + \bar{\tau}_{id}$ on-line. To simplify the denotation, we omit the subscript i. We suppose that there exists a two-layer feedforward NN, $f(X_i) = \bar{M}_i \ddot{p}_i^r + \bar{V}_i \dot{p}_i^r = W^T \sigma(V^T X) + \varepsilon$, where $X_i = [\,\dot{p}_i \quad \theta_i \quad \ddot{p}_i^r \quad \dot{p}_i^r\,]^T$, $V \in R^{N_i \times N_H}$ and $W \in R^{N_H \times N_O}$ represent the input-to-hidden-layer interconnection weights and the hidden-layer-to-outputs interconnection weights, ε is the NN functional approximation error. We construct a NN function $\hat{f}(X) = \hat{W}^T \sigma(\hat{V}^T X)$ to estimate $f(X)$, where $\hat{W}$ and $\hat{V}$ are estimates of NN weights. The estimated errors are defined as $\tilde{f} = f - \hat{f}$, $\tilde{W} = W - \hat{W}$, $\tilde{V} = V - \hat{V}$, $\tilde{\sigma} = \sigma - \hat{\sigma} = \sigma(V^T X) - \sigma(\hat{V}^T X)$.

We propose the following input-output feedback linearization control law and adaptive backpropagation learning algorithm as individual control strategy:

$$\tau = (S^T B)^{-1}(\hat{W}^T \sigma(\hat{V}^T X) - K\tilde{z} + \gamma), \tag{4}$$

$$\begin{array}{l} \dot{\hat{W}} = F\hat{\sigma}'\hat{V}^T X \tilde{z}^T - F\hat{\sigma}\tilde{z}^T - \kappa F\|\tilde{z}\|\hat{W} \\ \dot{\hat{V}} = -GX(\hat{\sigma}'^T \hat{W}\tilde{z})^T - \kappa G\|\tilde{z}\|\hat{V} \end{array}, \tag{5}$$

where $K = diag\{k_1, k_2\}$ in which k_1 and k_2 are positive, γ is a robust control term expressed as $\gamma = \begin{cases} -K_r(\|\hat{Y}\|_F + Y_M)\tilde{z} - J\frac{\tilde{z}}{\|\tilde{z}\|}, & \|\tilde{z}\| \neq 0 \\ -K_r(\|\hat{Y}\|_F + Y_M)\tilde{z}, & \|\tilde{z}\| = 0 \end{cases}$ to suppress $\bar{\tau}_d$ and ε, where J and K_Y are positive. F and G are positive definite design parameter matrices governing the speed of learning. Substituting control law and $f(X)$ into equation (3), adding and subtracting $W^T\hat{\sigma}$ and $\hat{W}^T\tilde{\sigma}$ respectively, we have

$$M_0 \dot{\tilde{z}} = -K\tilde{z} - \tilde{W}^T(\hat{\sigma} - \hat{\sigma}'\hat{V}^T X) - \hat{W}^T \hat{\sigma}'\tilde{V}^T X + s + \gamma, \tag{6}$$

where $s(t) = -\tilde{W}^T \hat{\sigma}' V^T X - W^T O(\tilde{V}^T X) - \bar{\tau}_d + \varepsilon$.

A Lyapunov function is defined as

$$L = \frac{1}{2}[\tilde{z}^T M_0 \tilde{z} + tr\{\tilde{W}^T F^{-1}\tilde{W}\} + tr\{\tilde{V}^T G^{-1}\tilde{V}\}]. \tag{7}$$

Differentiating equation (7) and substituting equation (6) into its derivative, yields

$$\dot{L} = -\tilde{z}^T K\tilde{z} + \kappa\|\tilde{z}\| tr\{\tilde{W}^T(W - \tilde{W})\} - \kappa\|\tilde{z}\| tr\{\tilde{V}^T(V - \tilde{V})\} + \tilde{z}^T(s + \gamma). \tag{8}$$

We define $Y = diag\{W, V\}$, $\hat{Y} = diag\{\hat{W}, \hat{V}\}$, and $\tilde{Y} = Y - \hat{Y}$. It holds that, $tr\{\tilde{Y}(Y - \tilde{Y})\} = \langle \tilde{Y}, Y\rangle_F - \|\tilde{Y}\|_F^2 \leq \|\tilde{Y}\|_F\|Y\|_F - \|\tilde{Y}\|_F^2$, and $\|s\| \leq c_0 + c_1\|\tilde{Y}\|_F + c_2\|\tilde{Y}\|_F\|\tilde{z}\|$ [5]. If we take $K_Y > c_2$, $J \geq \frac{\kappa C_3^2}{4} + c_0$,

$$\begin{array}{l} \dot{L} \leq -K_{min}\|\tilde{z}\|^2 + \kappa\|\tilde{z}\|(\|\tilde{Y}\|_F\|Y\|_F - \|\tilde{Y}\|_F^2) - K_Y(\|\hat{Y}\|_F + Y_M)\|\tilde{z}\|^2 \\ \quad + \|\tilde{z}\|(c_0 + c_1\|\tilde{Y}\|_F + c_2\|\tilde{Y}\|_F\|\tilde{z}\|) - J\|\tilde{z}\| \\ \quad \leq -\|\tilde{z}\|\big[K_{min}\|\tilde{z}\| + \kappa\big(\|\tilde{Y}\|_F - \frac{C_3}{2}\big)^2\big] \leq 0. \end{array} \tag{9}$$

where Y_M is the bound of ideal weights, and $C_3 = Y_M + \frac{c_1}{\kappa}$. According to LaSalle's principle, the system must stabilize to the invariant set $\{\tilde{z} \mid \dot{V} = 0\}$, where $\tilde{z} = 0$. Since $z^T M_0 z = \tilde{z}^T M_0 \tilde{z}$, the invariant set can also be expressed as $\{z \mid \dot{V} = 0\}$, where $z = 0$. Furthermore, e and $\dot{e}$ converge to zero too. Since $e_i = \lambda\Big[p_i - \sum_{j\in\Omega_i} S_{ij}(p_j + D_{ij}^d)\Big]$ and $\lambda > 0$, we can conclude that in the end $p_i = \sum_{j\in\Omega_i} S_{ij}(p_j + D_{ij}^d)$. Therefore the NN control law with robust term can make robot follow reference points shown in equation (2) without too much errors.

4 Performance of Formation

Every leader j in equation (2) except for the formation's leader can be expressed as $p_j = e_j + p_j^d = e_j + \lambda\sum_{k\in\Omega_j}[S_{jk}(p_k + D_{jk}^d)] + (1-\lambda)p_j$. Substituting it into equation (2) and considering that for arbitrary robot l, $\sum_{m\in\Omega_l} S_{lm} = 1$, we can rewrite the expression of robot i's reference points as

$$p_i^d = \lambda\left(p_0 + \sum_{k\in\Xi_i} b_k D_{ik}^d\right) + \sum_{j\in\Psi_i} a_j e_j + (1-\lambda)p_i, \tag{10}$$

where Ψ_i represents the set whose elements are robots on the links between the leader of formation and robot i, p_0 is the position of the formation's leader. a_j, b_k are constant coefficient matrices. We define a new error $\tilde{p}_i = \lambda\big[p_i - \big(p_0 + \sum_{k\in\Xi_i} b_k D_k^d\big)\big]$, and the filtered error $r = \dot{\tilde{p}}_i + \Lambda\tilde{p}_i$. Then we have $z_i = r_i - \sum_{j\in\Psi_i} a_j\dot{e}_j - \Lambda\sum_{j\in\Psi_i} a_j e_j$. Substituting into (6), we have

$$M_i^0\dot{\tilde{r}}_i = -K_i\tilde{r}_i - \tilde{W}_i^T(\hat{\sigma} - \hat{\sigma}'\hat{V}_i^T X_i) - \hat{W}_i^T\hat{\sigma}'\tilde{V}_i^T X_i + s_i + \gamma_i + u_i, \tag{11}$$

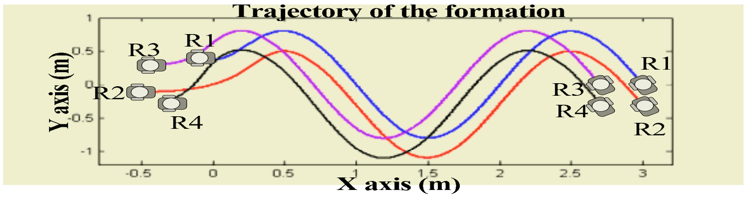
where $\tilde{r}_i = T_i r_i$, $u_i = T_i\sum_{j\in\Psi_i} a_j\ddot{e}_j + (T_i\Lambda_i + \dot{T}_i + K_i T_i)\sum_{j\in\Psi_i} a_j\dot{e}_j + (\dot{T}_i + K_i T_i)\cdot \Lambda_i\sum_{j\in\Psi_i} a_j e_j$. If $\|T_i\|$, $\|T_i\Lambda_i + \dot{T}_i + K_i T_i\|$ and $\|(\dot{T}_i + K_i T_I)\cdot\Lambda_i\|$ are bounded, there exists a positive value u_i^M so that $\|u_i(t)\| \le u_i^M$. Since $\forall j \in \Psi_i, \lim_{t\to\infty} e_j = 0$, therefore $\lim_{t\to\infty}\|u_i(t)\| = 0$. We define a positive differentiable function as

$$L = \frac{1}{2}\big[(\bar{T}R)^T\bar{M}(\bar{T}R) + tr\{\bar{\tilde{W}}^T\bar{F}^{-1}\bar{\tilde{W}}\} + tr\{\bar{\tilde{V}}^T\bar{G}^{-1}\bar{\tilde{V}}\}\big], \tag{12}$$

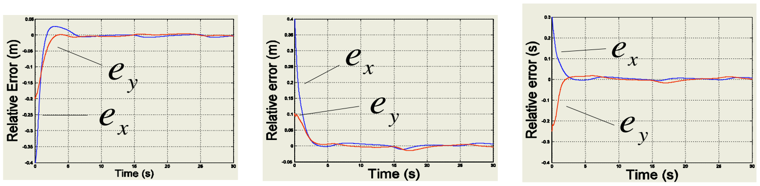
where $\bar{T} = diag\{T_i\}$, $\bar{M} = diag\{M_i^0\}$, $\bar{\tilde{W}} = diag\{\tilde{W}_i\}$, $\bar{\tilde{V}} = diag\{\tilde{V}_i\}$, $\bar{F} = diag\{F_i\}$, $\bar{G} = diag\{G_i\}$, $i = 1, \cdots, N$. Because $\bar{M}$, $\bar{F}^{-1}$, $\bar{G}^{-1}$are diagonal positive defined matrix, it must be held that, $\alpha_1(R, \bar{\tilde{W}}, \bar{\tilde{V}}) \le L \le \alpha_2(R, \bar{\tilde{W}}, \bar{\tilde{V}})$, where $\alpha_1(\cdot)$, $\alpha_2(\cdot)$ are class $\mathcal{K}$ functions. Differentiating L yields,

$$\dot{L} \le -\|\tilde{R}\|\big(\bar{K}_{min}\|\tilde{R}\| - \|U\|\big) - \sum_{i=0}^{N-1}\kappa\|\tilde{r}_i\|\big(\|\tilde{Y}_i\|_F - \frac{C_3}{2}\big)^2, \tag{13}$$

where $\tilde{R} = \bar{T}R$, $U = [0 \quad u_2 \quad \cdots \quad u_N]^T$. We can not guarantee that $\forall t > 0$, $\bar{K}_{min}\|\tilde{R}(t)\| \ge \|U(t)\|$. But since $\lim_{t\to\infty}\|U\| = \lim_{t\to\infty}\big(\sum_{i=2}^N\|u_i\|^2\big)^{0.5} = 0$, there must exist T_M so that $\forall t \geq T_M$, $\bar{K}_{min}\|\tilde{R}(t)\| \geq \|U(t)\|$. Therefore we can find a class $\mathcal{K}$ function $\alpha_3(\cdot)$ such that $\dot{L} \leq -\alpha_3(R, \tilde{\bar{W}}, \tilde{\bar{V}})$. According to the theorem of input-to-state stability [6], there exist a class $\mathcal{KL}$ function $\beta(\cdot)$ and a class $\mathcal{K}$ function $\gamma(\cdot)$, which make solution $\tilde{R}(t)$ satisfy $\|\tilde{R}(t)\| \leq \beta(\|\tilde{R}(T_M)\|, t - T_M) + \gamma(\sup_{T_M \leq \tau \leq t}\|U(t)\|)$. When $t \to \infty$, It reduces to $\|\tilde{R}(t)\| \leq \beta(\|\tilde{R}(T_M)\|, t - T_M)$. It's easy to prove that $\|R(t)\| = \|\tilde{R}(t)\|$. Therefore robot 2 to robot N will follow the reference points $p_0 + \sum_{k\in\Xi_i} b_k D_k^d$. Fig. 2 shows the result of formation simulation, whose pattern is shown in Fig. 1. We assume robot's size is 0.14×0.08, and $\|\tau_d\| \leq 0.005$. Robot 1 follows a path described as $y^d = 0.8sin(\pi x^d)$. The NN has a hidden-layer including 40 nodes.



(a)



Relative error of robot 2 Relative error of robot 3 Relative error of robot 4

(b)

Fig. 2. Simulation result.

5 Obstacle Avoidance of Formation

We use artificial potential principles to modify the coordinate of referenced point to realize obstacle avoidance. The reference point $p_i^d = \sum_{j\in\Omega_i} S_{ij}(p_j + D_{ij}^d)$ is used as attractor. The attraction potential profile can be expressed as $U_{att}(p_i) = \frac{1}{2}\lambda\rho_{att}^2(p_i)$, where $\rho_{att}(p_i)$ represents the distance between position of robot i and p_i^d. We adopt the FIRAS function [7] as repulsive potential profile which can be expressed as $U_{rep}(p_i) = \frac{1}{2}\eta\sum_{k\in H_i}\left(\frac{1}{\rho_k(p_i)} - \frac{1}{\rho_k^{eff}}\right)^2$, where H_j is the set of obstacles in the sensor scope of robot i, ρ^{eff} represents the sensor range, $\rho_k(p_i)$ is the distance between robot i and obstacle k, η is a positive scaling factor. The reference points result from gradient of artificial potential are

$$p_i^d = p_i + [-\nabla U_{att}(p_i) - \nabla U_{rep}(p_i)]$$
$$= \begin{cases} \lambda\sum_{j\in\Omega_i} S_{ij}(p_j + D_{ij}^d) + (1-\lambda)p_i + \eta\sum_{k\in H_i}\left[\left(\frac{1}{\rho_k(p_i)} - \frac{1}{\rho^{eff}}\right)\frac{1}{\rho_k^2(p_i)}\nabla\rho_k(p_i)\right], & \rho_k(p_i) \leq \rho^{eff}, \\ \lambda\sum_{j\in\Omega_i} S_{ij}(p_j + D_{ij}^d) + (1-\lambda)p_i, & \rho_k(p_i) > \rho^{eff}. \end{cases} \tag{14}$$

Obviously reference points are continuous everywhere and almost differentiable everywhere in case of $\rho_k(p_i) = \rho^{eff}$. When there is no obstacle within the sensor range of robot i, or, $\rho_k(p_i) > \rho^{eff}$, the reference point is in the form of equation (2). Since we change nothing about control strategy which is expressed in equation (4) and (5), after the robots travel through the area of obstacles, they must reconstruct the formation again.

Fig. 3 shows the simulation result for obstacle avoidance. It's clearly displayed that all robots can avoid the obstacles as shown in solid gray disks and reconstruct the formation again.

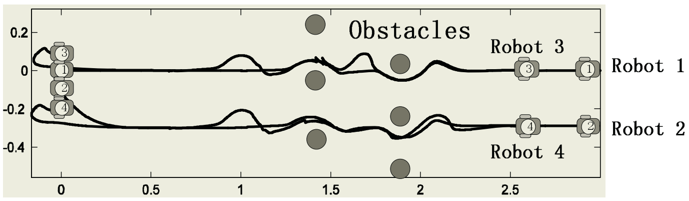


Fig. 3. Simulation on obstacle avoidance.

6 Conclusion

In this paper, it has been proved that the individual control strategy based on NN can make robots realize a formation, even if each member can only deal with local information. Since the robot can follow the leaders without errors, an artificial potential approach to get reference points is exploited, so that the system can avoid obstacles without any change of individual control law.

References

1. Desai, J.P., Ostrowski, J.P., Kumar, V.: Modeling and Control of Formations of Nonholonomic Mobile Robots. IEEE Transactions on Robotics and Automation, **17** (2001) 905–908
2. Lawton, J.R.T., Beard, R.W., Young, B.J.: A Decentralized Approach to Formation Maneuvers. IEEE Transactions on Robotics and Automation, **19** (2003) 933–941
3. Tanner, H.G., Pappas, G.J., Kumar, V.: Leader-to-Formation Stability. IEEE Transactions on Robotics and Automation, **20** (2004) 443–455
4. Fierro, R., Lewis, F.L.: Control of a Nonholonomic Mobile Robot Using Neural Networks. IEEE Transactions on Neural Networks, **9** (1998) 589–600
5. Lewis, F.L., Yegildirek, A., Kai Liu: Multilayer Neural-Net Robot Controller with Guarantee Tracking Performance. IEEE Transactions on Neural Networks, **7** (1996) 388–399
6. Khalil, H.K.: Nonlinear System. 2nd edn. Prentice Hall, Upper Saddle River, New Jersey, USA (1996)
7. Khatib, O.: Real Time Obstacle Avoidance for Manipulators and Bile Robots. International Journal of Robotics Research, **5** (1986) 90–99

Tip Tracking of a Flexible-Link Manipulator with Radial Basis Function and Fuzzy System*

Yuangang Tang, Fuchun Sun, and Zengqi Sun

State Key Lab of Intelligent Technology and Systems,
Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China
{tyg02,chunsheng,szq-dcs}@mail.tsinghua.edu.cn

Abstract. With output redefinition of flexible-link manipulators an adaptive controller for tip tracking is presented based on radial basis function network (RBFN) and fuzzy system. The uniformly asymptotical stability (UAS) of control system is guaranteed by the Lyapunov analysis and the adaptive laws including the centers and widths of Gaussian functions and coefficients of Takagi-Sugeno (TS) consequences can make the tracking error converge to zero. For comparison purpose, an RBFN controller with fixed centers and widths is also designed. The simulation results show that with similar performances the proposed controller can give smoother input torques than the conventional RBFN one.

1 Introduction

Flexible-link manipulators have been widely used in space structure. However, the unknown nonlinearities and strong coupling often caused by structural flexibility impede achieving good control performance. As universal approximators, neural networks [1] and fuzzy system [2] have been recently applied to the control of manipulator. In particular, RBFN has received considerable attention since it has a faster convergence property than a multilayer net and is mathematically tractable. Tso *et al.* [3] presented an RBFN to compensate for the highly nonlinear system uncertainties based on the inertia-related adaptive control scheme and Lee [4] employed an RBFN to approximate the nonlinear robot dynamics. Though the stability and robustness of the control system could be guaranteed, their studies don’t conclude a flexible-link manipulator. Jorge [5] used RBFN to perform adaptive control of a flexible-link manipulator, and yet it learned only weights of the RBFN with fixed centers and widths. Besides, fuzzy system is also involved to control a flexible manipulator such as [6]. Based on the functional equivalence between the RBFN and the fuzzy system [7], an RBFN with TS model consequences is considered in this paper. The proposed controller composed of an RBFN estimate and a robust term deals with tip tracking problem. The adaptive laws contain TS consequences coefficients and also both of centers and widths of Gaussian functions, which can guarantee the UAS of the control system according to Lyapunov theory. In addition, tip tracking error can go to zero even with bounded disturbance.

* This work was jointly supported by the National Excellent Doctoral Dissertation Foundation (Grant No: 200041) and the National Key Project for Basic Research of China (Grant No: G2002cb312205).

2 Preliminaries

2.1 Model and Properties of Flexible-Link Manipulators

The closed-form dynamics of a flexible-link manipulator can be written as [8] [9]

$$\begin{bmatrix} M_{\theta\theta} & M_{\theta\delta} \\ M_{\theta\delta}^T & M_{\delta\delta} \end{bmatrix}\begin{bmatrix} \ddot{\theta} \\ \ddot{\delta} \end{bmatrix}+\begin{bmatrix} C_{\theta\theta} & C_{\theta\delta} \\ C_{\delta\theta} & C_{\delta\delta} \end{bmatrix}\begin{bmatrix} \dot{\theta} \\ \dot{\delta} \end{bmatrix}+\begin{bmatrix} F_{\theta} \\ F_{\delta} \end{bmatrix}+\begin{bmatrix} G_{\theta} \\ G_{\delta} \end{bmatrix}+\begin{bmatrix} \tau_{d\theta} \\ \tau_{d\delta} \end{bmatrix}=\begin{bmatrix} \tau \\ \mathbf{0} \end{bmatrix}. \tag{1}$$

with $\theta \in R^s$ the joint variable vector, $\delta \in R^t$ the flexible modes vector, $M_{\theta\theta}$, $M_{\theta\delta}$, $M_{\delta\delta}$ the inertia matrix blocks, $C_{\theta\theta}$, $C_{\theta\delta}$, $C_{\delta\theta}$ and $C_{\delta\delta}$ the Coriolis and Centrifugal matrix blocks, F_θ and F_δ the components of the friction vector, G_θ and G_δ the components of the gravity vector, $\tau_{d\theta}$ and $\tau_{d\delta}$ the bounded un-modeled dynamics, τ the control torque vector. Two main matrix properties are used in this paper: (1) M is symmetric positive definite, (2) $\dot{M}-2C$ is skew-symmetric.

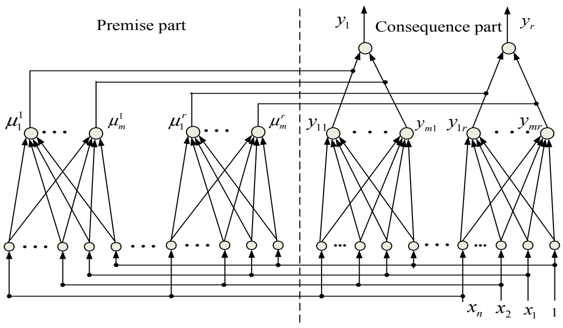


Fig. 1. Structure of the proposed RBFN

2.2 Architecture of RBFN with TS Consequences

The employed net can be seen in Fig. 1. When the net is considered as an extended version of RBFN, its weights can be regarded as linear combination of inputs (i.e. a feed-forward / parallel component is added to RBFN). In view of TS fuzzy system, it has composite premises (i.e. fuzzy rule memberships). The number of fuzzy rules is equal to the number of hidden layer's nodes of the RBFN. The fuzzy rules can be determined by expert experience, particular partition or clustering algorithm. For simplicity, a system with multi-input and single-output is assumed and the other output can be derived with the same method given in Section 3.

If Gaussian function is selected as basis function and the net output is calculated by the weighted sum method, for a certain output y_i simplified as y we have $y=x^T A\mu$, with $x=[x_0,x_1,x_2,\cdots,x_n]^T$, $x_0\equiv 1$, $A^T=[a_{ij}]$, $i=1,\cdots,m$, $j=0,\cdots,n$, $\mu=[\mu_1,\cdots,\mu_m]^T$, $\mu_i=\exp(-\|x-c_i\|^2/2\sigma_i)$, $c^T=[c_1,\cdots,c_m]^T=[c_{ij}]$, $\sigma=diag\{\sigma_1,\cdots,\sigma_m\}$. Define $\tilde{A}=A-\hat{A}$ $\tilde{\mu}=\mu-\hat{\mu}$, $\tilde{c}=c-\hat{c}$, $\tilde{\sigma}=\sigma-diag\{\hat{\sigma}_1,\cdots,\hat{\sigma}_m\}$, "^" denotes estimation value.

3 Controller Design

With simple calculation, equation (1) becomes

$$M_{\theta\theta}\ddot{\theta}+C_{\theta\theta}\dot{\theta}+M_{\theta\delta}S+C_{\theta\delta}\dot{\delta}+F_{\theta}+G_{\theta}-M_{\theta\delta}M_{\delta\delta}^{-1}\tau_{d\delta}+\tau_{d\theta}=\tau\,. \tag{2}$$

where $S=-M_{\delta\delta}^{-1}\left(M_{\theta\delta}^{T}\ddot{\theta}+C_{\delta\theta}\dot{\theta}+C_{\delta\delta}\dot{\delta}+F_{\delta}+G_{\delta}\right)$. To avoid non-minimum of flexible-link manipulators, output redefinition is applied, so the tip-positions of the links can be expressed as $y=\theta+d/l$, d is the tip deflection. $d=[d_1,\cdots,d_n]^T$, $d_g=\sum_h^{N_g}\phi_{gh}\delta_{gh}$, ϕ_{gh} is the spatial mode shape function, δ_{gh} is the hth flexible mode of link i, N_g is the flexible modes number of link g, l is the length of flexible link.

Given the desired tip trajectory y_d, tip tracking error and its derivative can be obtained $e=y_d-y=y_d-\theta-d/l$ and $\dot{e}=\dot{y}_d-\dot{\theta}-\dot{d}/l$. Considering (2) we have

$$M_{\theta\theta}\dot{e}=M_{\theta\theta}\dot{y}_d-M_{\theta\theta}\dot{\theta}-M_{\theta\theta}\dot{d}/l=-C_{\theta\theta}e-\tau+f+\overline{\tau}_d\,. \tag{3}$$

with $f=C_{\theta\theta}e+M_{\theta\theta}\dot{y}_d-M_{\theta\theta}\dot{d}/l-M_{\theta\theta}\dot{\theta}+M_{\theta\theta}\ddot{\theta}+C_{\theta\theta}\dot{\theta}+M_{\theta\delta}S+C_{\theta\delta}\dot{\delta}+F_{\theta}+G_{\theta}$, $\overline{\tau}_d=M_{\theta\delta}M_{\delta\delta}^{-1}\tau_{d\delta}+\tau_{d\theta}$.

If $x=[1,e,\dot{e},y_d,\dot{y}_d,\ddot{y}_d]^T$ and control input torque is in the form of

$$\tau=\hat{f}+R\,. \tag{4}$$

with R the robust term, the controller parameters can be determined by Theorem 1.

Theorem 1: Consider the dynamic system (1), for the bounded continuous desired tip trajectory with bounded velocity and acceleration, controller (4) can guarantee the UAS of the control system with $R=K_pe$, $K_p=diag\{k_{p1},\cdots,k_{pq}\cdots,k_{pn}\}$, $k_{pq}\geq\|\overline{\omega}\|$, $\overline{\omega}=x^T\tilde{A}(-c^T(\overline{x}-\hat{c})/\hat{\sigma}^2+\sigma^TB/\hat{\sigma}^3)\hat{\mu}+x^TA\,\mathrm{O}(\hat{c},\hat{\sigma})+\varepsilon+\overline{\tau}_d$, ε is the reconstruction error of the proposed network, $\mathrm{O}(\hat{c},\hat{\sigma})$ is Lagrange remainder of μ. And the adaptive laws are given by

$$\dot{\hat{A}}^T=\alpha\left(\left(\hat{c}^T(\overline{x}-\hat{c})-B+\hat{\sigma}^2\right)/\hat{\sigma}^2\right)\hat{\mu}\,e^Tx^T\,. \tag{5}$$

$$\dot{\hat{c}}=-\beta\frac{(\overline{x}-\hat{c})}{\hat{\sigma}^2}\hat{\mu}\,e^Tx^T\hat{A}\,. \tag{6}$$

$$\dot{\hat{\sigma}}=\gamma\frac{B}{\hat{\sigma}^3}\hat{\mu}\,e^Tx^T\hat{A}\,. \tag{7}$$

where $\overline{x}=[x^1,x^2,\cdots,x^m]$, $x^k=x$, $k=1,\cdots m$, α, β, γ are positive constants. Furthermore, tip tracking error converges to zero asymptotically.

Proof: Define the Lyapunov function

$$V=e^TM_{\theta\theta}e/2+tr\left(\tilde{A}\tilde{A}^T\right)/2\alpha+tr\left(\tilde{c}\tilde{c}^T\right)/2\beta+tr\left(\tilde{\sigma}\tilde{\sigma}^T\right)/2\gamma$$

Differentiating yields

$$\dot{V} = e^T M_{\theta\theta}\dot{e} + e^T \dot{M}_{\theta\theta} e/2 + tr\left(\tilde{A}\dot{\tilde{A}}^T\right)/\alpha + tr\left(\tilde{c}^T\dot{\tilde{c}}\right)/\beta + tr\left(\tilde{\sigma}^T\dot{\tilde{\sigma}}\right)/\gamma$$

$$= e^T\left(\dot{M}_{\theta\theta} - 2C_{\theta\theta}\right)e/2 + e^T\left(f - \hat{f} + \overline{\tau}_d - R\right) + tr\left(\tilde{A}\dot{\tilde{A}}^T\right)/\alpha + tr\left(\tilde{c}^T\dot{\tilde{c}}\right)/\beta + tr\left(\tilde{\sigma}^T\dot{\tilde{\sigma}}\right)/\gamma$$

Let $B = diag\{\|x-\hat{c}_1\|^2, \cdots, \|x-\hat{c}_m\|^2\}$ and $\mu = \hat{\mu} + \tilde{c}\cdot\partial\mu/\partial\hat{c} + \tilde{\sigma}\cdot\partial\mu/\partial\hat{\sigma} + \mathrm{O}(\hat{c},\hat{\sigma})$, we obtain

$$f - \hat{f} + \overline{\tau}_d - R = x^T A\mu + \varepsilon - x^T\hat{A}\hat{\mu} + \overline{\tau}_d - R = x^T A\tilde{\mu} + x^T\tilde{A}\hat{\mu} + \varepsilon + \overline{\tau}_d - R$$

$$= x^T\tilde{A}\tilde{\mu} + x^T\hat{A}\tilde{\mu} + x^T\tilde{A}\hat{\mu} + \varepsilon + \overline{\tau}_d - R$$

$$= x^T\tilde{A}\left(\left(\left(\hat{c}^T(\overline{x}-\hat{c}) - B\right)/\hat{\sigma}^2 - c^T(\overline{x}-\hat{c})/\hat{\sigma}^2 + \sigma^T B/\hat{\sigma}^3\right)\hat{\mu} + \mathrm{O}(\hat{c},\hat{\sigma})\right)$$

$$+ x^T\hat{A}\left(\left(-\tilde{c}^T(\overline{x}-\hat{c})/\hat{\sigma}^2 + \tilde{\sigma}^T B/\hat{\sigma}^3\right)\hat{\mu} + \mathrm{O}(\hat{c},\hat{\sigma})\right) + x^T\tilde{A}\hat{\mu} + \varepsilon + \overline{\tau}_d - R$$

$$= x^T\tilde{A}\left(\hat{c}^T(\overline{x}-\hat{c}) - B + \hat{\sigma}^2/\hat{\sigma}^2\right)\hat{\mu} - x^T\hat{A}\left(\tilde{c}^T(\overline{x}-\hat{c})/\hat{\sigma}^2\right)\hat{\mu} + x^T\hat{A}\left(\tilde{\sigma}^T B/\hat{\sigma}^3\right)\hat{\mu} + \overline{\omega} - R$$

where $\overline{\omega} = x^T\tilde{A}\left(-c^T(\overline{x}-\hat{c})/\hat{\sigma}^2 + \sigma^T B/\hat{\sigma}^3\right)\hat{\mu} + x^T A\,\mathrm{O}(\hat{c},\hat{\sigma}) + \varepsilon + \overline{\tau}_d$. Since $E^T F = tr\left(FE^T\right)$,

$$e^T\left(f - \hat{f} + \overline{\tau}_d - R\right) + tr\left(\tilde{A}\dot{\tilde{A}}^T\right)/\alpha + tr\left(\tilde{c}^T\dot{\tilde{c}}\right)/\beta + tr\left(\tilde{\sigma}^T\dot{\tilde{\sigma}}\right)/\gamma$$

$$= e^T x^T\tilde{A}\left(\left(\hat{c}^T(\overline{x}-\hat{c}) - B + \hat{\sigma}^2\right)/\hat{\sigma}^2\right)\hat{\mu} + tr\left(\tilde{A}\dot{\tilde{A}}^T\right)/\alpha - e^T x^T\hat{A}\left(\tilde{c}^T(x-\hat{c})/\hat{\sigma}^2\right)\hat{\mu}$$

$$+ tr\left(\tilde{c}^T\dot{\tilde{c}}\right)/\beta + e^T x^T\hat{A}\left(\tilde{\sigma}B/\hat{\sigma}^3\right)\hat{\mu} + tr\left(\tilde{\sigma}^T\dot{\tilde{\sigma}}\right)/\gamma + e^T(\overline{\omega} - R)$$

$$= tr\left(\tilde{A}\left(\dot{\tilde{A}}^T + \alpha\left(\left(\hat{c}^T(\overline{x}-\hat{c}) - B + \hat{\sigma}^2\right)/\hat{\sigma}^2\right)\hat{\mu}e^T x^T\right)\right)/\alpha + e^T(\overline{\omega} - R)$$

$$+ tr\left(\tilde{c}^T\left(\dot{\tilde{c}} - \beta\left((\overline{x}-\hat{c})/\hat{\sigma}^2\right)\hat{\mu}e^T x^T\hat{A}\right)\right)/\beta + tr\left(\tilde{\sigma}^T\left(\dot{\tilde{\sigma}} + \gamma\left(B/\hat{\sigma}^3\right)\hat{\mu}e^T x^T\hat{A}\right)\right)/\gamma$$

With the adaptive laws (5)-(7) and $R = K_p e$, $K_p = diag\{k_{p1}, \cdots k_{pq}, \cdots k_{pn}\}$, we have

$$e^T(\overline{\omega} - R) \le \|e^T\|\|\overline{\omega}\| - \left(k_{p1}\|e_1\| + \cdots + k_{pn}\|e_n\|\right) \le \left(\|e_1\| + \cdots + \|e_n\|\right)\|\overline{\omega}\| - \left(k_{p1}\|e_1\| + \cdots + k_{pn}\|e_n\|\right)$$

Selecting $k_{pq} \ge \|\overline{\omega}\|$, $q = 1, \cdots n$, then $\dot{V} \le 0$.

For comparison, a conventional RBFN controller in the form of (4) is designed for (1) with fixed centers and widths, which are determined by the regular lattice method [10]. The controller parameters can be determined by Theorem 2.

Theorem 2: For the bounded, continuous desired tip trajectory with bounded velocity and acceleration, controller (4) for (1) can guarantee the UAS of the control system with $R' = K'_p e$, $K'_p = diag\{k'_{p1}, \cdots k'_{pq}, \cdots k'_{pn}\}$, $k'_{pq} \ge \|\overline{\omega}'\|$, $\overline{\omega}' = \eta + \overline{\tau}_d$, η is the RBFN reconstruction error. And the weights adaptive laws of the RBFN are given by $\dot{\hat{W}}^T = \kappa\mu e^T$, with κ a positive constant. Moreover, e goes to zero asymptotically.

4 Simulation Results

In this section, simulation results for both controllers are presented. A planar two-link flexible manipulator [9] is considered, especially $m_p = 0.01kg$, $D = 0.5 * sqrt(K)$, K is stiffness matrix. Two RBFN controllers are designed: (1) conventional one with fixed c , σ ; (2) proposed one with adjustable c , σ and TS consequences coefficients. As for the control goal, the final desired tip-position of each flexible link

$$y_{1d} = \begin{cases} \sin(t) & 0 \le t < 3\pi/2 \\ \sin(3\pi/2) & t \ge 3\pi/2 \end{cases} \qquad y_{2d} = \begin{cases} \cos(t) & 0 \le t < 3\pi/2 \\ \cos(3\pi/2) & t \ge 3\pi/2 \end{cases}$$

should be reached. On set $[-1,1]\times[-1,1]$ we use 5 RBFN nodes with variance $\sigma^2 = \pi$ to spread over the regular grid. Some other parameters are specified as following: $m = 5$, initial weight values of two RBFNs are all zeros, initial values of the generalized coordinates are also zeros.

For the conventional RBFN controller $\kappa = 1000$, $K'_p = diag\{150,150\}$. For the proposed one $\alpha = \beta = \gamma = 1000$, $K_p = diag\{80,80\}$. Simulation results on tip tracking of the flexible-link manipulator can be shown in Fig.2-3. Fig.2 describes the tip tracking errors of both the flexible links and Fig.3 illustrates the control input torques of both

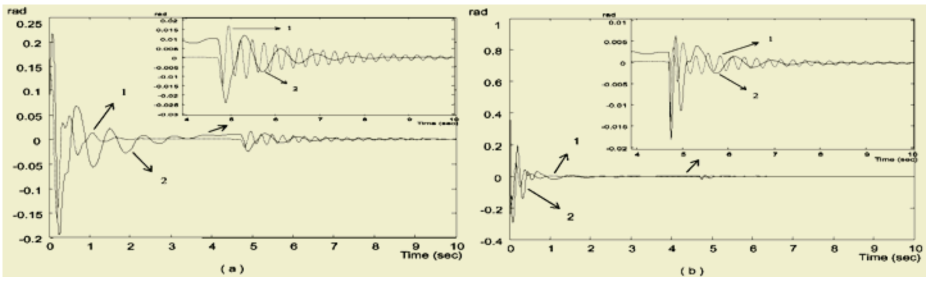


Fig. 2. Tip tracking errors of both flexible links. (a) Tip tracking error of the first link with fixed c, σ (dashed lines "1") and adjustable c, σ (solid lines "2"). (b) Tip tracking error of the second link with fixed c, σ and adjustable c, σ

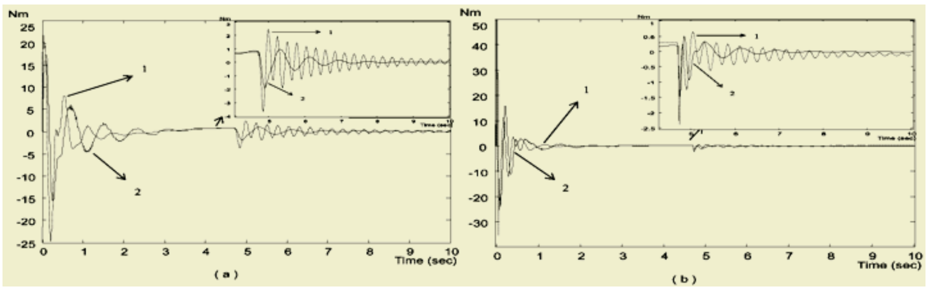


Fig. 3. Control torques of both flexible links. (a) Torque of the first link with fixed c, σ (dashed lines "1") and adjustable c, σ (solid lines "2"). (b) Torque of the second link with fixed c, σ and adjustable c, σ

links. We can see that (1) both RBFN controllers can effectively suppress tip vibration, and (2) with similar tracking performance the proposed controller can give smoother control torques than the conventional one because Theorem 1 provides a finer tuning algorithm than Theorem 2 by applying more adjustable parameters.

5 Conclusions

This paper proposes a stable adaptive controller based on RBFN and fuzzy TS consequences, which is employed to deal with the tip tracking of a flexible-link manipulator. The controller can guarantee the UAS of closed-loop system and also can effectively suppress the vibration of flexible links. In addition, smooth torques are obtained, which can avoid chattering problems. Comparative simulations show the superiority of the proposed controller to the conventional RBFN one.

References

1. Poggio, T., Girosi, F.: Networks for Approximation and Learning. In Proc. of the IEEE, **78** (1990) 1481-1497
2. Ying, H.: General Takagi-Sugeno Fuzzy Systems are Universal Approximators. In Proc. of the IEEE International Conference on Fuzzy Systems, **1** (1998) 819-823
3. Tso, S.K., Lin, N.L: Adaptive Control of Robot Manipulator with Radial-basis-function Neural Network. IEEE International Conference on Neural Networks, **3** (1996) 1807-1811
4. Lee, M.J., Choi, Y.K.: An Adaptive Neurocontroller Using RBFN for Robot Manipulators. IEEE Trans. on Industrial Electronics, **51** (2004) 711-717
5. Arciniegas, J.I., Eltimsahy, A.H., Cios, K.J.: Neural-networks-based Adaptive Control of Flexible Robotic Arms. Neurocomputing, **17** (1997) 141-157
6. Mannani, A., Talebi, H.A.: TS-model-based Fuzzy Tracking Control for a Single-link Flexible Manipulator. In Proc. of IEEE Conference on Control Applications, **1** (2003) 374-379
7. Anderson, H.C., Lotfi, A., Westphal, L.C., Jang, J.R.: Comments on Functional Equivalence Between Radial Basis Function Networks and Fuzzy Inference Systems [and reply]. IEEE Trans. on Neural Networks, **9** (1998) 1529-1532
8. De Luca, A., Siciliano, B.: Closed-form Dynamic Model of Planar Multilink Lightweight Robots. IEEE Trans. on Systems, Man and Cybernetics, **21** (1991) 826-839
9. De Luca, A., Siciliano, B.: Regulation of Flexible Arms Under Gravity. IEEE Trans. on Robotics and Automation, **9** (1993) 463-467

Obstacle Avoidance for Kinematically Redundant Manipulators Using the Deterministic Annealing Neural Network

Shubao Liu and Jun Wang

Department of Automation and Computer-Aided Engineering
The Chinese University of Hong Kong, Shatin, N.T., Hong Kong
{sbliu,jwang}@acae.cuhk.edu.hk

Abstract. With the wide deployment of redundant manipulators in complex working environments, obstacle avoidance emerges as an important issue to be addressed in robot motion planning. In this paper, a new obstacle avoidance scheme is presented for redundant manipulators. In this scheme, obstacle avoidance is mathematically formulated as a time-varying linearly constrained quadratic programming problem. To solve this problem effectively in real time, the deterministic annealing neural network is adopted, which has the property of low structural complexity. The effectiveness of this scheme and the real time solution capability of the deterministic neural network is demonstrated by using a simulation example based on the Mitsubishi PA10-7C manipulator.

1 Introduction

Kinematically redundant manipulators are those having more degrees of freedom (DOFs) than required to perform a given manipulation task. The redundant DOFs can be utilized to optimize certain performance criteria and avoid obstacles in their workspace, while performing the given motion task [1],[2],[3],[4],[5]. Being dexterous and flexible, redundant manipulators have been widely deployed in dynamic environments, where an important issue to be considered is how to effectively avoid the moving objects in the workspace of the manipulators.

Path planning, as a central task for robot control, has been widely studied under the framework of optimization [4],[5]. We further extend the optimization model to incorporate the obstacle avoidance. Recently, Zhang and Wang has done some work on this area [4], but their formulation was restrictive and reduced the feasible solution set. In this paper, the obstacle avoidance is formulated as an inequality constraint in the framework of optimization with feasible solution set as the superset of that of the formulation in [4].

Recurrent neural networks provide an efficient computing paradigm for online solution of kinematics problem. Several network models have been proposed such as penalty-parameter neural network [6], Lagrangian neural network [7], dual neural network [3], primal-dual neural network [9], and deterministic neural network [8]. Here the deterministic neural network is selected due to its low structural complexity.

2 Problem Formulation

2.1 Inverse Kinematics

Consider a redundant manipulator in which the end-effector pose (position and orientation) vector $r_E(t) \in R^m$ in Cartesian space is related to the joint-space vector $\theta(t) \in R^n$ $(n > m)$ through the following forward kinematic equation:

$$r_E(t) = f(\theta(t)). \tag{1}$$

The manipulator path planning problem (also called inverse kinematics problem or kinematic control problem) is to find the joint variable $\theta(t)$ for any given $r_E(t)$ through the inverse mapping of (1). Unfortunately, it is usually impossible to find an analytic solution due to the nonlinearity of $f(\cdot)$. The inverse kinematics problem is thus usually solved at the velocity level with the relation

$$J_E(\theta)\dot{\theta} = \dot{r}_E, \tag{2}$$

where $J_E(\theta) = \partial f(\theta)/\partial\theta \in R^{m\times n}$ is the Jacobian matrix.

For a redundant manipulator, (2) is underdetermined. That is, there are multiple solutions for this equation. Then we can select the best solution based on some performance criteria. For engineering plausibility and mathematical tractability, the norm of velocities is often chosen as the criteria. Then the following time-varying linearly constrained quadratic programming problem is used for online solution of manipulator path planning problem.

$$\text{minimize } \tfrac{1}{2}||\dot{\theta}||_2^2 \quad \text{subject to } J_E(\theta)\dot{\theta} = \dot{r}_E. \tag{3}$$

2.2 Obstacle Avoidance Scheme

For obstacle avoidance, the first step is to identify the critical point C on each link of the manipulator by calculating/measuring the distance between obstacles and the link. The second step is to assign the critical point C a desired velocity which directs the vulnerable link away from the obstacle.

Critical Point Location. The critical point C is defined as the point on the link L which is closest to the obstacle point O. As shown in Fig. 1(a) and Fig. 1(b), corresponding to different relative positions of the obstacle and the link, there are two possible cases

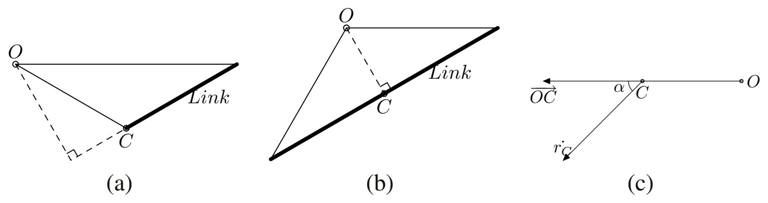


Fig. 1. Critical Point Location and Escape Direction

for locating the critical point C on the vulnerable link L. Here the obstacle point O is the representative of the obstacle object. In model-based control, the position of the obstacle is *a priori* available while in sensor-based control, it is determined by synthetic information of sensor fusion technology, e.g., utilizing vision, ultrasonic, and infrared sensors. The critical point C is thus derived via the online distance minimization between the manipulator link and the obstacle object.

Equality Constraints Formulation. We want to devise a scheme to incorporate the obstacle avoidance into the framework of optimization formulation (3). A direct thought is to turn it into a constraint. Now let's go into the details.

If the obstacle-link distance OC is less than the safety threshold d_T, the critical point Jacobian $J_C(\theta) \in R^{3\times n}$ has to be calculated. Then an escape velocity $\dot{r}_C \in R^3$ is derived and assigned to the critical point C, which will direct the manipulator link L away from the obstacle O. The next step for obstacle avoidance is to treat the collision-free criterion as the dynamic equality constraints $J_C(\theta)\dot{\theta} = \dot{r}_C$, and thus the equivalent QP formulation is

$$\text{minimize } \tfrac{1}{2}||\dot{\theta}||_2^2 \quad \text{subject to } J_E(\theta)\dot{\theta} = \dot{r}_E, \quad J_{C_i}\dot{\theta} = \dot{r}_{C_i}, \quad i = 1, \cdots, p, \tag{4}$$

where p is the number of critical points.

This method ensures that the manipulator moves away from the obstacle once it enters the danger zone. But the following problems remain to be answered:

- How to determine the suitable magnitude of escape velocity $\dot{r}_{C_i}$?
- Suppose that there are p critical points. If $m + p > n$, the optimization problem (4) is overdetermined, i.e., it has no solution.

Inequality Constraints Formulation. To avoid unnecessarily reducing the solution space, the obstacle avoidance equality constraints can be replaced by inequality constraints. As shown in Fig.1(c), we can just constrain the direction of $\dot{r}_{C_i}$ in a range and let the optimization process to determine its accurate direction and magnitude.

The shortest distance OC between the obstacle O and the link L will non-decrease, if and only if $-\pi/2 \leq \alpha \leq \pi/2$. The proof is easy. Because in the direction of OC, the velocity of C is always away from O. That is, OC is always non-descreasing.

Thus the QP formulation for obstacle avoidance evolves to this form

$$\begin{aligned} &\text{minimize } \tfrac{1}{2}||\dot{\theta}||_2^2 \\ &\text{subject to } J_E(\theta)\dot{\theta} = \dot{r}_E, \quad -\pi/2 \leq \alpha_i \leq \pi/2, \quad i = 1, \cdots, p. \end{aligned} \tag{5}$$

Because in the optimization formulation (5), the variable is $\dot{\theta}$, the obstacle avoidance inequality constraints should be turn into the expression of $\dot{\theta}$. Notice that

$$-\pi/2 \leq \alpha_i \leq \pi/2 \Longleftrightarrow \cos(\alpha_i) \geq 0 \Longleftrightarrow \overrightarrow{O_iC_i} \cdot \dot{r}_{C_i} \geq 0,$$

and

$$\overrightarrow{O_iC_i} = \left[x_{C_i} - x_{O_i} \; y_{C_i} - y_{O_i} \; z_{C_i} - z_{O_i} \right]^T,$$

we can get

$$-\pi/2 \leq \alpha_i \leq \pi/2 \Longleftrightarrow \left[x_{C_i} - x_{O_i} \; y_{C_i} - y_{O_i} \; z_{C_i} - z_{O_i} \right] J_{C_i}(\theta)\dot{\theta} \geq 0. \tag{6}$$

Denote

$$L(\theta) := \begin{pmatrix} \left[x_{C_1} - x_{O_1} \; y_{C_1} - y_{O_1} \; z_{C_1} - z_{O_1} \right] J_{C_1}(\theta) \\ \cdots \\ \left[x_{C_p} - x_{O_p} \; y_{C_p} - y_{O_p} \; z_{C_p} - z_{O_p} \right] J_{C_p}(\theta) \end{pmatrix}.$$

From (6), formulation (5) is equivalent to

$$\text{minimize } \tfrac{1}{2}||\dot{\theta}||_2^2 \quad \text{subject to } J_E(\theta)\dot{\theta} = \dot{r}_E, \quad L(\theta)\dot{\theta} \leq 0. \tag{7}$$

Now let's compare the proposed scheme with the one proposed in [4] where the obstacle avoidance is formulated as an inequality constraint

$$J_N(\theta)\dot{\theta} \leq 0, \tag{8}$$

where $J_N(\theta) = -sgn(\overrightarrow{OC}) \diamond J_C(\theta)$ [1]. The vector-matrix multiplication operator $\diamond$ is defined as $u \diamond V = [u_1V_1, u_2V_2, \cdots, u_pV_p]^T$, where column vector $u := [u_1, u_2, \cdots, u_p]^T$ and the row vector V_i denotes the ith row of matrix V.

From (8), we can see that the direction of escape velocity is confined in the quadrant where $\overrightarrow{OC}$ lies, which varies with the Cartesian coordinates setted up. While in the this proposed scheme, the direction is only required to be on the perpendicular half plane on which side $\overrightarrow{OC}$ lies. That is to say, half of feasible solution set is unnecessarily removed in the scheme in [4].

While doing trajectory planning and obstacle avoidance, joint velocity bounds must be taken into consideration. Otherwise, the optimization solution may be unreachable in real world. Joint velocity bounds can be written as inequality constraints $\eta^- \leq \dot{\theta} \leq \eta^+$.

After considering the joint velocity limits, the quadratic programming formulation becomes this form

$$\begin{aligned} &\text{minimize } \tfrac{1}{2}||\theta||_2^2 \\ &\text{subject to } J_E(\theta)\dot{\theta} = \dot{r}_E, \quad L(\theta)\dot{\theta} \leq 0, \quad \eta^- \leq \dot{\theta} \leq \eta^+. \end{aligned} \tag{9}$$

3 Neural Network Model

Define

$$p(u) = \sum_{i=1}^{p} \frac{1}{2}[h(L_i(\theta)u)]^2 + \sum_{i=1}^{n} \frac{1}{2}[[h(u_i - \eta_i^+)]^2 + [h(\eta_i^- - u_i)]^2] + \frac{1}{2}||J_E(\theta)u - \dot{r}_E||^2,$$

where $h(x) = \begin{cases} 0, & x \leq 0 \\ x, & x > 0 \end{cases}$. Then according to [8], the deterministic annealing neural network for problem (9) can be written as

$$\begin{aligned} \epsilon \tfrac{du}{dt} &= -\alpha p(u)u - (\left[\textstyle\sum_{i=1}^{p} h(L_i(\theta)u)L_i(\theta)^T\right] + h(u - \eta^+) - h(\eta^- - u) \\ &\quad + J_E(\theta)^T J_E(\theta)u - J_E(\theta)^T \dot{r}_E), \\ \tfrac{d\theta}{dt} &= u, \end{aligned} \tag{10}$$

[1] The notations in [4] are revised to be consistent with the notations in this paper.

where $L_i(\theta)$ is the i-th row vector of $L(\theta)$, ϵ is a scaling parameter which controls the convergence rate of the neural network and α is a weighting parameter.

In the deterministic annealing network, there are n neurons whereas the dual neural network adopted in [4] will need $m + n + p$ neurons. We can see that the structural complexity is greatly reduced by adopting the deterministic annealing network.

From the structure of the deterministic annealing network (10), we can see that the objective function, trajectory tacking constraint, joint and joint velocity limits and obstacle avoidance are weighted to get a balance between these constraints and objective function. When the link comes near the obstacle, the weight on obstacle avoidance can be increased until infinity to ensure no collision. So, the deterministic annealing network can be further developed as

$$\begin{aligned} \epsilon\tfrac{du}{dt} &= -\alpha\bar{p}(u)u - \left(\left[\textstyle\sum_{i=1}^{p}\gamma_i h(L_i(\theta)u)L_i(\theta)^T\right] + h(u-\eta^+) - h(\eta^- - u)\right.\\ &\quad \left. + J_E(\theta)^T J_E(\theta)u - J_E(\theta)^T\dot{r}_E\right), \\ \tfrac{d\theta}{dt} &= u, \end{aligned} \tag{11}$$

where $\gamma_i = \gamma\ln(O_iC_i/d_T)$ is the weighting coefficient on obstacle avoidance and

$$\bar{p}(u)=\sum_{i=1}^{p}\frac{1}{2}\gamma_i[h(L_i(\theta)u)]^2+\sum_{i=1}^{n}\frac{1}{2}[[h(u_i-\eta_i^+)]^2+[h(\eta_i^- -u_i)]^2]+\frac{1}{2}||J_E(\theta)u-\dot{r}_E||^2.$$

There are at least two advantages for this model compared with model (10). First, the obstacle avoidance term takes effect continuously, instead of suddenly. This can make the control variable (signal) change smoothly. In addition, the tracking accuracy can be sacrificed for avoiding obstacles when it is impossible to avoid the obstacle while accurately tracking the trajectory.

4 Simulation Results

In this section, the 7-DOF Mistsubishi PA10-7C manipulator is used to demonstrate the validity of the obstacle avoidance scheme and the online solution capability of the deterministic annealing network. The coordinates are setted up according to [2], and the structure parameters, joint limits, and joint velocity limits can also be found there. In the workspace there are two obstacle points, respectively $[-0.0454m; -0.0737m; 0.8367m]^T$, and $[-0.1541m; -0.0609m; 0.5145m]^T$. In this study, only the position of the end-point is concerned, then $m = 3$ and $n = 7$. The safety threshold d_T is chosen as $5cm$. And parameters for the deterministic annealing network are chosen as $\epsilon = 10^{-6}$, $\alpha = 1$ and $\gamma = 10$.

The desired motion of the end-effector is a circle of radius $r = 20cm$ with the revolute angle about the x axis $\pi/6$. The task time of the motion is $10s$ and the initial joint variables $\theta(0) = [0; -\pi/4; 0; \pi/2; 0; -\pi/4; 0]^T$. Fig. 2 illustrates the simulated motion of the PA10 manipulator in the 3D workspace, which is sufficiently close to the desired one with the tracking error less than $0.6mm$, as seen from Fig. 3.

The distances between obstacles and links are shown in Figs. 4, 5 and 6, where the links always tend to move away from the obstacle once they enter the danger zone. Fig. 7 shows the comparison between the case with and without obstacle avoidance

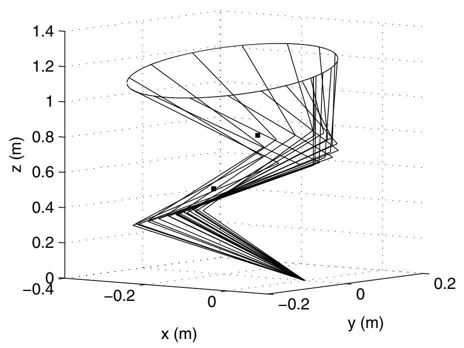


Fig. 2. Motion trajectory of PA10 manipulator while tracking a circle

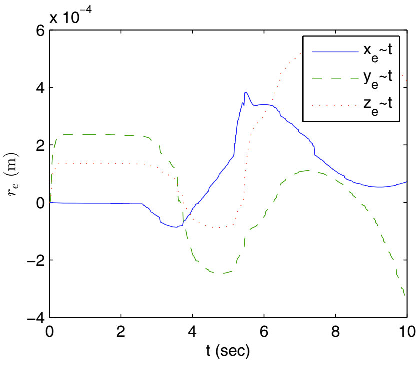


Fig. 3. Position Error of the end-effector

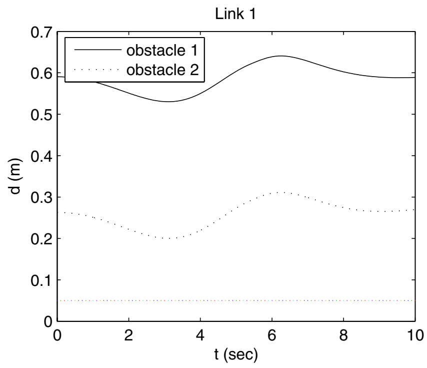


Fig. 4. Distance between obstacles and Link 1

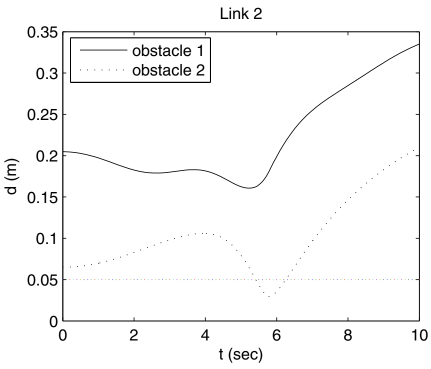


Fig. 5. Distance between obstacles and Link 2

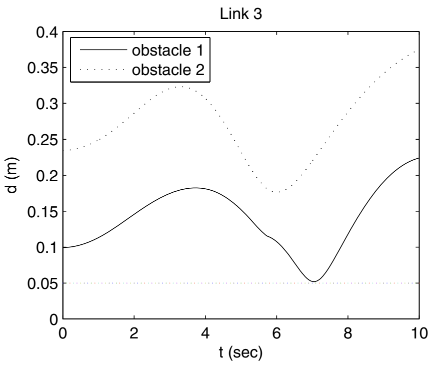


Fig. 6. Distance between obstacles and Link 3

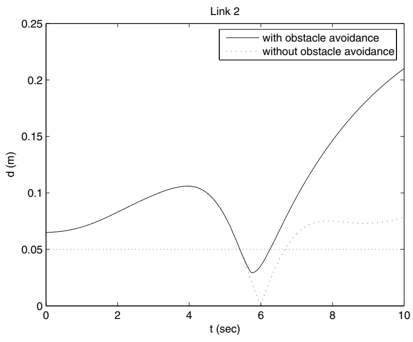


Fig. 7. Comparision of distance between obstacle 2 and link 2 with and without obstacle avoidance

scheme. Without obstacle avoidance, the link will collide with an obstacle as shown by the dotted line in Fig. 7. But when the proposed scheme is adopted, the link will move away from the obstacle once entering the danger zone, which can be observed from the solid line in Fig. 7.

5 Concluding Remarks

A new obstacle avoidance scheme is proposed and its quadratic programming formulation is presented. The solution to this time-varying optimization problem by the deterministic annealing network is simulated with the PA10 robot manipulators. The results show that this scheme is effective and can be performed in real time[2].

References

1. Maciekewski, A.A., Klein C.A.: Obstacle Avoidance for Kinematically Redundant Manipulators in Dynamicall Varying Enviorements. International J. of Robotics Research **4** (1985) 109–117
2. Wang, J., Hu, Q., Jiang, D.: A Lagrangian Network for Kinematic Control of Redundant Manipulators. IEEE Trans. Neural Networks **10** (1999) 1123–1132
3. Xia, Y., Wang, J.: A Dual Neural Network for Kinematic Control of Redundant Robot Manipulators. IEEE Trans. Syst. Man, Cybern. **31** (2001) 147–154
4. Zhang, Y., Wang, J.: Obstacle Avoidance for Kinematically Redundant Manipulators Using a Dual Neural Network. IEEE Trans. Syst. Man, Cybern. – Part B: Cybernetics **34** (2004) 752-759.
5. Liu, S., Wang, J.: A Dual Neural Network for Bi-Criteria Torque Optimization of Redundant Robot Manipulators. Lecture Notes in Computer Science, Vol. 3316. Springer-Verlag, Berlin Heidelberg New York (2004) 1142–1147
6. Kennedy, M.P., Chua, L.O.: Neural Networks for Nonlinear Programming. IEEE Trans. Circ. Syst. **35** (1988) 554–562.
7. Zhang, S., Constantinides, A.G.: Lagrange Programming Neural Networks. IEEE Transactions on Circ. Syst. **39** (1992) 441–452.
8. Wang, J.: A Deterministic Annealing Neural Network for Convex Programming. Neural Networks **5** (1994) 962–971
9. Xia, Y.: A New Neural Network for Solving Linear and Quadratic Programming Problems. IEEE Trans. Neural Networks **7** (1996) 1544–1547.

[2] This work was supported by the Hong Kong Research Grants Council under Grant CUHK4165/03E.

BP Networks Based Trajectory Planning and Inverse Kinematics of a Reconfigurable Mars Rover*

Liping Zhang[1,3], Shugen Ma[2,3], Bin Li[2],
Zheng Zhang[1], Guowei Zhang[2], and Binggang Cao[3]

[1] School of Mechanical Engineering, Xi'an Jiaotong University,
Xi'an, Shaanxi 710049, China
lipingzhang221@Sohu.com, zhangzh@mail.xjtu.edu
[2] Robotics Laboratory, Shenyang Institute of Automation (SIA),
Shenyang, Liaoning 110016, China
{libin,zhgw}@sia.ac.cn
[3] Department of Systems Engineering, Faculty of Engineering,
Ibaraki University, Japan
shugen@dse.ibaraki.ac.jp

Abstract. The inverse kinematics of series manipulators presents an inherent complexity due to their various structures and kinematics constraints. To a novel reconfigurable Mars rover's arm, the inverse kinematics had been solved by numerical method combined with space geometry relations, but the complex calculating process can not be utilized in real-time control. In this paper, some actions in common use are programmed in the child rover, and the BP neural network was used to solve the inverse kinematics and trajectory planning. To a desired trajectory, some solutions by direct kinematics algorithms and geometry relations corresponding to the trajectory were used as the BP network's training data set, and the non-linear mapping from joint-variable space to operation-variable space was obtained with iterative training and learning method. The results show the output of training is consistent with the desired path, and the obtained reference trajectory is smooth.

1 Introduction

There has been increasing interest in planetary Rover. A robotic approach to explore these bodies has the benefit of being able to achieve many things that a human could do but at lower cost without endangering human life [1]. Coordination and expandability are necessary in these system, the modular reconfigurable robots are thus becoming increasingly important. On the other hand, most actions of Planetary Rovers are planed off-line for remote distance and communication delay. At the same time, the space finiteness of the computer memory demands that the program must be simplest. Therefore the neural network which has higher intelligence, are used widely day by day [2],[3]. In this paper, Section II introduced the system structure of the reconfigurable Mars rover. The kinematic mode was analyzed in section III, and the BP network was constructed, trained in section IV. To validate the trained result, the error curves were analyzed according to various situations. At last, Section V discussed the conclusion of this paper.

* This work is supported by the National High Technology Research and Development Program of China (2002AA422130).

2 System Structure of Mars Rover

The robot introduced in this paper is a child-robot of the reconfigurable Mars Rover, which consists of one main body and six child-robots as shown in Fig.1. The main body can not move by itself, but the child-robots hold the main body by their manipulators and act as active wheels of the main body. If it is necessary, each child robot can separate from the main body and move around to undergo separate missions. Each child-robot consists of a triangle pedrail wheel and an arm for manipulation, which can exchange information with the main body or get their energy from the main body.

The child-robot has five revolute joints, as shown in Fig.2. The basement of it is a triangle pedrail wheel, which can rotate round the axis of Link1. The axes of Link2, Link3 and Link4 are parallel and the fifth axis of Link5 is vertical to the anterior three axes. The child-robot can work in locomotion mode or work mode, as shown in Fig.1. In work mode, its manipulator can move in the work space. For example, it can pick up a rock, catch up with other child-robots' connector or instrument. The child-robots can transform from locomotion mode to work mode autonomously, and vice versa. Its single chip processor used the FUJITSU MB90F549G which ROM capacity is 256 Kbytes. In order to simplify the inverse kinematics program and insure it can be loaded in single chip microprocessor, the neural networks which had been trained well are used in the inverse kinematics solution and the trajectory planning.

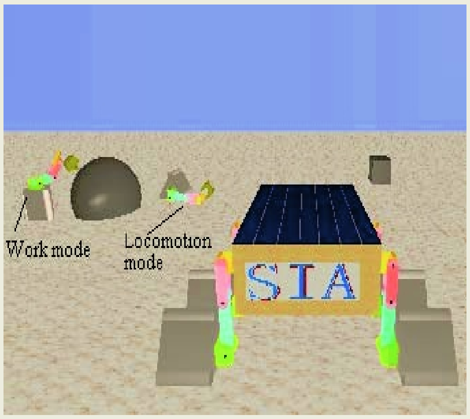


Fig. 1. Reconfigurable Mars rover

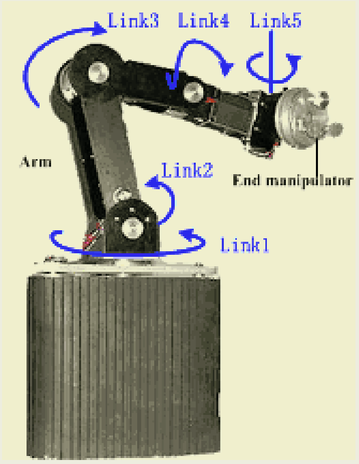


Fig. 2. Child-robot in work mode

3 Kinematic Model of Child Robot's Manipulator

The Denavit–Hartenberg method involves the allocation of coordinate frames to each link using a set of rules to locate the origin of the frame and the orientation axes. The poses of subsequent links are then defined by the homogeneous transformation matrix, which transforms the frame attached to link i–1 into a frame fixed to link i. This transformation is obtained from simpler transformations representing the three basic translations along, and three rotations about, the frames' x-, y- and z-axes [2].

The child-robot's D–H parameters of each link are shown in Table 1 [4]. Where θ_i is the joint angle, d_i is the joint offset, a_i is the link length, and α_i is the link twist. For revolute joint, a_i, α_i , d_i are constants and θ_i is the joint variable. The translation matrix between links is expressed by:

$$T_i^{i\text{-}1} = \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 & a_{i-1} \\ \sin\theta_i\cos\alpha_{i-1} & \cos\theta_i\cos\alpha_{i-1} & -\sin\alpha_{i-1} & -d_i\sin\alpha_{i-1} \\ \sin\theta_i\sin\alpha_{i-1} & \cos\theta_i\sin\alpha_{i-1} & \cos\alpha_{i-1} & d_i\cos\alpha_{i-1} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The position and pose of the manipulator is calculated by

$$T_6^0 = T_1^0(\theta_1)T_2^1(\theta_2)T_3^2(\theta_3)T_4^3(\theta_4)T_5^4(\theta_5)T_6^5(\theta_6)$$

Supposed $\mathbf{T}_0^6 = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$, then the position of the end manipulator in base coordinate can be represented as P = $[p_x , p_y, p_z]^T$, and the Euler angles are calculated by Eqs. 1a to 1c:

$$\varphi_z = \tan 2^{-1}(n_y, n_x) \tag{1a}$$

$$\varphi_y = \tan 2^{-1}(-n_z, n_x\cos\varphi_z + n_y\sin\varphi_z) \tag{1b}$$

$$\varphi_x = \tan 2^{-1}(a_x\sin\varphi_z - a_y\cos\varphi_z, o_y\cos\varphi_z - o_x\sin\varphi_z) \tag{1c}$$

Where $\tan 2^{-1}(x, y)$ is the arctangent function determined by the sign of x and y.

It can be seen that the forward kinematic problem can be easily solved by giving each joint angle.

Table 1. D–H parameters of child rover manipulator

Link	a_i(mm)	α_i (rad)	d_i(mm	q_i(rad)	Jonint range(rad)
Link1	0	0	247	0	-3.14~3.14
Link2	0	-1.14	0	-3.14	-1.57~1.57
Link3	180	0	0	0.52	-1.75~1.75
Link4	120	0	0	0.52	-1.75~1.75
Link5	100	1.14	0	0	-1.57~1.57

The solution of direct kinematics is a one-to-one mapping from joint to the Cartesian space, and the inverse kinematics is on the contrary. In engineering applications, the inverse kinematics is the base of locomotion planning, trajectory control and off-line programming. Giving the solution by one method is not at all, it has more demands on the calculation effect and computing precision. For most industrial or commercial manipulators, we can usually get a closed-form solution by an algebraic method. However, the main shortcomings of the algebraic method are the huge computation cost, requiring experience and the lack of a geometric concept. in most cases, the algebraic method is always associated with a geometric method [5],[6]. The inverse kinematics of the child robot in this paper had been achieved according to the algebraic method combined with the space geometric relations [7], but the complex calculating process can not be utilized in real-time control of the single chip microprocessor. Therefore, a rapid and close arithmetic must be searched for the child-robot' five freedom manipulator.

4 Construction of BP Neural Network

In the application of artificial neural network, a BP network is commonly used and its weights are modified by the negative gradient descent method. Increasing the number of layers can decrease the system error and improve the precision, but which complicate the network greatly and prolong the training time. At the same time, the proper number of neural knots and a suitable activation transfer function can also increase the system's precision. Here an executed line trajectory was adopted as the goal trajectory, which pose and position at two ends are expressed by:

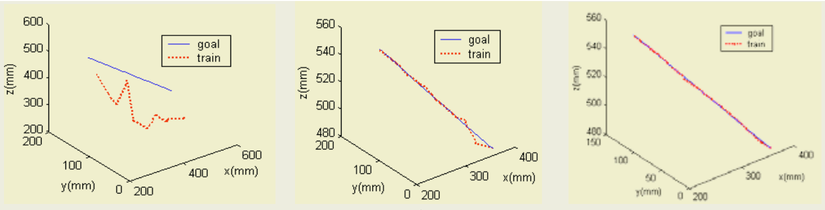
$$X_1 = [0, 1.5708, 0, 354.9, 0, 485.5]^T \ ; \quad Y_1 = [3.1416, 1.5707, -2.5882, 192.2, 118.7, 567.5]^T$$

The BP network was constructed by considering the training data set, the activation transfer function, the number of hidden layers and knots in every layer.

First, the selection of the training data set is important. The particularity of the rover's mechanics makes the space geometrical relationship must be considered in the trajectory planning. The space line X_1Y_1 was planed by rotate the joint 1first, the median three parallel joints second, and the fifth joint can not rotate if possible. According to the rule, the results of the forward kinematics were used as the training data, the solution of the inverse kinematics on the trajectory is implemented by using BP networks. The training data on the line were proportional spacing. The number of training data can affect the BP network's stability and convergence velocity, many simulation experiments were done to detect the least data which can make the network stationary, the results show that thirteen data on X_1Y_1 is proper.

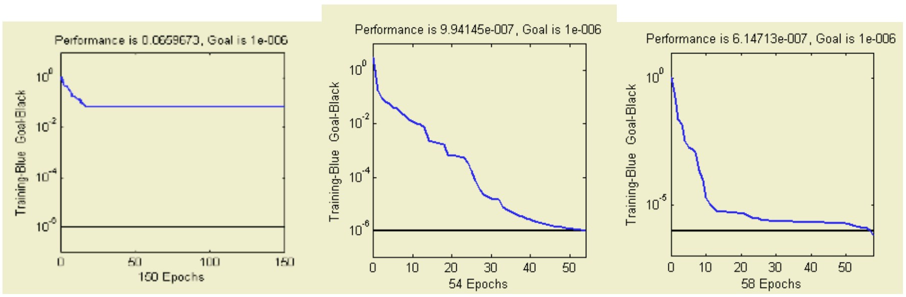
Second, the number of hidden layers and the knots in every layer is also very import for network's construction. In theory, a network which has deviation and at least one sigmoid hidden layer appended with a linear output layer can approach any rational function. The increase of layers may decrease the system error and improve its precision, but which complicate the network and prolong the training time. At the same time, more the knots has, higher error precision can get. The later can be observed and adjusted easily than the former. Therefore, increasing the number of the knots may be considered early.

In order to validate the trained inverse kinematics and linear trajectory, the goal trajectory and the trajectory constructed by the trained points were expressed in one figure. Fig. 3 is the trajectory comparison of varying layers and knots, the blue solid line is the goal and the red dotted line is the trained line. The Levenberg-Marquardt



(a) (b) (c)

Fig. 3. (a) has one hidden layer, the hidden and output layers have (12,5) knots respectively. (b) has two hidden layers, the knots number of two hidden layers and the output layer are (6,10,5) respectively. (c) has two hidden layers, the knots number of two hidden layers and the output layer are (12,10,5) respectively training method was utilized. The training goal is to make the error function less than 10^{-6}. Through many simulation experiments, two hidden layers which have 12 and 10 neural knots respectively are adopted. Fig. 4 is the training times curve. We can see, when the BP network has a hidden layer, the goal can not meet and the trajectory can not match well. When given two hidden layers, the network can convergence soon, too many layers can increase the network's complexity greatly, so two hidden layers were adopted here. If the layer number is certain, fewer knots in each layer will decrease the training time but the curve can not match very well, too many knots have no practice meaning and make the network more complex.



(a) (b) (c)

Fig. 4. (a), (b), (c) are the training time corresponding to the Fig. **3.** (a), (b), (c) respectively

At last, the activation transfer function is another important key for BP network. In output layer, the pureline function is adopted generally, log-sigmoid and tan-sigmoid functions are utilized in hidden layer. To the BP network constructed above, after fifty-eight times trains, the BP constructed by log-sigmoid function meets the error precession and agrees with the given space very well, but the BP constructed by tan-sigmoid meet the error precession after sixty-eight times trains, and agree with the goal not very well.

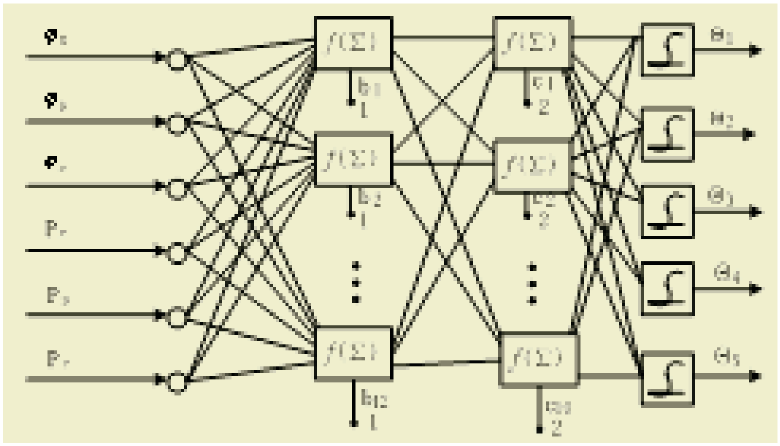


Fig. 5. Structure of BP networks

5 Conclusions

This paper applies BP networks to solve the trajectory planning and inverse kinematic questions of reconfigurable Mars rover. This method can meet the error requirement of the trajectory, avoid the complicated formula derivation and programming procedure, improve the calculation velocity and save the limited ROM space of the single chip microprocessor. It is more appropriate to improve the ability of the manipulator's real time and off-line control, which is important in planetary exploration.

Acknowledgements

The authors would like to thank Prof. Shigeo Hirose of Tokyo Institute of Technology for cooperating in this research.

References

1. JanIjspeert, A., Martinoli, A., Billard, A., et, al.: Collaboration through the Exploitation of Local Interactions in Autonomous Collective Robotics: The Stick Pulling Experiment. Autonomous Robots (2001) 149-171
2. Zhang, P., Lv, T., Song, L.: Study on BP Networks-based Inverse Kinematics of MOTOMAN Manipulator. Mechanical & Electrical Engineering Magazine, (2003) 56-59
3. Karlik, B., Aydin, S.: An Improved Approach to the Solution of Inverse Kinematics Problems for Robot Manipulators. Engineering Application of Artificial Intelligence, 13 (2000) 159-164
4. Xiong, Y.: Robotics. China Mechanical Press (1993)
5. Wang, X., Xu, S., Hao, J.: New Inferential Method and Efficient Solutions for Inverse Kinematics Equations of Robot MOTOMAN. Journal of China University of Mining & Technology, **30** (2001) 73-86
6. Wang, Q., Xu, X.: A New Inferential Method and Efficient Solutions for Inverse Kinematics Equations of PUMA Robot Manipulator. Robot, **20** (1998) 81-87
7. Zhang, L., Ma, S., Li, B., et al.: Kinematic Modeling and Trajectory Planning for Reconfigurable Planetary Robot System. Journal of Xi'an Jiaotong University (Natural Science), **39** (2005) 1110~1114

A Novel Path Planning Approach Based on AppART and Particle Swarm Optimization

Jian Tang, Jihong Zhu, and Zengqi Sun

State Key Laboratory of Intelligent Technology and Systems,
Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China
tjian99@mails.tsinghua.edu.cn

Abstract. Due to the NP-hard complexity, the path planning problem may perhaps best be resolved by stochastically searching for an acceptable solution rather than using a complete search to find the guaranteed best solution. Most other evolutionary path planners tend to produce jagged paths consisting of a set of nodes connected by line segments. This paper presents a novel path planning approach based on AppART and Particle Swarm Optimization (PSO). AppART is a neural model multidimensional function approximator, while PSO is a promising evolutionary algorithm. This path planning approach combines neural and evolutionary computing in order to evolve smooth motion paths quickly. In our simulation experiments, some complicated path-planning environments were tested, the result show that the hybrid approach is an effective path planner which outperforms many existing methods.

1 Introduction

The path planning problem that involves computing an optimal or near optimal collision free path between two locations is an optimization problem, where a series of configurations is sought, beginning from the initial location and ending at the goal location with respect to the problem criteria, e.g., distance, time, smoothness, energy and safety. The distance is the most typical criterion. Shorter paths are executed faster and require less energy. Path smoothness, on the other hand, enlarges the energy consumption and the execution time, but it is also a constraint and affects most mobile robots because of the bounded turning radius. For example, a car-like robot has this constraint due to the mechanical limitations of its steering angle. This type of path planning is used in a large number of robotics applications such as mobile robots navigation, manufacturing, assembly, transportation and services.

Path planning algorithms may be classified as exact and heuristic according to completeness [1]. Exact algorithms commit themselves to find an optimal solution if one exists, or prove that there is no feasible solution. On the contrary, heuristic algorithms aim to search for an acceptable path in a short time. Exact algorithms are usually computationally expensive; however, heuristic algorithms may fail to find a solution for some difficult problems.

Most of conventional path planning approaches [2], such as cell decomposition, road map and potential field, have some weakness in common. First of all, they lack adaptability and robustness. secondly, they are not suitable for dynamic environments because they generate a single path depending on a sequential search algorithm. This solution may become infeasible when a change in the environment is detected and a new solution has to be generated from scratch. To overcome the weakness of these approaches, genetic algorithm (GA) has been introduced into the field of path planning in recent years [3],[4],[5],[6]. However most GA approaches operate in a grid map or tend to produce jagged paths consisting of a set of nodes connected by line segments. To get smoother paths, more nodes must be given, which results in higher complexity, but still do not produce very smooth paths.

This paper presents a novel path planning approach based on AppART and Particle Swarm Optimization (PSO). AppART is an adaptive resonance theory low parameterized neural model which is able to incrementally approximate continuous-valued multidimensional functions from noisy data using biologically plausible processes[7], while PSO is a promising evolutionary algorithm [8]. This path planning approach combines neural and evolutionary computing in order to evolve smooth motion paths quickly.

In the rest of this paper, we will give some brief introduction to PSO and AppART in section 2 and 3. In section 4, the proposed hybrid path planning algorithm is presented. Section 5 presents the conducted experimental results. Conclusions are presented in section 6.

2 PSO Algorithm

The development of particle swarm optimization (PSO) was based on observations of the social behavior of animals such as bird flocking, fish schooling, and swarm theory. Each individual in PSO is assigned with a randomized velocity according to its own and its companions' flying experiences, and the individuals, called particles, are then flown through hyperspace. Compared with GA, PSO has some attractive characteristics. It has memory, so knowledge of good solutions is retained by all particles; whereas in GA, previous knowledge of the problem is destroyed once the population changes. Besides, PSO has constructive cooperation between particles, particles in the swarm share information between them.

A Particle Swarm Optimization algorithm is defined by the evolution of a population of particles, represented as vectors in a n-dimensional real-valued search space of possible problem solutions. Every particle has a position vector $\vec{x}$ encoding a candidate solution to the problem and a velocity vector $\vec{v}$. Moreover, each particle contains a small memory that stores its individual best position seen so far $\vec{p}$ and a global best position $\vec{p}_g$ obtained through communication with its neighbor particles in the same group. There exist many kinds of topologies for particles' group organization, such as Gbest and Lbest, the most common ones, and some local cluster social networks [9]. In Gbest, all particles know each other, while in Lbest, each individual has only two neighbors.

Information about good solutions spreads through the swarm, and thus the particles tend to move to good areas in the search space. At each time step t, the velocity is updated and the particle is moved to a new position. This new position is calculated as the sum of the previous position and the new velocity:

$$\vec{x}(t+1)=\vec{x}(t)+\vec{v}(t+1). \tag{1}$$

The update of the velocity from the previous velocity is determined by the following equation:

$$\vec{v}(t+1)=w\times\left(\vec{v}(t)+c_1\phi_1\left(\vec{p}(t)-\vec{x}(t)\right)+c_2\phi_2\left(\vec{p}_g(t)-\vec{x}(t)\right)\right). \tag{2}$$

where w controls the magnitude of $\vec{v}$, c_1 and c_2 are positive constants and ϕ_1 and ϕ_2 are uniformly distributed random numbers in [0, 1]. Changing velocity this way enables the particle to search around its individual best position and global best position.

3 AppART as a Function Approximation Method

AppART [7] is an adaptive resonance theory low parameterized neural model which is able to incrementally approximate continuous–valued multidimensional functions from noisy data using biologically plausible processes. It performs a higher order Nadaraya–Watson regression:

$$y=\hat{F}_h(x)=\frac{\sum_{l=1}^{L} y_j^{(l)} \exp\left(-\frac{\left\|x-x^{(l)}\right\|}{2\sigma^2}\right)}{\sum_{l=1}^{L} \exp\left(-\frac{\left\|x-x^{(l)}\right\|}{2\sigma^2}\right)}. \tag{3}$$

An AppART has the capability of approximating any function with any desired degree of accuracy, because it is an stable learning neural implementation of (3). AppART has an architecture of four layers, being a layer of afferent or input nodes, F1, a classification layer, F2, a prediction layer, P, and an output layer O (see Fig. 1).

Input x is presented to F1 and then propagated to F2. F2 activation v measures the degree of membership of x to F2–defined categories. If v does not meet the GF2 vigilance a node is committed in F2. Prediction o is calculated by P and O. If o does not meet the GO error criterion a match tracking algorithm takes place.

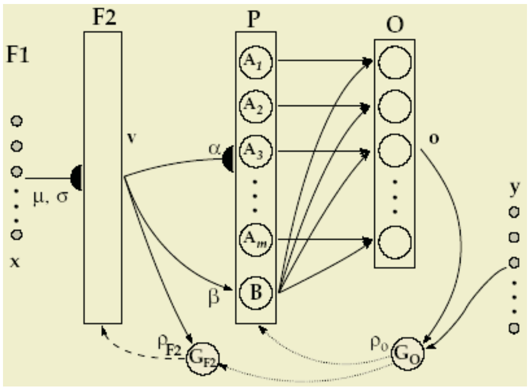


Fig. 1. AppART architecture.

Besides its capability of approximating any function, An AppART is attractive because its fast on-line learning algorithm, training data only need to propagate forward once, unlike most other back-propagation neural networks, where data may propagate forward and backwards many times until an acceptable error is found.

4 A Novel Path Planning Algorithm Based on PSO and AppART

Using PSO and AppART, this section introduces a novel hybrid algorithm to solve the path planning problem. This new approach combines neural and evolutionary computing in order to evolve smooth motion paths quickly.

The algorithm works by using (x, y) coordinates to act as control points to describe a curved path through a scene. The x values are fixed at equally spaced intervals. The y values of all control points are encoded into a particle of PSO algorithm. With PSO algorithm's running, these y values of control points are changing in hope of finding a better path. At the same time, the evolving (x, y) coordinate pairs of any particle are used by an AppART as training examples to construct a path function $y = f(x)$. Plotting $y = f(x)$ along the x-axis creates a smooth continuous path. From this path function, we can calculate the path feasibility and length in order to evaluate the PSO particle containing these (x, y) coordinate pairs. In order to accomplish a path this smooth, most evolutionary path planners would have to use a very large chromosome containing quite a lot of genes to describe the path, while $O(n)$ coordinate pairs is enough for our algorithm, where n is the number of obstacles in the environment. The pseudocode of the algorithm is as follows:

```
PSO-AppART Path Planning algorithm
{
 Initialize(P);// initialize P paths as particles in PSO
 for t = 1 to max_generation
 {
  for i = 1 to P
  {
   Evaluate fitness for particle i;
   If needed, update particle's $\vec{p}$ and $\vec{p}_g$;
   Update particle's velocity using formula (2);
   Update particle's position using formula (1);
  }
  Terminate if $\vec{p}_g$ meets problem requirements;
 }
}
Evaluate fitness for particle
{
 Decode all $(x, y)$ coordinate pairs from input particle;
 Train AppART with these decoded $(x, y)$ points;
 Calculate the feasibility and length of the curved path;
 Return a fitness based on the path feasibility and length;
}
```

In the algorithm, every particle in PSO represents a candidate path and contains $O(n)$ pairs of (x, y) coordinates. In every iteration of PSO algorithm, every particle is evaluated according to its encoded control points. These points are decoded as training data to train an AppART neural network. Then the AppART regression path is used to calculate the fitness of the particle. The particle which represents feasible and short path will get a good evaluation and dominate the evolutionary PSO process.

5 Simulation Experiment

To validate the novel path planning algorithm, A variety of polygonal environments were tested. Figure 2 shows a typical path planning solution in a static environment. The curve from point S to point D across many polygonal obstacles represents the path found by our algorithm.

Unlike most other evolutionary path planners which tend to produce jagged paths consisting of a set of nodes connected by line segments, a lot of simulation experiments indicate that the paths produced by our novel neuro-evolutionary path planner have very good smoothness, as is very important for some mobile robots with mechanical constraints.

In real-life applications, robot navigates in dynamic environments adjusting and modifying its planned paths according to the changes that occur in the environment, as is shown in figure 3. At first, the robot was on position S, it calculated an original path using the algorithm presented in section 4. While the robot was executing the original path from position S to point R, the two largest obstacles were moving across the environment either. When the robot arrived at point R, the original path was not feasible any more, so the robot had to plan a new path to point D, which is shown in the right picture of figure 3. Dynamic environments' experiments indicate that our new algorithm is very fast that it can be executed in real-time mode in most of cases.

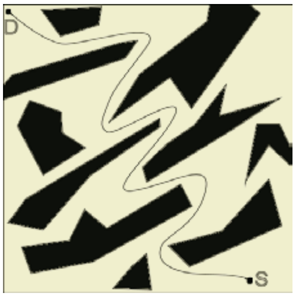


Fig. 2. Static environment.

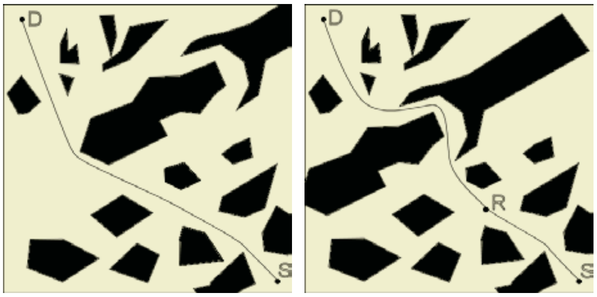


Fig. 3. Dynamic environment.

6 Conclusions

The proposed path planning algorithm in this paper combines neural and evolutionary computing in order to evolve smooth motion paths quickly. The promising PSO algorithm and the compact path representation make our path planning algorithm very fast, and the regression capacity of AppART makes the produced path very smooth. Simulation experiments show that the hybrid approach is an effective path planner.

Acknowledgements

This work was jointly supported by the National Key Project for Basic Research of China (Grant No: G2003cb312205) and the National Natural Science Foundation of China (Grant No:60174018, 90205008, 60305008).

References

1. Hwang, Y. K. , Ahuja, N.: Gross Motion Planning - A Survey. ACM Computing Survey, **24** (1992) 219-291
2. Latombe, J. C.: Robot Motion Planning. Boston: Kulwer Academic publishers (1991)
3. Shibata, T., Fukuda, T., Kosuge, K., Arai, F.: Selfish and Coordinative Planning for Multiple Mobile Robots by Genetic Algorithm. Proc. of the 31st Conf. on Decision and Control, **3** (1992) 2686-2691
4. Leung, C. H., Zalzala, A. M. S.: A Genetic Solution for the Motion of Wheeled Robotic Systems in Dynamic Environments. Int. Conf. on Control, **1** (1994) 760-764
5. Trojanowski, K., Michalewicz, Z. , Xiao, J.: Adding Memory to the Evolutionary Planner/Navigator. Proc. of the 4th IEEE ICEC (1997) 483-487
6. Gerke, M.: Genetic Path Planning for Mobile Robots. Proc. of the American Control Conference, **4** (1999) 2424-2429
7. Martí, L., Policriti, A., García, L.: AppART: An ART Hybrid Stable Learning Neural Network for Universal Function Approximation. Hybrid Information Systems, Heidelberg: Springer (2002) 93-120
8. Kennedy, J., Eberhart, R.: Particle Swarm Optimization. Proc IEEE Int. Conf. Neural Networks, Perth, Australia (1995) 1942-1948
9. Mendes, R. C., Rocha, P. M. , Neves, J.: Particle Swarms for Feedforward Neural Network Training. Proc. Int. Joint Conf. Neural Networks (2002) 1895-1899

A Neuro-fuzzy Controller for Reactive Navigation of a Behaviour-Based Mobile Robot*

Anmin Zhu, Simon X. Yang**, Fangju Wang, and Gauri S. Mittal

School of Engineering, University of Guelph, Guelph, ON, N1G 2W1, Canada

Abstract. In this paper, a novel neuro-fuzzy controller is proposed for reactive navigation control of a mobile robot in complex environments with uncertainties. A fuzzy logic system is designed with three behaviours: target seeking, obstacle avoidance, and barrier following. A learning algorithm based on neural network technique is developed to tune the parameters of membership functions, which smooths the trajectory generated by the fuzzy logic system. Under the control of the proposed neuro-fuzzy model, the mobile robot can preferably avoid static and moving obstacles, and generate smooth trajectories toward the target in various situations. The effectiveness and efficiency of the proposed approach are demonstrated by simulation studies.

1 Introduction

In recent years, many approaches to steering mobile robots have been developed. Graph-based methods [1] combine a graph searching technique with obstacle pruning to generate trajectory for mobile robots. These graph-based methods first need to construct a data structure that is then used to find paths between configurations of the robot and the target. The data structure tends to be very large, and the geometric search computation required is complex and expensive. In addition, these methods are usually used for robot path planning only without considering the robot motion control. Artificial potential field methods [2] are proposed to get rid of the computational complexity in graph-based methods. These methods assume that each obstacle in the environment exerts a repulsive force on the mobile robot, and the target exerts an attractive force. These two types of forces are combined at every step and used to determine the next step robot movement. But the algorithm is limited to a point robot and allows the robot approach too close to the surface of obstacles. In order to overcome the shortcomings of potential field methods, a vector field histogram (VFH) method [3] was proposed, which was used for fast-running mobile robots and less likely to get trapped in local minima comparing with the potential field methods. However the VFH methods ignored the dynamic and kinematic constraints of mobile robots. Fuzzy logic approaches to mobile robot navigation and obstacle avoidance have been investigated by several researchers. Fuzzy systems have the ability to treat uncertain and imprecise information using linguistic rules, thus, they offer possible implementation of human knowledge and experience. Saffiotti

* This work was supported by Natural Sciences and Engineering Research Council (NSERC), Canada.

** Corresponding Author. E-mail: `syang@uoguelph.ca`.

$et\ al.$ [4] proposed some fuzzy logic methods for robot navigation. However, the drawbacks mainly result the difficulty in defining proper membership functions and lack of a systematic procedure to transform expert knowledge into the rule base. In addition, the process of tuning the parameters of fuzzy rules may be rather difficult. Neural network learning techniques are capable of automatically obtaining proper system parameters to achieve a satisfactory performance. Therefore, the neuro-fuzzy controller becomes one of the most popular techniques in robot navigation study. Godjevac [5] proposed a neuro-fuzzy model for a mobile robot to avoid obstacles . More than 600 rules are formulated, where many of them are redundant and there is no method to suppress the useless rules.

In this paper, a novel neuro-fuzzy model, including a set of linguistic behaviour-based fuzzy rules and a learning algorithm, is presented for navigation control of a mobile robot. Under the control of the model, the mobile robot can preferably avoid all static and moving obstacles automatically in various situations.

2 The Proposed Neuro-fuzzy Controller

While a mobile robot is moving in an unknown and changing environment, it is important to compromise between avoiding collisions with obstacles and moving toward targets, depending on the sensed information about the environment. Fuzzy systems can be used to realise this reactive strategy. Neural networks have the ability to learn knowledge from "experience". Combing these two paradigms, a neuro-fuzzy controller is developed to navigate a mobile robot in unknown environments. Fig. 1 shows a brief structure of the proposed neuro-fuzzy controller, which consists of a fuzzy controller and a learning adaptation model.

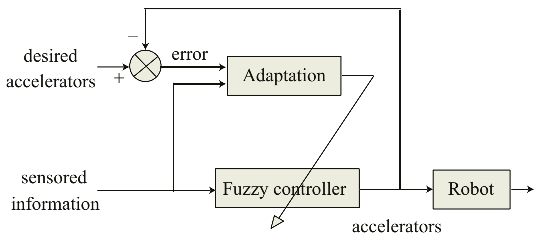


Fig. 1. The schematic diagram of the proposed neuro-fuzzy controller.

The structure of the proposed neuro-fuzzy controller with the relationship to fuzzy logic system is shown in Fig. 2, where $\{u_1, u_2, u_3, u_4, u_5\} = \{d_l, d_f, d_r, t_d, r_s\}$ is the input vector; d_l, d_f and d_r represent obstacle distances from the left, front and right sensor groups, respectively; t_d denotes the angle between the robot moving direction and the line connecting the robot centre with the target; r_s is the current robot speed; $\{y_1, y_2\} = \{a_l, a_r\}$ is the output vector; and a_l and a_r represent the accelerations of the left and right wheels.

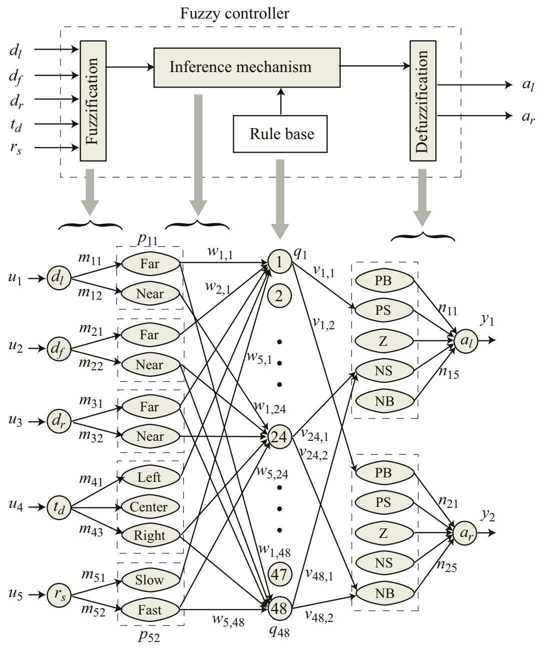

Fuzzy controller
Fuzzification
Inference mechanism
Rule base
Defuzzification
d_l d_f d_r t_d r_s
a_l a_r
u_1 u_2 u_3 u_4 u_5
m_{11} m_{12} m_{21} m_{22} m_{31} m_{32} m_{41} m_{43} m_{51} m_{52}
p_{11} p_{52}
Far Near Far Near Far Near Left Center Right Slow Fast
$w_{1,1}$ $w_{2,1}$ $w_{5,1}$ $w_{1,24}$ $w_{5,24}$ $w_{1,48}$ $w_{5,48}$
q_1 q_{48}
$v_{1,1}$ $v_{1,2}$ $v_{24,1}$ $v_{24,2}$ $v_{48,1}$ $v_{48,2}$
PB PS Z NS NB
n_{11} n_{15} n_{21} n_{25}
y_1 y_2

Fig. 2. The structure of the proposed neuro-fuzzy model and the relationship with the fuzzy logic system. m_{ij},n_{ls}: the centres of MFs for input, output variables; p_{ij}: the degree of Memberships; q_k: conjunction degree of IF part of rules; $w_{i,k}$: weights related to m_{ij}; $v_{k,l}$: weights related to n_{ls}.

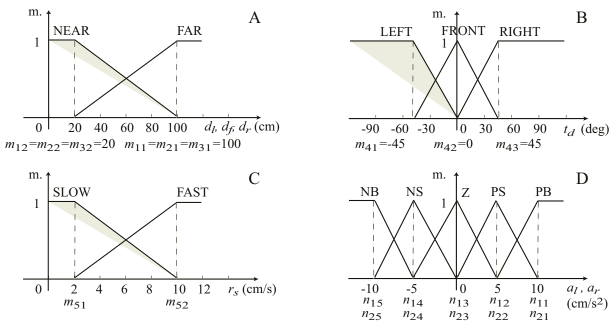


Fig. 3. Membership functions for input variables (A: obstacle distances; B: target direction; C: current robot speed) and output variables (D: accelerations of two wheels). m_{ij}, n_{ls}: the centres of membership functions.

In the fuzzy controller, the process has three steps, which are fuzzification, fuzzy inference, and defuzzification. The fuzzification procedure maps the crisp input values to the linguistic fuzzy terms with the membership values between 0 and 1. In this paper, triangle functions are chosen to represent fuzzy membership functions. Membership functions are shown in Fig. 3. The outputs of the fuzzification procedure are as follows.

$$p_{ij} = \begin{cases} 1 - \frac{2|u_i - m_{ij}|}{\sigma_{ij}}, & \text{if } m_{ij} - \frac{\sigma_{ij}}{2} < u_i < m_{ij} + \frac{\sigma_{ij}}{2}, \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

where $i = 1, 2, \ldots, 5$ is the index number of input signals; $j = 1, 2, \ldots, 5$ is the index number of terms of the input variables; p_{ij} is the degree of membership for the ith input corresponding to the jth term of the input variable; u_i is the ith input signal to the fuzzy controller; m_{ij} is the centre of the membership function corresponding to the ith input and the jth term of the input variable; and σ_{ij} is the width of the membership function corresponding to the ith input and the jth term of the input variable.

The inference mechanism is responsible for decision making in the fuzzy controller using approximate reasoning. Forty eight rules are formulated for the proposed controller. The output of the aggregation procedure to get degree of IF part of every rule is given as

$$q_k = \min\{p_{1k_1}, p_{2k_2}, p_{3k_3}, p_{4k_4}, p_{5k_5}\}, \tag{2}$$

where q_k is the conjunction degree of the IF part of the kth rule, k =1, 2, ..., 48; and p_{ik_1} is the degree of the membership for the ith input contributing to the kth rule, $i = 1, 2, \ldots, 5; k_i = 1, 2, \ldots, 5$.

Defuzzification procedure maps the fuzzy output from the inference mechanism to a crisp signal. The "centre of gravity (CoG)" method is used in the proposed controller. The values of the output variables a_l and a_r are given as

$$a_l = \frac{\sum_{k=1}^{48} v_{k,1} q_k}{\sum_{k=1}^{48} q_k} \quad \text{and} \quad a_r = \frac{\sum_{k=1}^{48} v_{k,2} q_k}{\sum_{k=1}^{48} q_k}, \tag{3}$$

where $v_{k,1}$ and $v_{k,2}$ denote the estimated values of the outputs provided by the kth rule, which are related to the centre of membership functions of the output variables.

3 The Learning Algorithm

To smooth the trajectory generated by the fuzzy logic model, a learning algorithm based on neural network technique is developed. The effect of the output variables mainly depends on the centre of membership functions when the rule base is designed. The widths of membership functions can be disregarded, which are usually set to be constants. The membership function centre values of the input and output variables may be improved by the neural network learning property. The vector of 21 parameters could be tuned in the proposed model, which is set as

$$\begin{aligned} Z = \{ \; & m_{11}, m_{12}, m_{21}, m_{22}, m_{31}, m_{32}, m_{41}, m_{42}, m_{43}, m_{51}, m_{52}, \\ & n_{11}, n_{12}, n_{13}, n_{14}, n_{15}, n_{21}, n_{22}, n_{23}, n_{24}, n_{25} \; \}. \end{aligned} \tag{4}$$

In this paper, the least-mean square (LMS) algorithm is used to adjust the system parameters by minimising errors between the desired output and the actual output of the system using the following criterion function. Thus, the parameters would be adapted as

$$m_{ij}(t+1) = m_{ij}(t) - \varepsilon_m \frac{\partial E}{\partial m_{ij}}, \quad i = 1, \ldots, 5; j = 1, 2, 3, \tag{5}$$

$$n_{ls}(t+1) = n_{ls}(t) - \varepsilon_n \frac{\partial E}{\partial n_{ls}}, \quad l = 1, 2; s = 1, \ldots, 5, \tag{6}$$

where ε_m and ε_n are the learning rates. Therefore, it is only necessary to calculate the partial derivative of the criterion function with respect to the particular parameter to get the expression for each of them.

$$\begin{aligned}
\frac{\partial E}{\partial m_{ij}} &= \sum_{l=1}^{2} \left(\frac{\partial E}{\partial y_l} \frac{\partial y_l}{\partial q_k} \frac{\partial q_k}{\partial p_{ij}} \frac{\partial p_{ij}}{\partial m_{ij}} \right) \\
&= -2 \sum_{l=1}^{2} \left[(y_l - \hat{y}_l) \frac{v_{k,l} \sum_{k=1}^{48} q_k - \sum_{k=1}^{48} v_{k,l} q_k}{(\sum_{k=1}^{48} q_k)^2} \right] \times \frac{\text{sign}(u_i - m_{ij})}{\sigma_{ij}}, \\
\frac{\partial E}{\partial n_{ls}} &= \sum_{l=1}^{2} \left(\frac{\partial E}{\partial y_l} \frac{\partial y_l}{\partial v_{k,l}} \frac{\partial v_{k,l}}{\partial n_{ls}} \right) = (y_l - \hat{y}_l) \frac{q_k}{\sum_{k=1}^{48} q_k}.
\end{aligned} \tag{7}$$

4 Simulation Studies

The model parameters are initialised first in the training phase. After the learning, the better parameters of the membership functions are given as

$$\begin{aligned}
&\{m_{11}, m_{12}, m_{21}, m_{22}, m_{31}, m_{32}, m_{41}, m_{42}, m_{43}, m_{51}, m_{52}\} \\
&= \{9.5, 5.4, 0, -5.5, -9.3, 9.4, 5.6, 0, -5.3, -9.4\}, \quad (8) \\
&\{n_{11}, n_{12}, n_{13}, n_{14}, n_{15}, n_{21}, n_{22}, n_{23}, n_{24}, n_{25}\} \\
&= \{10, 5, 0, -5, -10, 10, 5, 0, -5, -10\}. \quad (9)
\end{aligned}$$

To show the effectiveness of the the proposed controller, several cases are studied. Before the learning, It generates a winding trajectory as shown in Figs. 4A and 4B

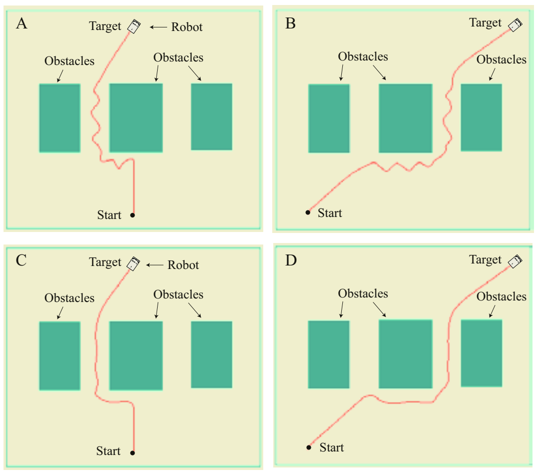


Fig. 4. Mobile robot trajectories. A, B: winding trajectories without the learning algorithm. C, D: smooth trajectories with the proposed learning algorithm.

because of the model parameters. After the training, the trajectories become smoother as shown in Figs. 4C and 4D.

5 Conclusions

A neuro-fuzzy control system combining a fuzzy controller and a neural network learning technique is proposed for real-time reactive navigation of a mobile robot. Under the control of the proposed neuro-fuzzy model, the mobile robot can autonomously generate a smooth trajectory toward the target. Different from other neuro-fuzzy methods, several features of the proposed approach are as follows: (1) The accelerations are used as the controller outputs, which can resolve the speed jump problem; (2) The structure of the proposed neuro-fuzzy model is very simple with only 11 hidden nodes and 48 fuzzy rules; (3) The proposed learning algorithm provides a easy way to tune the model parameters and smooth the trajectory; And (4) the proposed approach keeps the physical meanings of the membership functions during the training.

References

1. Jiang, K., Seneviratne, L., Earles, S.: Time-optimal smooth-path motion planning for a mobile robot with kinematic constraints. Robotica, **15** (1997) 547–553
2. Alsultan, K., Aliyu, M.: A new potential field-based algorithm for path planning. Journal of Intelligent and Robotic Systems, **17** (1996) 265–282
3. Borenstein, J., Koren, Y.: The vector field histogram - fast obstacle avoidance for mobile robots. IEEE Journal of Robotics and Automation, **7**(3) (1991) 278–288
4. Saffiotti, A., Ruspini, E.H., Konolige, K.: Using fuzzy logic for mobile robot control. In Zimmermann, H.J., ed.: Practical Applications of Fuzzy Technologies. Kluwer Academic Publisher (1999) 185–206
5. Godjevac, J.: A learning procedure for a fuzzy system: application to obstacle avoidance. In: Proceedings of the International Symposium on Fuzzy Logic, Zurich (1995) 142–148

Research on the Calibration Method for the Heading Errors of Mobile Robot Based on Evolutionary Neural Network Prediction*

Jinxia Yu[1,2], Zixing Cai[1], Xiaobing Zou[1], and Zhuohua Duan[1,3]

[1] College of Information Science and Engineering, Central South University, Changsha, Hunan 410083, China
[2] Department of Computer Science & Technology, Henan Polytechnic University, Jiaozuo, Henan 454003, China
melissa2002@163.com
[3] Department of Computer Science, Shaoguan University Shaoguan, Guangdong 512003, China

Abstract. Fiber optic gyros (FOG) is the important sensor for measuring the heading of mobile robot. Combined with measured data of E-Core RD1100 interferometric FOG made by American KVH company, the paper analyses the common calibration for the heading errors of mobile robot caused by the drift of FOG, and uses the method of evolutionary neural networks prediction to compensate it. By the experiments of mobile robot prototype, the paper also proves this method can reduce the error influence of FOG on the heading of mobile robot and enhance the localization precision of mobile robot navigation.

1 Introduction

The localization of mobile robot, where am I, is the basic problem for the navigation technique [1]. Dead reckoning is broadly utilized in mobile robot localization because it is self-contained and always capable of providing the robot with a pose estimate. The information generally required for dead reckoning purpose is the direction, speed of mobile robot and so on. The precise heading is one of the parameters that needs to be accurately measured in order to describe the pose of mobile robot. Fiber optic gyros (FOG) is the important sensor for measuring the heading of mobile robot for its obvious advantages [2-5]. However, one key concern with the use of FOG as a means for improving dead reckoning is the inherent drift because it is necessary to perform numeric integral of the output readings which will accumulate and grow without bound, resulting in ever-increasing heading errors. Therefore, it needs to accurately identify and validly calibrate the heading errors of mobile robot caused by the drift of FOG.

Faced with the above problem, many researchers have been making a great deal of study on it. In many methods, neural network is able to identify the non-linear system for its strong function approximation ability, so it is broadly applied to the domain of

* This work is supported by the National Natural Science Foundation of China (No. 60234030).

prediction [6,7]. Back propagation neural network (BPNN) is commonly adopted in predictive algorithms, but it inevitably exists the problem of local minima, slower training speed and lower efficiency. Radial basis function neural network (RBFNN) overcomes the above drawbacks to some extent, whereas it has too many neurons and evaluating parameters for the hidden layer. Traditional learning algorithms are hardly efficient to estimate the parameters, while genetic algorithm (GA) has unique advantages of dealing with the non-linear system problem and has great capabilities in computation. The application of GA to neural network optimization would make neural network has the self-evolution and self-adaptability capability. It is proposed that RBFNN combines with GA to predict the heading errors of mobile robot based on the analysis of the drift of FOG and the study on common calibration for the heading errors of mobile robot in the paper. By the experiments of mobile robot prototype, the validity of the method is also proved.

2 Analysis and Calibration of the Heading Errors

FOG used in our research is KVH E-Core RD1100 gyro that is open-loop interferometric FOG. RD1100 gyro is a true single-axis rotation rate sensor insensitive to cross-axis motion and errors. Technical specifications for E-Core RD1100 can be gotten from reference 8.

2.1 Analysis of the Heading Errors

The reason why FOG causes the drift errors is relatively complex. The performance of FOG is mainly characterized by static bias drift (bias stability), scale factor linearity and stability, and a random noise component termed angle random walk (ARW). The drift errors of FOG are very complex because of the above factors correlated with and influenced on each other. It can be shown in the figure 1 for the output of stationary FOG without any calibration from which we can see the change of angular rate is not obvious while the angular drift varies from 50 degree to 150 degree within 5 minutes. It indicates that mobile robot cannot accurately navigate if not calibrating the heading errors caused by the drift of FOG.

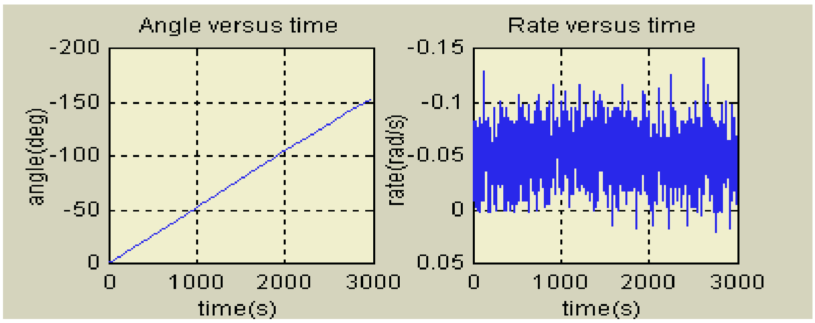


Fig. 1. Output of stationary FOG without any calibration (Temp:28.2°~34.4°)

2.2 Methods of Common Calibration

(1) Elimination of Static Bias Drift

Static bias drift is the deviation degree of the output of FOG from its average value. That is to say, a stationary FOG should output reading of 0.00º/s while in practice the reading will differ from that value due to static bias drift. Static bias drift is the inherent error of FOG that can be solved by software method. A common compensation procedure for static bias drift is to collect several samples (for, say, 10-20 seconds) with the FOG stationary and immediately prior to every mission. Then the average of those samples is computed as static bias drift which is subtracted from subsequent measurements during the immediately following mission.

(2) Elimination of Determinate Trend

After the elimination of static bias drift, it is discovered that the drift error of FOG is unsteady stochastic time series and is formed by the superposition of the determinate part and stochastic part. Hence we must eliminate the determinate trend of the drift error of FOG. Here is adopted the stepwise regression and is estimated the multiple regressive coefficient by the least square method.

With the common calibration above, the angular drift of FOG can be controlled between 15 and 26 deg/h, but mobile robot using FOG to improve the dead reckoning still exists the heading errors because of FOG's real-time application, inherent instability and sensitivity to the variation of the ambient temperature. In order to realize the more accurate navigation, it proposed that RBFNN combines with GA to predict the drift of FOG after calibrated by the common methods and applied it to the predictive compensation of the heading errors in the navigation system of mobile robot.

3 Predictive Method Using Neural Network

3.1 Model of RBFNN

Similar to feedforward multilayered neural network, RBFNN is composed of the input, hidden and output layer. Figure 2 shows a RBFNN with n input nodes and a output node. In order to predict the output $\hat{X}(t)$, we suppose the input is n dimensional vector $X=[X(t\text{-}1), X(t\text{-}2),\dots, X(t\text{-}n)]^T$ and use Gauss function as RBF. It is supposed that the j-th hidden node of RBF related to the input vector X is related as follows.

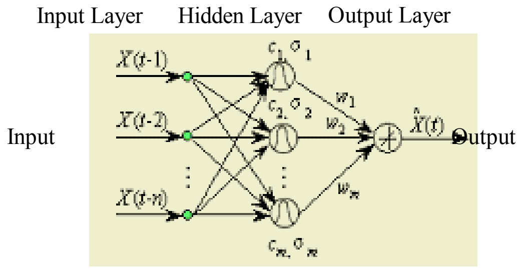


Fig. 2. Structure of RBFNN

$$f_j(X) = \exp(-\frac{\| X - c_j \|^2}{2\sigma_j^2}) \cdot \tag{1}$$

where, $c_j = (c_{j_1}, c_{j2}, \cdots, c_{j_n})$ is the central vector of the *j*-th RBF of the hidden layer; $\sigma_j = (\sigma_{j_1}, \sigma_{j2}, \cdots, \sigma_{j_n})$ is the parameter of the *j*-th RBF of the hidden layer which determinates the width for the distance response from the *j*-th RBF of the hidden layer to its corresponding input vector X; j =1,2,…,m (m is the number of the neurons in the hidden layer) . Then, the output of RBFNN is shown as follows.

$$\hat{X}(t) = \sum_{j=1}^{m} w_j f_j(X) \cdot \tag{2}$$

where, w_j is the weight values from the the *j*-th neuron node of the hidden layer to the output layer. Hence, RBFNN realizes the non-linear mapping from R^n to R^m by the linear combination of non-linear RBF.

3.2 Network Training Based on GA

The kernel of RBFNN training is able to determinate the central position and the width of RBF and the weight values from hidden layer to output layer quickly and effectively. When using GA to optimize RBFNN, it needs to encode for chromosome, to determinate the number of the group and the adaptive function, to perform the operation of selection, crossover and mutation, and lastly get the terminal condition. The algorithm is explained as following.

(1) Encoding method of chromosome: The real number encoding method is adopted in the paper. The structure parameters in RBFNN include the central position c_{ji} and the width σ_{ji} of the RBF and the weight values in the output layer w_j et. al. where, i=1,2,…,n; j =1,2,…,m and the number of the evaluating parameters are $2mn+m$, the encoding of the individual G_θ is shown as follows.

$$G_\theta = \{(w_1, \ldots, w_m, c_{11}, \ldots, c_{1n}, c_{m1}, \ldots, c_{mn}, \sigma_{11}, \ldots, \sigma_{1n}, \sigma_{m1}, \ldots, \sigma_{mn}) \\ \mid w_j, c_{ji}, \sigma_{ji} \in \mathrm{R}; i = 1,2, \ldots, n; j = 1,2, \ldots m\}$$

(2) Size of the group: the size of the group will directly affect the final result and the efficiency of GA. The smaller the optimal effect is not good because of existing the local optimum solution, the bigger the computational complexity is too high to make against solving. The size of the group adopted the number 10 by experience. So $P=\{X_k \mid k=1,2, \ldots, P_{size}\}$.

(3) Adaptive function: the adaptive function is the evaluated standard of good or not for the individual G_θ whose value is positive. It is used the goal function of RBFNN as the adaptive function here in which $X(t)$ denotes that the actual output of FOG, $\hat{X}_k(t)$ the predictive output of NN.

$$f(G_\theta) = \varepsilon(t) = \frac{1}{2Psize} \sum_{k=1}^{Psize} [X_k(t) - \hat{X}_k(t)]^2 \cdot \tag{3}$$

(4) Genetic operators: it includes selection, crossover, mutation. Selection operation denotes that the ordinary individual be exposed to being replaced by a new

comer according to the probability (see formula 4), Crossover operation denotes that the Elitist individual copy its section of chromosome to the ordinary one's. Mutation is a method of improving the ordinary ones' quality. One mutation operation only changes one gene at a random position.

$$P(G_\theta) = \frac{f(G_\theta)}{\sum_{i=1}^{10} f(G_i)} . \tag{4}$$

The population of the chromosome group is 11in which ten of chromosomes are ordinary ones. Only the best one of the all chromosomes has the privilege of crossover. The generation cycle is ended when there is no new elitist one emerges after 40 continuous generation cycles.

4 Experiments of Mobile Robot Prototype

In order to compensate the drift of FOG, the output of FOG is used as the input of RBFNN, and the output of motorized precision rotating table as the output of it. There is an appropriate mapping for RBFNN by using GA to optimize it. The sum-squared error is used as the target function and the training of RBFNN is terminated when evolutionary generation is more than 260 generation for the error is small engough.

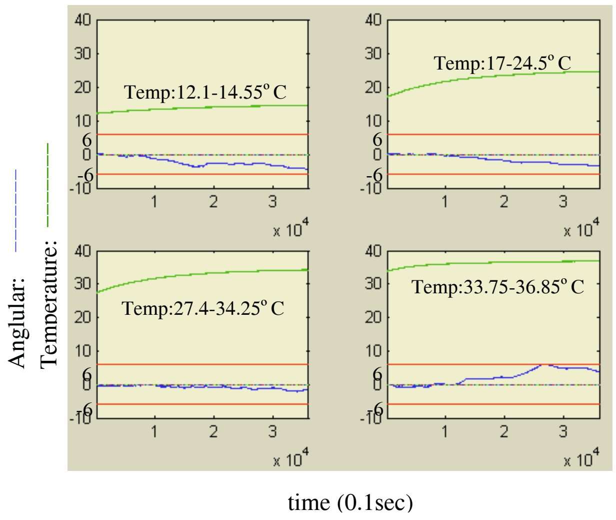


Fig. 3. Output of FOG at different temperature with evolutionary neural network calibration

Based on the prototype platform of mobile robot, the heading errors of mobile robot caused by the drift of FOG is compared before and after the calibration using evolutionary neural network. The output rate of FOG at continuous ten sampling time is used as the input of RBFNN to predict the output of FOG at next sample time. Before the compensation of RBFNN, the output of FOG and the ambient temperature are taken up the nonlinearity in some degree and the drift of FOG is 15~26 deg/h after the common calibration with the variable temperature from 10 to 40°C. With compensation, the drift of FOG is controlled to 3~6 deg/h under the same experiment environment which is closed to the output standard of KVH E-Core RD 1100 gyro at constant temperature. Figure 3 illustrates the drift of FOG with the calibration of evolutionary neural network can be controlled in 6 degree in an hour(3600 seconds) at different ambient temperature. By all appearances, it can reduce the drift error of FOG affected by the ambient temperature to a great extent and realize the more accurate dead reckoning of mobile robot in its navigation system.

5 Conclusions

Aimed at the accumulated heading errors by dead decking in mobile robot localization, evolutionary neural network is applied to predict and calibrate the heading errors of mobile robot caused by the drift of FOG after its common calibration in this paper. With the strong function approximation of RBFNN and the optimization of GA, the stability and anti-jamming capability of FOG are greatly improved, so is the pose resolution of mobile robot. By the experiments of mobile robot prototype, the results show this method is feasible and effective.

References

1. Cai, Z.X., He, H.G., Chen, H.: Some Issues for Mobile Robot Navigation Under Unknown Environments (In Chinese). Control and Decision, **17** (2002) 385-391
2. Barshan, B., and Durrant-Whyte, H. F.: Inertial Navigation Systems for Mobile Robots. IEEE Trans. on Robotics and Automation, **11** (1995) 328-342
3. Borenstein, J.: Experimental Evaluation of a Fiber Optics Gyroscope for Improving Dead-reckoning Accuracy in Mobile Robots. In: IEEE International Conference on Robotics and Automation, Leuven, Belgium (1998) 3456-3461
4. Bennett, S.M., Dyott, R., Allen, D. et al.: Fiber Optic Rate Gyros as Replacements for Mechanical Gyros. American Institute of Aeronautics & Astronautics (1998) 1315-1321
5. Chung, H., Ojeda, L., Borenstein, J.: Accurate Mobile Robot Dead-reckoning with a Precision-calibrated Fiber Optic Gyroscope. IEEE Trans. on Robotics and Automation, **17** (2001) 80-84
6. Meng, Z.Q., Cai Z.X.: Identification Method of Nonlinear Systems Based on Parallel Genetic Algorithm (in Chinese). Control and Decision, **18** (2003) 368-370
7. Amir, F.A., Samir, I.S.: A Comparison Between Neural Network Forecasting Techniques-Case Study: River Flow Forecasting. IEEE Trans. on Neural Networks, **10** (1999) 402-409
8. KVH, 8412 W. 185th St., Tinley Park, IL 60477, USA. http://www.kvh.com

Adaptive Neural-Network Control for Redundant Nonholonomic Mobile Modular Manipulators

Yangmin Li[1], Yugang Liu[1], and Shaoze Yan[2]

[1] Faculty of Science and Technology, University of Macau,
Av. Padre Tomas Pereira S.J., Taipa, Macao S.A.R., P.R. China
{ymli,ya27401}@umac.mo

[2] Department of Precision Instruments and Mechanology, Tsinghua University,
Beijing 100084, P.R. China
yansz@tsinghua.edu.cn

Abstract. This paper discusses the trajectory following issue for redundant nonholonomic mobile modular manipulators. Dynamic model is established and an adaptive neural-network controller is developed to control the end-effector to follow a desired spacial trajectory. The proposed algorithm doesn't need any priori dynamics and provides a new solution for stabilization of redundant robotic self-motions. Simulation results for a real robot demonstrate the proposed algorithm is effective.

1 Introduction

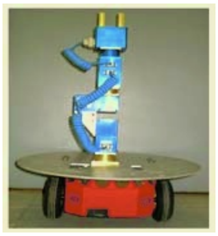
A nonholonomic mobile modular manipulator can be defined as a kind of robot integrating a N-degree of freedom (DOF) modular manipulator together with a M-wheeled nonholonomic mobile platform. If the integrated structure has more DOFs than required, it is called a redundant one. This integration extends the workspace of the entire robot drastically. However, the nonholonomic constraints, the interactive motions, as well as the self-motions make the trajectory following task difficult to realize. Neural networks (NNs) with characteristics of not needing exact priori dynamic parameters and universal approximators are being widely used for robotic control.

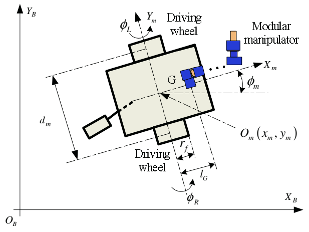
In related research work, back-propagation (BP) NN was used for vibration control of a 9-DOF redundant modular manipulator [1]. A multi-layer NN controller, which did not need off-line learning, was designed to control rigid robotic arms [2]. A fuzzy-Gaussian NN controller was proposed for trajectory tracking control of mobile robots [3]. A dual NN was presented for the bi-criteria kinematic control of redundant manipulators [4]. A sliding mode adaptive NN controller was devised for control of mobile modular manipulators [5].

In this paper, the dynamic model is established in Section 2. An adaptive NN controller (ANNC) is designed in task space in Section 3. Simulations are carried out in Section 4. Finally, some conclusions are given in Section 5.

2 Dynamic Modeling

In this paper, a 3-wheeled nonholonomic mobile platform is studied as shown in Fig. 1(a), and only end-effector positions $x = [p_x \;\; p_y \;\; p_z]^T$ are concerned. The mobile platform is supposed to just move on a horizontal plane. The coordinate system can be defined as follows: an arbitrary inertial base frame $O_B X_B Y_B Z_B$ is fixed on the motion plane, while a frame $O_m X_m Y_m Z_m$ is attached to the mobile platform. The parameters can be observed in Fig. 1(b) in details.


(a) A real robot



(b) Platform motion

Fig. 1. A real mobile modular manipulator and its motion on a plane

Define $\xi = \left[x_m \; y_m \; \phi_m \; \phi_L \; \phi_R\right]^T$, then the nonholonomic velocity constraints can be described as follows, [6]

$$\begin{bmatrix} C_m & S_m & -\frac{d_m}{2} & -r_f & 0 \\ C_m & S_m & \frac{d_m}{2} & 0 & -r_f \\ -S_m & C_m & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} r_f \cdot C_m/2 & r_f \cdot C_m/2 \\ r_f \cdot S_m/2 & r_f \cdot S_m/2 \\ -r_f/d_m & r_f/d_m \\ 1 & 0 \\ 0 & 1 \end{bmatrix} = 0 \tag{1}$$

In short $A(\xi) \cdot S(\xi) = 0$.

Define $q = \left[\phi_L \; \phi_R \; q_1 \cdots q_N\right]^T$, and $\zeta = \left[\xi^T \; q_1 \cdots q_N\right]^T$, then the Jacobian matrix can be derived by

$$J = \frac{\partial x}{\partial \zeta^T} \cdot \bar{S}(\xi) = \frac{\partial x}{\partial \zeta^T} \cdot \begin{bmatrix} S(\xi) & 0_{5\times N} \\ 0_{N\times 2} & I_{N\times N} \end{bmatrix} \tag{2}$$

The constrained dynamics can be determined by Eqn. 3, see [5] in details.

$$H \cdot \ddot{\zeta} + V \cdot \dot{\zeta} + G = B \cdot \left(\tau + J^T \cdot F_{ext}\right) + C \cdot \lambda \tag{3}$$

Where H, V and G denote the inertial matrix, the centripetal and coriolis matrix, and the gravitational force vector. $B = \left[0_{n\times 3} \; I_{n\times n}\right]^T$, $C = \left[A(\xi) \; 0_{3\times N}\right]^T$, $\tau = \left[\tau_L \; \tau_R \; \tau_1 \cdots \tau_N\right]^T$, $F_{ext} = \left[F_{ext}^x \;\; F_{ext}^y \;\; F_{ext}^z\right]^T$ is an external-force vector.

From Eqn. 2

$$\dot{x} = J \cdot \dot{q}, \;\; \dot{\zeta} = \bar{S} \cdot \dot{q}. \tag{4}$$

Solving Eqn. 4 and its derivative, yields

$$\begin{aligned} \dot{\zeta} &= \bar{S} J^\dagger \dot{x} + \bar{S}\left(I_n - J^\dagger J\right)\dot{q}_s \\ \ddot{\zeta} &= \bar{S}\left[J^\dagger \ddot{x} + \left(I_n - J^\dagger J\right)\ddot{q}_s\right] + \left(\dot{\bar{S}} - \bar{S} J^\dagger \dot{J}\right)\left[J^\dagger \dot{x} + \left(I_n - J^\dagger J\right)\dot{q}_s\right] \end{aligned} \tag{5}$$

Where $J^\dagger = J^T \cdot \left(J \cdot J^T\right)^{-1}$ is the Moore-Penrose generalized inverse of J; $\dot{q}_s \in \Re^n$ is an arbitrary joint velocity vector; $J^\dagger \cdot \dot{x}$ is a least-square solution; $\left(I_n - J^\dagger \cdot J\right) \cdot \dot{q}_s \in \aleph\left(J\right)$, the null space of J, is a homogeneous solution.

Define $J_\aleph \in \Re^{n\times(n-m)}$ as a matrix with all its columns being the normalized bases of $\aleph\left(J\right)$, $J_E^\dagger = \left[\, J^\dagger \quad | \quad J_\aleph \,\right]$, $\dot{x}_\aleph = J_\aleph^T \cdot \dot{q}_s$, and $x_E = \left[\, x^T \quad | \quad x_\aleph^T \,\right]^T$. Substituting Eqn. 5 into 3, and left multiplying $\left(J_E^\dagger\right)^T \cdot \bar{S}^T$, yields

$$\bar{H} \cdot \ddot{x}_E + \bar{V} \cdot \dot{x}_E + \bar{G} = \bar{\tau} \tag{6}$$

Where $\bar{H} = \left(J_E^\dagger\right)^T \cdot \bar{S}^T \cdot H \cdot \bar{S} \cdot J_E^\dagger$, $\bar{V} = \left(J_E^\dagger\right)^T \cdot \bar{S}^T \cdot \left[H \cdot \left(\dot{\bar{S}} - \bar{S} \cdot J^\dagger \cdot \dot{J}\right) + V \cdot \bar{S}\right]$, $\bar{G} = \left(J_E^\dagger\right)^T \cdot \bar{S}^T \cdot G$, $\bar{\tau} = \left(J_E^\dagger\right)^T \cdot \bar{S}^T \cdot B \cdot \left(\tau + J^T \cdot F_{ext}\right)$; $\left(J_E^\dagger\right)^T \cdot \bar{S}^T \cdot C \cdot \lambda = 0$.

Remark 1. For any $r \in \Re^n$, $r^T \cdot \bar{H} \cdot r \geq 0$.

Remark 2. For any $r \in \Re^n$, $r^T \cdot \left(\dot{\bar{H}} - 2\bar{V}\right) \cdot r = 0$.

Remark 3. If J is full rank, $J_E^\dagger$ is invertible, and $J_E = \left(J_E^\dagger\right)^{-1} = \left[\, J^T \quad | \quad J_\aleph \,\right]^T$.

Remark 4. $\bar{H}$, $\bar{V}$, $\bar{G}$ are all bounded as long as the Jacobian J is full rank.

3 Controller Design

Let x_d, $\dot{x}_d$ and $\ddot{x}_d$ be the desired trajectory, velocity and acceleration in task space. The desired self-motions can be used to fulfil a secondary task. In this paper, the system is assumed to be far away from singularity, physical limits, and obstacles. So, $\dot{q}_{sd}$ and $\ddot{q}_{sd}$ can be selected for the optimization problem of: $min\{\dot{q}^T \cdot \dot{q}\}\ \ subject\ \ to\ \ \dot{x} = J \cdot \dot{q}$. Then, $x_{\aleph d}$, $\dot{x}_{\aleph d}$ and $\ddot{x}_{\aleph d}$ can be determined.

Let $x_{Ed} = [\, x_d^T \quad | \quad x_{\aleph d}^T \,]^T$, then the error system can be defined by

$$e\left(t\right) = x_E\left(t\right) - x_{Ed}\left(t\right), \ \ \dot{x}_r\left(t\right) = \dot{x}_{Ed}\left(t\right) - \Lambda \cdot e\left(t\right), \ \ r\left(t\right) = \dot{x}_E\left(t\right) - \dot{x}_r\left(t\right). \tag{7}$$

Where $r\left(t\right)$ is the tracking error measure; $\Lambda \in \Re^{n\times n} > 0$ is a constant matrix.

Substituting Eqn. 7 into 6, yields

$$\bar{H} \cdot \dot{r}\left(t\right) + \bar{V} \cdot r\left(t\right) + \bar{H} \cdot \ddot{x}_r + \bar{V} \cdot \dot{x}_r + \bar{G} = \bar{\tau} \tag{8}$$

It is verified that a multilayer perceptron (MLP) trained with BP algorithm can approximate any continuous multivariate functions to any desired degree of accuracy, provided that sufficient hidden neurons are available [7]. To ensure rapid convergence, multiple-input single-output (MISO) MLPs with only one hidden layer are applied in this paper as shown in Fig. 2(a). Output of this MLP is:

$$f_{NN}\left(x, w_{ih}, w_{ho}, \theta_h, \theta_o\right) = \sum_{j=1}^{N_h}\left[\varphi\left(\sum_{i=1}^{N_i} x_i \cdot w_{ihji} + \theta_{hj}\right) \cdot w_{hoj}\right] + \theta_o \tag{9}$$

Where $\varphi\left(\circ\right)$ is the activation function named hyperbolic tangent function; N_i and N_h represent neuron numbers; x is the inputs; w_{ih}, w_{ho}, θ_h and θ_o denote weights and thresholds accordingly; the subscript "i, h and o" represents the input, hidden and output layer respectively.

Define $x_{in} = \begin{bmatrix} \zeta^T & \dot{q}^T & x_{Ed}^T & \dot{x}_{Ed}^T & \ddot{x}_{Ed}^T \end{bmatrix}^T$, $h(x_{in}) = \bar{H} \cdot \ddot{x}_r + \bar{V} \cdot \dot{x}_r + \bar{G}$. From Remark 4 and Eqn. 7, all the elements $h_k (k = 1, 2, \cdots n)$ are bounded as long as J keeps full rank. Then, they can be approximated by MISO NNs,

$$h_k(x_{in}) = h_{NNk}(x_{in}) + \epsilon_k(x_{in}) \tag{10}$$

Where $h_{NNk} = f_{NN}(x_{in}, w_{kih}, w_{kho}, \theta_{kh}, \theta_{ko})$; ϵ_k is the approximated error.

Let $\hat{w}_{kih}$, $\hat{w}_{kho}$, $\hat{\theta}_{kh}$ and $\hat{\theta}_{ko}$ be estimates of w_{kih}, w_{kho}, θ_{kh} and θ_{ko} respectively, define $\hat{h}_{NNk} = f_{NN}\big(x_{in}, \hat{w}_{kih}, \hat{w}_{kho}, \hat{\theta}_{kh}, \hat{\theta}_{ko}\big)$, then the Taylor series expansions of h_{NNk} around $\hat{h}_{NNk}$ can be derived

$$\begin{aligned}\tilde{h}_{NNk} = &\sum_{j=1}^{N_h}\left\{\sum_{i=1}^{N_i}\left[\frac{\partial \hat{h}_{NNk}}{\partial w_{kihji}} \cdot \tilde{w}_{kihji} + O\big(\tilde{w}_{kihji}^2\big)\right]\right\} + \frac{\partial \hat{h}_{NNk}}{\partial \theta_{ko}} \cdot \tilde{\theta}_{ko} + O\big(\tilde{\theta}_{ko}^2\big)\\ &+ \sum_{j=1}^{N_h}\left\{\frac{\partial \hat{h}_{NNk}}{\partial w_{khoj}} \cdot \tilde{w}_{khoj} + \frac{\partial \hat{h}_{NNk}}{\partial \theta_{khj}} \cdot \tilde{\theta}_{khj} + O\big(\tilde{w}_{khoj}^2\big) + O\big(\tilde{\theta}_{khj}^2\big)\right\}\end{aligned} \tag{11}$$

Where $\tilde{h}_{NNk} = h_{NNk} - \hat{h}_{NNk}$, $\tilde{w}_{kihji} = w_{kihji} - \hat{w}_{kihji}$, $\tilde{w}_{khoj} = w_{khoj} - \hat{w}_{khoj}$, $\tilde{\theta}_{khj} = \theta_{khj} - \hat{\theta}_{khj}$, and $\tilde{\theta}_{ko} = \theta_{ko} - \hat{\theta}_{ko}$; $O\big(\tilde{w}_{kihji}^2\big)$, $O\big(\tilde{w}_{khoj}^2\big)$, $O\big(\tilde{\theta}_{khj}^2\big)$ and $O\big(\tilde{\theta}_{ko}^2\big)$ are higher-order terms.

The ANNC is given by Eqn. 12, and the control system is shown in Fig. 2(b).

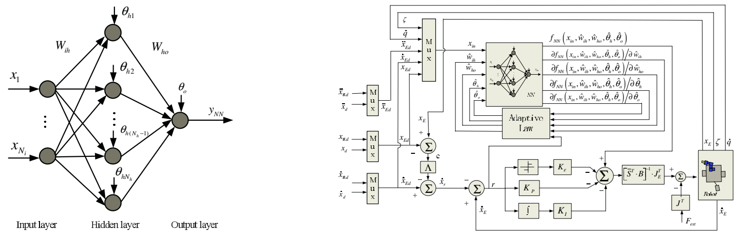
$$\tau = \big(\bar{S}^T \cdot B\big)^{-1} \cdot J_E^T \cdot \left\{\hat{h}_{NN} - K_P \cdot r - K_I \cdot \int_0^t r(t)\,dt - K_\epsilon \cdot sgn(r)\right\} - J^T \cdot F_{ext} \tag{12}$$

Where $\hat{h}_{NN} \in \Re^n$ forms the adaptive NN term; $K_P, K_I \in \Re^{n\times n}$ are proportional and integral gain matrices of the PID controller; $K_\epsilon = diag\begin{Bmatrix} k_{\epsilon 1} & k_{\epsilon 2} & \cdots & k_{\epsilon n}\end{Bmatrix}$ is the gain matrix of the robust term, and its elements are selected as follows:

$$k_{\epsilon k} \geq \left|\sum_{j=1}^{N_h}\left\{\sum_{i=1}^{N_i} O\big(\tilde{w}_{kihji}^2\big) + O\big(\tilde{w}_{khoj}^2\big) + O\big(\tilde{\theta}_{khj}^2\big)\right\} + O\big(\tilde{\theta}_{ko}^2\big)\right| + |\epsilon_k| \tag{13}$$

Substituting Eqn. 12 into 8, and considering 6 at the same time, yields

$$\bar{H} \cdot \dot{r} + \bar{V} \cdot r + K_P \cdot r + K_I \cdot \int_0^t r(t)\,dt + K_\epsilon \cdot sgn(r) + \tilde{h}_{NN} + \epsilon = 0 \tag{14}$$



(a) A MISO NN (b) An adaptive NN controller

Fig. 2. A MISO NN and an adaptive NN controller

Theorem 1. *If $K_P > 0$, $K_I^T = K_I > 0$, the closed-loop system in Eqn. 14 is asymptotically stable under the adaptation laws given by Eqn. 15. The error signals are convergent with time, i.e., $e(t), \dot{e}(t) \to 0$, as $t \to +\infty$.*

$$\begin{aligned} \dot{\hat{w}}_{kihji} &= -\mathit{\Gamma}_{w_{kihji}} \cdot r_k \cdot \frac{\partial \hat{h}_{NNk}}{\partial w_{kihji}}, \qquad & \dot{\hat{\theta}}_{khj} &= -\mathit{\Gamma}_{\theta_{khj}} \cdot r_k \cdot \frac{\partial \hat{h}_{NNk}}{\partial \theta_{khj}}, \\ \dot{\hat{w}}_{khoj} &= -\mathit{\Gamma}_{w_{khoj}} \cdot r_k \cdot \frac{\partial \hat{h}_{NNk}}{\partial w_{khoj}}, & \dot{\hat{\theta}}_{ko} &= -\mathit{\Gamma}_{\theta_{ko}} \cdot r_k \cdot \frac{\partial \hat{h}_{NNk}}{\partial \theta_{ko}}. \end{aligned} \tag{15}$$

Where $\mathit{\Gamma}_{w_{kihji}}$, $\mathit{\Gamma}_{w_{khoj}}$, $\mathit{\Gamma}_{\theta_{khj}}$, and $\mathit{\Gamma}_{\theta_{ko}}$ are positive constants. The terms with partial differentiation can be derived from Eqn. 10, details will not be listed here.

Proof. Considering the following nonnegative Lyapunov candidate:

$$\begin{aligned} V_S &= \tfrac{1}{2} \cdot r^T \cdot \bar{H} \cdot r + \tfrac{1}{2} \cdot \left[\int_0^t r(t)\,dt\right]^T \cdot K_I \cdot \left[\int_0^t r(t)\,dt\right] \\ &\quad + \tfrac{1}{2} \cdot \sum_{k=1}^{n} \left\{ \sum_{j=1}^{N_h} \left[\sum_{i=1}^{N_i} \left(\frac{\tilde{w}_{kihji}^2}{\mathit{\Gamma}_{w_{kihji}}} \right) + \frac{\tilde{w}_{khoj}^2}{\mathit{\Gamma}_{w_{khoj}}} + \frac{\tilde{\theta}_{khj}^2}{\mathit{\Gamma}_{\theta_{khj}}} \right] + \frac{\tilde{\theta}_{ko}^2}{\mathit{\Gamma}_{\theta_{ko}}} \right\} \geq 0 \end{aligned} \tag{16}$$

The time derivative of Lyapunov candidate is

$$\begin{aligned} \dot{V}_S &= r^T \cdot \left\{ \bar{H} \cdot \dot{r} + K_I \cdot \left[\int_0^t r(t)\,dt\right] \right\} + \tfrac{1}{2} \cdot r^T \cdot \dot{\bar{H}} \cdot r + \tfrac{1}{2} \cdot \sum_{k=1}^{n} \left\{ \frac{\tilde{\theta}_{ko} \cdot \dot{\tilde{\theta}}_{ko}}{\mathit{\Gamma}_{\theta_{ko}}} \right\} \\ &\quad + \tfrac{1}{2} \cdot \sum_{k=1}^{n} \left\{ \sum_{j=1}^{N_h} \left[\sum_{i=1}^{N_i} \left(\frac{\tilde{w}_{kihji} \cdot \dot{\tilde{w}}_{kihji}}{\mathit{\Gamma}_{w_{kihji}}} \right) + \frac{\tilde{w}_{khoj} \cdot \dot{\tilde{w}}_{khoj}}{\mathit{\Gamma}_{w_{khoj}}} + \frac{\tilde{\theta}_{khj} \cdot \dot{\tilde{\theta}}_{khj}}{\mathit{\Gamma}_{\theta_{khj}}} \right] \right\} \end{aligned} \tag{17}$$

Notice that $\dot{\tilde{w}}_{kihji} = -\dot{\hat{w}}_{kihji}$, $\dot{\tilde{w}}_{khoj} = -\dot{\hat{w}}_{khj}$, $\dot{\tilde{\theta}}_{khj} = -\dot{\hat{\theta}}_{khj}$, $\dot{\tilde{\theta}}_{ko} = -\dot{\hat{\theta}}_{ko}$. Substituting Eqn. 12 into 15, then substituting the result together with Eqns. 14,16 into 18, and considering Remark 2 at the same time, yields

$$\dot{V}_S \leq -r^T \cdot K_P \cdot r \leq 0 \tag{18}$$

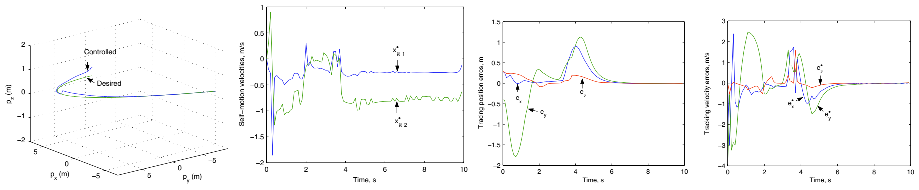
Therefore V_S is a Lyapunov function, iff $r = 0$, V_S and $\dot{V}_S$ equal to zero. According to LaSalle's theorem, the system is asymptotically stable and $r \to 0$ as $t \to +\infty$.

Define $\ell_p = \{x(t) \in \Re^n : \|x\|_p < \infty\}$ the $p-norm$ space. From Eqns. 16 and 18, $r(t) \in \ell_2$. According to Eqns. 7, $e(t) \in \ell_2 \cap \ell_\infty$, $\dot{e}(t) \in \ell_2$, and $e(t) \to 0$, as $t \to +\infty$. It is obvious that the higher-order terms in Eqn. 11 are bounded, so $K_\epsilon \in \ell_\infty$. Then, from Eqn. 14, $\dot{r}(t) \in \ell_\infty$. Since $r(t) \in \ell_2$ and $\dot{r}(t) \in \ell_\infty$, $r(t) \to 0$ as $t \to +\infty$, which is followed by $\dot{e}(t) \to 0$. End of proof.

4 Simulation Results

The simulation is performed on a real robot as shown in Fig. 1(a), in which $N = 3$ and $n = 5$. Simulation time is 10 seconds and the end-effector is to follow the desired trajectory in Fig. 3(a). The gain matrices and constants are: $N_h = 200$, $K_P = diag\{50\}$, $K_I = diag\{10\}$, $\mathit{\Gamma}_{w_{kihji}} = \mathit{\Gamma}_{w_{khoj}} = \mathit{\Gamma}_{\theta_{khj}} = \mathit{\Gamma}_{\theta_{ko}} = 0.01$, $K_\epsilon = diag\{50\}$, $\mathit{\Lambda} = diag\{2.0\}$. To examine the disturbance suppression ability of the ANNC, an external force $F_{ext}^z = 1N$ is applied to the end-effector along the direction Z_m at the time instant $t = 2s$.

The desired and the controlled locus are shown in Fig. 3(a). Figure 3(b) shows the self-motion velocities. The tracking position and velocity errors are given by Fig. 3(c)–3(d). From these figures, we can see that the ANNC is effective to control the end-effector to follow a desired spacial trajectory.



(a) Spacial locus (b) Self-motions (c) Position errors (d) Velocity errors

Fig. 3. Simulation results

5 Conclusions

Dynamic model is established and an ANNC is developed for a general mobile modular manipulator. The proposed controller does not need precise knowledge of dynamic parameters in advance and can suppress bounded external disturbances effectively. The torque instability problem caused by self-motions of the redundant robot can be solved by the presented control algorithm. The simulation is carried out on a real redundant nonholonomic mobile modular manipulator, which has verified the effectiveness of the dynamic modeling method and the controller design method.

References

1. Li, Y., Liu, Y., Liu, X. and Peng, Z.: Parameters Identification and Vibration Control in Modular Manipulators. IEEE/ASME Trans. Mechatronics. **9** (2004) 700–705
2. Lewis, F., Yegildirek, A. and Liu, K.: Multilayer neural-net robot controller with guaranteed tracking performance. IEEE Trans. on Neural Networks. **7** (1996) 388–399
3. Watanabe, K., Tang, J., Nakamura, M., Koga, S. and Fukuda, T.: A Fuzzy-Gaussian Neural Network and Its Application to Mobile Robot Control. IEEE Trans. on Control Systems Technology. **4** (1996) 193–199
4. Zhang, Y. N., Wang, J. and Xu, Y. S.: A Dual Neural Network for Bi-Criteria Kinematic Control of Redundant Manipulators. IEEE Trans. on Robotics and Automation. **18** (2002) 923–931
5. Li, Y. and Liu, Y.: Sliding Mode Adaptive Neural-Network Control for Nonholonomic Mobile Modular Manipulators. The 2nd Int. Conf. on Autonomous Robots and Agents(ICARA04), New Zealand (2004) 95–100
6. de Wit, C. C., Siciliano, B. and Bastin, G.: Theory of Robot Control. Springer-Verlag London Limited (1996)
7. Haykin, S.: Neural Networks: A Comprehensive Foundation. 2nd edition. Prentice-Hall, Inc. (1999)

A Neural Network-Based Camera Calibration Method for Mobile Robot Localization Problems

Anmin Zou, Zengguang Hou, Lejie Zhang, and Min Tan

Laboratory of Complex Systems and Intelligence Science, Institute of Automation,
Chinese Academy of Sciences, Beijing 100080, China
{anmin.zou,zengguang.hou,lejie.zhang,min.tan}@mail.ia.ac.cn

Abstract. To navigate reliably in indoor environments, a mobile robot has to know where it is. The methods for pose (position and orientation) estimation can be roughly divided into two classes: methods for keeping track of the robot's pose and methods for global pose estimation [1]. In this paper, a neural network-based camera calibration method is presented for the global localization of mobile robots with monocular vision. In order to localize and navigate the robot using vision information, the camera has to be first calibrated. We calibrate the camera using the neural network based method, which can simplify the tedious calibration process and does not require specialized knowledge of the 3D geometry and computer vision. The monocular vision is used to initialize and recalibrate the robot's pose, and the extended Kalman filter is adopted to keep track of the mobile robot's pose.

1 Introduction

The problem of accurate localization is fundamental to autonomous mobile robots. Existing techniques for estimating the pose (position and orientation) of a mobile robot can be roughly divided into two classes [1]: methods for keeping track of the robot's pose and methods for global pose estimation. The first class assumes that the initial pose of the robot is known *a priori*, whereas the methods of the second class allow mobile robots to be started even under complete uncertainty with respect to their initial pose. Kalman filter based techniques have been proven to be robust and accurate for keeping track of the robot's pose, however, they are not capable of efficiently solving global localization problems [2]. Though Markov localization [3] and Monte Carlo localization methods [4] have been proposed to solve the global pose estimation problems, the running time of the Kalman filter based localization method is much less than that of Markov localization and Monte Carlo localization methods [5].

Natural and artificial landmarks are distinct features that a robot can recognize using its sensory inputs. Natural landmark approach is more flexible because no explicit artificial landmarks are needed, but it may not function well when landmarks are sparse and usually the environment must be known *a priori*. Although the artificial landmark approach is less flexible, the ability to find landmarks is enhanced and the process of map building is simplified [6].

Monocular vision has been used to detect and recognize the landmarks for many years [7], [8], [9]. In [7], [8], [9], the robot's pose was estimated by a triangulation approach, and the camera was used to measure the bearing of one landmark relative to another. There are some other methods for the monocular vision-based localization such as model-based [10], [11] and learning-based [12], [13], [14]. The model-based methods use the image feature such as the geometric feature to estimate the robot's pose, however, extracting the geometric information of the environment from the camera is both time-consuming and intolerant of noise [13], [14]. The learning-based methods use view images as the representation of poses. When the view image seen in the robot matches one of the memorized view images, the robot estimates its pose. But these require extensive training sets and huge storage space [11].

Neural networks, motivated by how the human brain works, are increasingly being employed in various fields, including signal processing, pattern recognition, medicine, speech production and recognition, and business [15]. In recent years, neural networks have been used to solve the camera calibration problems [16], [17]. In this paper, a neural network-based camera calibration method is presented for the global localization of mobile robots with monocular vision. In [16], the neural network is used to learn the transformation from 3D reference coordinates to 2D image coordinates. In some cases, such as mobile robot localization using vision information, we have to transform the image coordinates to reference coordinates. We use the neural network to implement this transformation in this paper. The monocular vision for estimating the robot's pose is used to measure the relative location between the landmark and the robot, which is different from [7], [8], [9]. An extended Kalman filter (EKF) is adopted to keep track of the mobile robot's pose. This method not only retains the advantage of the Kalman filter based localization method but also can solve the global localization problems.

This paper is organized as follows. The camera is calibrated using a neural network method in Section 2. The localization algorithm on the basis of monocular vision and artificial landmarks is described in Section 3. In Section 4, the method for keeping track of the robot's pose based on EKF is presented, which fuses the data from both the vision and odometry. Experimental results are presented in Section 5 and conclusions are given in Section 6.

2 A Neural Network-Based Camera Calibration Method

In order to localize and navigate the robot using the vision information, the camera has to be first calibrated. The camera is fixed on the top of the robot, and pointing to the floor. The camera calibration is a process that models the relationship between the 2D image coordinates and the 3D reference coordinates [16]. In many cases, such as mobile robot localization using vision information, the overall performance of the machine vision system strongly depends on the accuracy of the camera calibration. Some applications, for example vision-based mobile robot localization, do not need the values of the internal and external parameters. A neural network based camera calibration method is proposed in this Section. The proposed method can simplify the tedious calibration process and does not require specialized knowledge of the 3D geometry and computer vision.

Since the landmarks are placed on the floor (as shown in Fig.1. (a)), the 3D robot coordinates of the floor landmarks can be reduced to 2D, we can calibrate the camera using 2D image coordinates and 2D robot coordinates. Now we have a set of image points whose robot coordinates are known *a priori*. Training data for the neural network are defined as a set of known robot coordinates of the control points and the corresponding image coordinates [17].

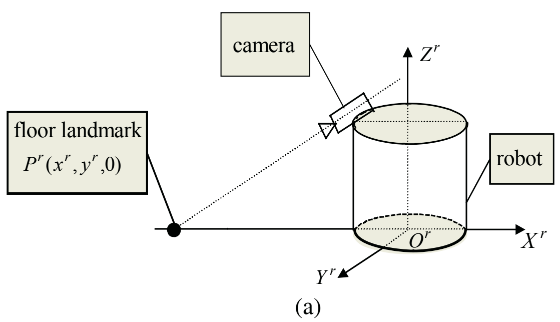


Fig. 1. (a) The robot coordinate system and a floor landmark: $O^rX^rY^rZ^r$ is the robot coordinate system; $P^r(x^r, y^r, 0)$ is a floor landmark in the robot coordinate system; (b) Mobile Robot CASIA-I.

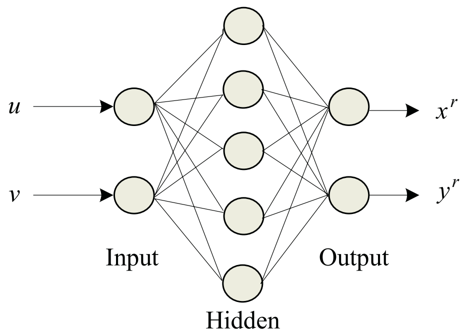


Fig. 2. The structure of the multi-layer perceptron for the camera calibration: with one-input, one-hidden, and one-output layers; $(u, v)^T$ is one point's image coordinate; $(x^r, y^r)^T$ is in the robot coordinate system; input and output layers have two nodes, respectively, and hidden layer has five nodes.

The multi-layer perceptron (as shown in Fig.2) has a role to transform image coordinates into robot coordinates. We use sigmoid function as the activation function between input and hidden layers, and linear function as the activation function between hidden and output layers. Back-propagation learning algorithm for training the multi-layer perceptron is a gradient descent error-correction algorithm that minimizes the errors between the desired outputs and the actual computed outputs by modifying the connection strengths, or weights, between the units in the network [18], [19].

3 Localization Using Monocular Vision

We assume that the robot has the map of the hallway environment (as shown in Fig.3. (a)). When the robot observes the floor landmarks p_a, p_b and p_c and (as shown in Fig.3. (b)), it will obtain their image coordinates $(u_a, v_a)^T$, $(u_b, v_b)^T$ and $(u_c, v_c)^T$, and then can compute their robot coordinates p_a^r, p_b^r and p_c^r using $(u_a, v_a)^T$, $(u_b, v_b)^T$ and $(u_c, v_c)^T$ as the inputs of the multi-layer perceptron, respectively.

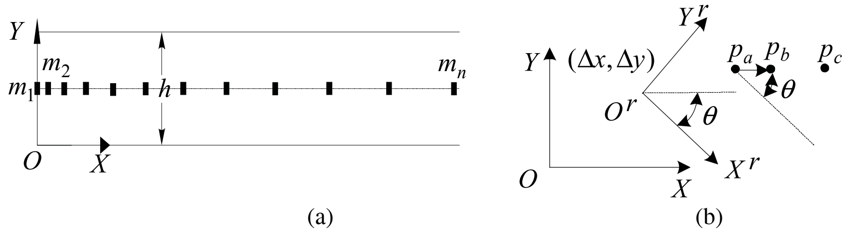


Fig. 3. (a) Map of the hallway environment: *h* is the width of the hallway; *OXY* is the world coordinate system; m_1, m_2, $\cdots m_n$ represent *n* floor landmarks that lie on a line which is parallel to the *OX* axis, and the distance between every two neighboring floor landmarks is different; $m_i = (x_i, y_i)^T$ (i = 1, 2, ⋯, n) is the *i*-th floor landmark in the world coordinate system; (b) Three floor landmarks p_a, p_b, and p_c and in the world coordinate system and the robot coordinate system: *OXY* is the world coordinate system; $O^rX^rY^r$ is the robot coordinate system; $(\Delta x, \Delta y\ \theta)^T$ denotes the pose of the robot in the world coordinate system.

We assume that the landmarks m_i and m_{i+1} ($i \le n-1$) in the map correspond to p_a and p_b, and then we have:

$$\left\|p_a^r - p_b^r\right\| = \left\|m_i - m_{i+1}\right\|. \tag{1}$$

The distance between every two neighboring floor landmarks is different, so we can obtain i by the following equation:

$$i = \arg(\min_j \left\| \left\|p_a^r - p_b^r\right\| - \left\|m_j - m_{j+1}\right\| \right\|),\ j = 1,2,\cdots,n-1. \tag{2}$$

In this case, we have to determine whether p_a corresponds to m_i or m_{i+1}. If $i = 1$, then p_a corresponds to m_1; else if $i = n-1$, then p_a corresponds to m_n; else we can compare $\left\| \left\|p_c^r - p_b^r\right\| - \left\|m_i - m_{i-1}\right\| \right\|$ and $\left\| \left\|p_c^r - p_b^r\right\| - \left\|m_{i+1} - m_{i+2}\right\| \right\|$, and we can obtain the landmarks in the map corresponding to p_a, p_b and p_c, respectively.

Since the floor landmarks in the map lie on a line that is parallel to the OX axis, we have:

$$\overrightarrow{p_a p_b} \,//\, \overrightarrow{OX}. \tag{3}$$

If p_a corresponds to m_i, then we have:

$$\theta = \begin{cases} -\angle(\overrightarrow{p_a p_b}, \overrightarrow{OX_r}), y_b^r - y_a^r \ge 0 \\ \angle(\overrightarrow{p_a p_b}, \overrightarrow{OX_r}), y_b^r - y_a^r < 0 \end{cases}, \tag{4}$$

else:

$$\theta = \begin{cases} -\pi + \angle(\overrightarrow{p_a p_b}, \overrightarrow{OX_r}), y_b^r - y_a^r \ge 0 \\ \pi - \angle(\overrightarrow{p_a p_b}, \overrightarrow{OX_r}), y_b^r - y_a^r < 0 \end{cases}. \tag{5}$$

We assume that the corresponding landmark in the map of p_a is m_i, and then we can obtain Δx and Δy by the following equation:

$$\begin{bmatrix} x_a^r \\ y_a^r \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_i - \Delta x \\ y_i - \Delta y \end{bmatrix} . \tag{6}$$

4 Pose Tracking Algorithm

The sensors, such as the camera, which estimate the relative location between the landmark and the robot, have limited precision and their signals are in general noisy. Therefore the estimated pose of the robot may not accurate. We assume that all uncertainty sources have unimodal Gaussian distributions and provide a model for the robot, the landmark position, and the sensors. The EKF is used to keep track of the robot's pose and fuse the data from both the camera and odometry.

The EKF method has the following form. For each step k the robot's pose and its error covariance are denoted by $X(k) = (x(k), y(k), \theta(k))^T$ and $P(k)$.

4.1 The Motion Model

The motion of the robot in the environment is modeled by:

$$X(k+1) = f(X(k), U(k)) + \omega(k) . \tag{7}$$

where

$$f(X(k), U(k)) = \begin{bmatrix} x(k) + \Delta d(k)\cos(\theta(k)) \\ y(k) + \Delta d(k)\sin(\theta(k)) \\ \theta(k) + \Delta\theta(k) \end{bmatrix} \tag{8}$$

is the state transition function; $U(k) = (\Delta d(k), \Delta\theta(k))^T$ is the control input, by which from k to $k+1$ the robot moves in the direction at k by $\Delta d(k)$ and then rotates around its axis by $\Delta\theta(k)$; $\omega(k)$ is a Gaussian noise $N(0, Q(k))$.

The predicted robot's state is:

$$X(k+1 \mid k) = f(X(k), U(k)) . \tag{9}$$

The covariance matrix associated with the prediction error is computed as:

$$P(k+1 \mid k) = \nabla f P(k \mid k) \nabla f^T + Q(k) . \tag{10}$$

where ∇f is the Jacobian matrix of the transition function f.

4.2 The Measurement Model and Matching Algorithm

The measurement model is:

$$Z_i(k+1) = h(X(k+1)) + v(k) . \tag{11}$$

where

$$h(X(k+1)) = \begin{bmatrix} \cos(\theta(k+1)) & \sin(\theta(k+1)) \\ -\sin(\theta(k+1)) & \cos(\theta(k+1)) \end{bmatrix} \begin{bmatrix} x_i - x(k+1) \\ y_i - y(k+1) \end{bmatrix}. \tag{12}$$

is nonlinear measurement function; $Z_i(k+1)$ is the location of the i-th floor landmark in the robot coordinate system; $v(k)$ is again Gaussian noise $N(0, R(k))$.

The predicted measurement is:

$$\hat{Z}_i(k+1) = h(X(k+1|k)). \tag{13}$$

The innovation term and the associated covariance are computed as:

$$v(k+1) = Z_i(k+1) - \hat{Z}_i(k+1). \tag{14}$$

$$S(k+1) = \nabla h P(k+1|k) \nabla h^T + R(k+1). \tag{15}$$

where ∇h is the Jacobian matrix of the measurement function h.

For each measurement, a validation gate is used to decide whether it is a match or not:

$$v(k+1)S(k+1)v(k+1)^T \le G. \tag{16}$$

If it is true, the current measurement is accepted; else, it is disregarded.

4.3 Updating Algorithm

The state estimate and corresponding state estimate covariance are updated by:

$$X(k+1|k+1) = X(k+1|k) + W(k+1)v(k+1). \tag{17}$$

$$P(k+1|k+1) = P(k+1|k) - W(k+1)S(k+1)W^T(k+1). \tag{18}$$

where $W(k+1)$ is the Kalman gain matrix:

$$W(k+1) = P(k+1|k)\nabla h^T S^{-1}(k+1). \tag{19}$$

5 Experimental Results

In this Section, we will provide some experimental results using our proposed methods. The mobile robot CASIA-I used in our experiment is shown in Fig.1. (b).

5.1 Camera Calibration

In this experiment, we have 60 image points whose robot coordinates are known *a priori*. We use 40 points to calibrate the camera and the remaining 20 points to assess the accuracy of the network. Experimental results are shown in Table 1. The results show that the proposed method using neural network is efficient and is able to offer accurate vision information for mobile robot localization.

Table 1. Calibration errors: x_E^r and y_E^r denote absolute error between the desired output and actual output in x^r-direction and in y^r-direction, respectively

	Max.	Min.	Ave.
x_E^r (cm)	0.2650	0.001	0.0343
y_E^r (cm)	0.2014	0.0001	0.0708

5.2 Localization Experiments

We test the proposed localization algorithm in this way: following a path along the hallway (whose map is shown in Fig.3. (a)). The initial pose of the robot is unknown, and in 5 different locations we use our localization method to estimate the pose of the robot. Experimental results are shown in Table 2. The results show that the proposed localization method is efficient, and we can obtain accurate pose of the robot, which can satisfy the requirement of the robot navigation in indoor environments.

Table 2. Errors in the 5 estimated poses along the hallway

Pose/errors	1	2	3	4	5
X (cm)	1.2	−1.8	2.1	1.6	2.5
Y (cm)	1.4	0.5	−1.3	0.7	1.7
θ(deg)	−2.6	1.4	2.0	−1.3	1.6

6 Conclusions

In this paper, we present a neural network-based camera calibration method for the global localization of mobile robots. The camera is calibrated using a neural network method, which can simplify the tedious calibration process. The experimental results show that the proposed method is efficient and is able to offer accurate vision information for mobile robot localization. The robot's initial pose is estimated on the base of monocular vision and artificial landmarks, and the EKF is used to keep track of the mobile robot's pose. This localization algorithm requires less time and smaller storage space. The experimental results show that the proposed algorithm is efficient and can estimate the robot pose accurately.

Acknowledgements

This research was supported partially by the National Natural Science Foundation of China (Grant Nos. 60205004, 50475179 and 60334020), the National Basic Research Program (973) of China (Grant No. 2002CB312200), the Hi-Tech R&D Program (863) of China (Grant No. 2002AA423160), and the Science and Technology New Star Program of Beijing (Grant No. H020820780130).

References

1. Borenstein, J., Everett, B., Feng, L.: Navigating Mobile Robots: Systems and Techniques. A.K.Peters, LTD., Wellesley, MA (1996)
2. Gutmann, J.-S., Burgard, W., Fox, D., Konolige, K.: An Experimental Comparison of Localization Methods. Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, **2** (1998) 736-743
3. Burgard, W., Derr, A., Fox, D., Cremers, A. B.: Integrating Global Position Estimation and Position Tracking for Mobile Robots: the Dynamic Markov Localization Approach. Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, **2** (1998) 730-735
4. Dellaert, F., Fox, D., Burgard, W., Thrun, S.: Monte Carlo Localization for Mobile Robots. Proceedings of the IEEE International Conference on Robotics and Automation, **2** (1999) 1322-1328
5. Gutmann, J.-S., Fox, D.: An Experimental Comparison of Localization Methods Continued. Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, **1** (2002) 454-459
6. Hu, H. S., Gu, D. B.: Landmark-Based Navigation of Mobile Robot in Manufacturing. Proceedings of the IEEE International Conference on Emerging Technologies and Factory Automation (1999) 114-121
7. Sugihara, K.: Some Location Problems for Robot Navigation Using a Single Camera. Computer Vision, Graphics, and Image Processing, **42** (1988) 112-129
8. Krotkov, E.: Mobile Robot Localization Using a Single Image. Proceedings of the IEEE International Conference on Robotics and Automation, **2** (1989) 978-983
9. Munoz, A. J., Gonzalez, J.: Two-Dimensional Landmark-Based Position Estimation From a Single Image. Proceedings of the IEEE International Conference on Robotics and Automation, **4** (1998) 3709-3714
10. Feddema, J. T., Lee, C. S. G., Mitchell, O. R.: Weighted Selection of Image Features for Resolved Rate Visual Feedback Control. IEEE Transactions on Robotics and Automation,. 7 (1991) 31-47
11. Georgiev, A., Allen, P. K.: Vision for Mobile Robot Localization in Urban Environments. Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and System, **1** (2002) 472-477
12. Meng, M., Kak, A. C.: Fast Vision-Guided Mobile Robot Navigation Using Neural Networks. Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, **1** (1992) 111-116
13. Nayar, S. K., Murase, H., Nene, S. A.: Learning, Positioning, and Tracking Visual Appearance. Proceedings of the IEEE International Conference on Robotics and Automation, **4** (1994) 3237-3244
14. Matsumoto, Y., Inaba, M., Inoue, H.: Visual Navigation Using View-Sequenced Route Representation. Proceedings of the IEEE International Conference on Robotics and Automation, **1** (1996) 83-88
15. Haykin, S.: Neural Networks: A Comprehensive Foundation. 2 nd ED., Upper Saddle River, NJ: Printice Hall (1999)
16. Ahmed, M., Farag, A.: Locked, Unlocked and Semi-locked Network Weights for Four Different Camera Calibration Problems. Proceedings of International Joint Conference on Neural Networks, **4** (2001) 2826-2831
17. Jun, J., Kim, C.: Robust Camera Calibration Using Neural Network. Proceedings of the 1999 IEEE Region 10 Conference on TENCON, **1** (1999) 694-697
18. Lippman, R.: An Introduction to Computing with Neural Nets. IEEE ASSP Magazine, **4** (1987) 4-22
19. Simpson, P. K.: Artificial Neural Systems. Pergamon Press, New York (1990)

Abnormal Movement State Detection and Identification for Mobile Robots Based on Neural Networks*

Zhuohua Duan[1,2], Zixing Cai[1], Xiaobing Zou[1], and Jinxia Yu[1,3]

[1] College of Information Science and Engineering, Central South University, Changsha, Hunan 410083, China
[2] Department of Computer Science, Shaoguan University, Shaoguan, Guangdong 512003, China
duanzhuohua@163.com
[3] Department of Computer Science & Technology, Henan Polytechnic University, Jiaozuo, Henan 454003, China

Abstract. Movement state estimation plays an important role in navigating and movement controlling for wheeled mobile robots (WMRs), especially those in unknown environments such as planetary exploration. When exploring in unknown environments, mobile robot suffers from many kinds of abnormal movement state, such as baffled by an obstacle, slipping, among others. This paper employs neural network method to detect abnormal movement states. Specifically, it exploits the kinematics of the normal and abnormal movement states of the monitored robot. Several residuals are exploited and four probabilistic neural networks are used to classify the residuals. Simulation experiments show that the methods can detect and identify most abnormal movement states.

1 Introduction

Abnormal state detection and identification plays an important role in wheeled mobile robots (WMRs) navigating, dead reckoning and movement controlling, especially in unknown environments such as planetary exploration. Typically, abnormal states of WMRs can be split into two categories: faults of internal components and abnormal movement states when interacting with the environments. Several researchers have studied the first category [1]-[5]. However, the second category of abnormity receives little attentions except the work of [6]. For this reason, this paper focuses on the second category.

Neural networks are widely used for the problem of pattern classification. There are many neural networks available for abnormal movement state detection and identification such as multi layer perceptron, feedforward network, radial basis function network, adaptive resonance theory, self organizing feature map network, among others [7]. The main disadvantage of neural networks is that they require a lot of data to give good confidence in the results. Probabilistic neural networks (PNNs) are parallel implementations of a standard Bayesian classifier that can efficiently perform pattern classification. A primary advantage of the PNN is that it does not require extensive training [8]. For this reason, this paper employs PNN based methods to

* This work is supported by the National Natural Science Foundation of China Grant #60234030 to Zixing Cai.

analyze residuals for abnormal movement state detection and identification of mobile robots. Specifically, the method uses several probabilistic neural networks to classify the residuals into normal and several abnormal movement states including slipping, blocking and dead blocking.

The rest of the paper is organized as follows. Section 2 illustrates the wheeled mobile robot system and exploits its kinematics models of normal and several abnormal movement states. Section 3 details the integrated neural network method, including the structure of the network, feature selection, residual generation and so on. Section 4 illustrates the simulation and experiment result. Section 5 provides the conclusion and offers some future trends.

2 Mobile Robot and Its Kinematics

2.1 The Monitored Mobile Robot

Cai, Zou et al. developed a mobile robot with the ability of roving in complex terrain [9]. The prototype of the robot is shown in fig 1. The robot adopts an adjustable architecture: six-wheel architecture with a good climbing capability and five-wheel architecture with good turning performance. In both architectures, the four front wheels are driving wheels and each front wheel is equipped with a step motor and a wheel encoder. In addition, a fiber gyro is fixed in the platform to detect the yaw rate.

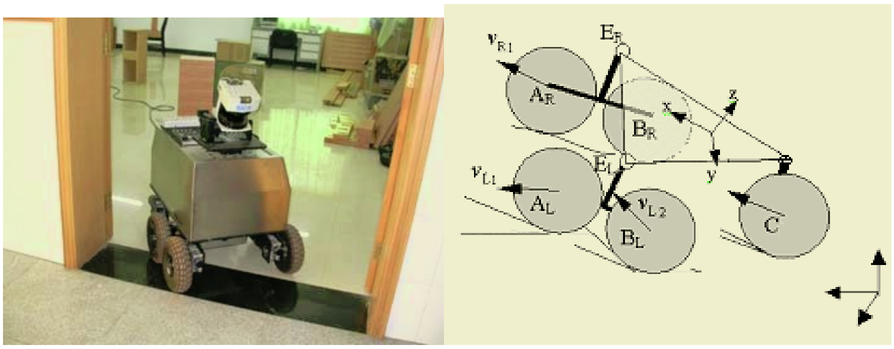


Fig. 1. The monitored mobile robot and its architecture

The robot is equipped with other internal sensors such as rotation angle sensors, inclinometer and external sensors such as vision cameras system and laser range finders. In this paper, we limit our work on the plane movement and only exploit the kinematics models concerning the wheel encoders and the setting speed of step motor.

2.2 The Kinematics of the Robot

Kinematics model of mobile robot is essential for abnormal state detection and identification. The kinematics model of the robot described above moving in plane is shown in equations 1-4.

$$\omega_{L1}=u_{L1}+e_{L1},\ \omega_{R1}=u_{R1}+e_{R1},\ \omega_{L2}=u_{L2}+e_{L2},\ \omega_{R2}=u_{R2}+e_{R2} \tag{1}$$

$$V_{L1}=r_{L1}.\omega_{L1},\ V_{L2}=r_{L2}.\omega_{L2},\ V_L=(V_{L1}+V_{L2})/2 \tag{2}$$

$$V_{R1}=r_{R1}.\omega_{R1},\ V_{R2}=r_{R2}.\omega_{R2},\ V_R=(V_{R1}+V_{R2})/2 \tag{3}$$

$$\omega=(V_R - V_L)/w \tag{4}$$

where u_{L1}, u_{L2}, u_{R1}, u_{R2} denote the setting speed of left front, left rear, right front, right rear driving wheel respectively. ω_{L1}, ω_{L2}, ω_{R1}, ω_{R2} denote measurements of left front, left rear, right front, right rear wheel encoder respectively. V_{L1}, V_{L2}, V_{R1}, V_{R2} denote linear velocity of left front, left rear, right front, right rear driving wheel respectively. r_{L1}, r_{L2}, r_{R1}, r_{R2} denote radii of left front, left rear, right front, right rear driving wheel respectively. VL, VR denote the linear velocity of left side, right side. ω_{L1}, ω_{L2}, ω_{R1}, ω_{R2} are measured with left front, left rear, right front, right rear driving wheel encoder respectively. e_{L1}, e_{L2}, e_{R1}, e_{R2} are white Gassian noise with mean zero, denote measurement noise of left front, left rear, right front, right rear driving wheel encoder respectively. ω denotes the yaw rate and is measured with gyroscope. w denotes the length of the axis.

This paper focuses on abnormal movement state estimating when the robot is moving in plane. In this setting, the two driving wheels of each side are driven with the same setting speed. Let u_L denotes the setting speed of left side, and u_R denotes the setting speed of right side. Hence, $u_{L1}=u_{L2}=u_L$, $u_{R1}=u_{R2}=u_R$.

2.3 Abnormal Movement States and Its Kinematics

Typically, wheels may suffer from abnormity such as slipping, blocking and dead blocking. Let $D=\{S_{L1}, S_{L2}, S_{R1}, S_{R2}, B_{L1}, B_{L2}, B_{R1}, B_{R2}, D_{L1}, D_{L2}, D_{R1}, D_{R2}\}$ denotes the abnormal movement states set, where S_{L1}, S_{L2}, S_{R1}, S_{R2} denote slipping abnormal of left front, left rear, right front, right rear driving wheel respectively. B_{L1}, B_{L2}, B_{R1}, B_{R2}, denote blocking abnormal of left front, left rear, right front, right rear driving wheel respectively. D_{L1}, D_{L2}, D_{R1}, D_{R2}, denote dead blocking abnormal of left front, left rear, right front, right rear driving wheel respectively.

The slipping abnormal is typically modeled as that the wheel encoder measurement is larger than the actual speed. The blocking abnormal is modeled as that the measurement smaller than the set speed. The dead blocking abnormal is modeled as that the measurement is nearly zero while the set speed is non-zero.

3 Abnormal Movement State Detection and Identification Based on Neural Networks

Neural network state estimation method typically consists of two steps, residual generation and residual classification. This paper focus on how to select residuals through exploits the redundancy of the system and how to handle multiple abnormal movement states through multiple neural networks.

3.1 Feature Selection and Residual Generation

Typically, the process of feature (residual) selection depends on the domain knowledge of the monitored system. We adopt 8 features reflecting the movement states of driving wheels, as follows. The residuals are normally zero, and become non-zero as a result of abnormal.

Table 1. Residuals for abnormal detection

Left wheel residual	Right wheel residuals
$R1=\lvert\, \lvert\omega_{L1}\rvert-\lvert\omega_{L2}\rvert \,\rvert$	$R5=\lvert\, \lvert\omega_{R1}\rvert-\lvert\omega_{R2}\rvert \,\rvert$
$R2=\lvert\, \lvert\omega_{L1}+\omega_{L2}\rvert/2-\lvert u_L\rvert \,\rvert/\lvert u_L\rvert$ when $\lvert u_L\rvert>0$	$R6=\lvert\, \lvert\omega_{R1}+\omega_{R2}\rvert/2-\lvert u_R\rvert \,\rvert/\lvert u_R\rvert$ when $\lvert u_R\rvert>0$
$R3=\lvert\, \lvert\omega_{L1}\rvert-\lvert u_L\rvert \,\rvert/\lvert u_L\rvert$ when $\lvert u_L\rvert>0$	$R7=\lvert\, \lvert\omega_{R1}\rvert-\lvert u_R\rvert \,\rvert/\lvert u_R\rvert$ when $\lvert u_R\rvert>0$
$R4=\lvert\, \lvert\omega_{L2}\rvert-\lvert u_L\rvert \,\rvert/\lvert u_L\rvert$ when $\lvert u_L\rvert>0$	$Ral8=\lvert\, \lvert\omega_{R2}\rvert-\lvert u_R\rvert \,\rvert/\lvert u_R\rvert$ when $\lvert u_R\rvert>0$

3.2 The Multiple Neural Networks Structure

To estimate movement state of each driving wheel efficiently, we adopt a multiple neural network structure, as shown below in Fig 2.

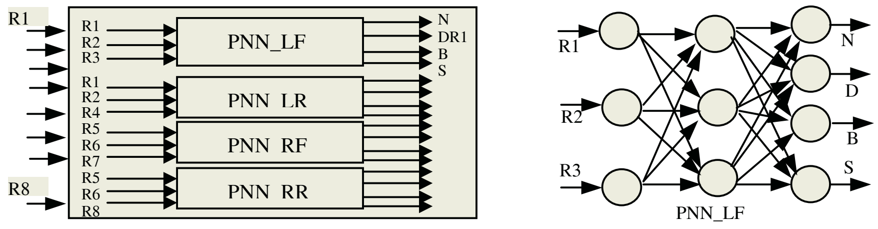


Fig. 2. Multiple neural network architecture and probabilistic neural network for left front wheel(PNN_LF)

Specifically, for each driving wheel, a probabilistic neural network (PNN) is used to classifying the residuals into four classes. Each PNN takes three residuals as inputs and classifies the residuals into 4 classes, namely normal (N), dead blocking (D), blocking (B) and slipping (S). PNN_LF, PNN_LR, PNN_RF, PNN_RR are used for left front, left rear, right front, right rear wheel respectively, and there input residuals are {R1, R2, R3}, {R1, R2, R4}, {R5, R6, R7}, {R5, R6, R8} respectively. PNN_LF is shown in the right side of Fig2, and the others have the same structure. The training process of each PNN is detailed in section 4.

4 Simulation Experiment and Results

Simulation experiment is performed on the mobile robot mentioned above in section 2 in our laboratory. That is, we deliberately block one or two of the driving wheels when the robot is moving forward or backward in the plane.

In detail, we let the robot move backward firstly (from time step 425 to 4093), then let the robot move forward (from time step 4121 to 7186). During these periods, we block the robot intentionally. When a wheel is blocked, its actual speed (given by wheel encoder) is less than the setting speed. Fig 3 illustrates the setting speed of left and right wheel and the measurements of four driving wheels given by wheel encoders.

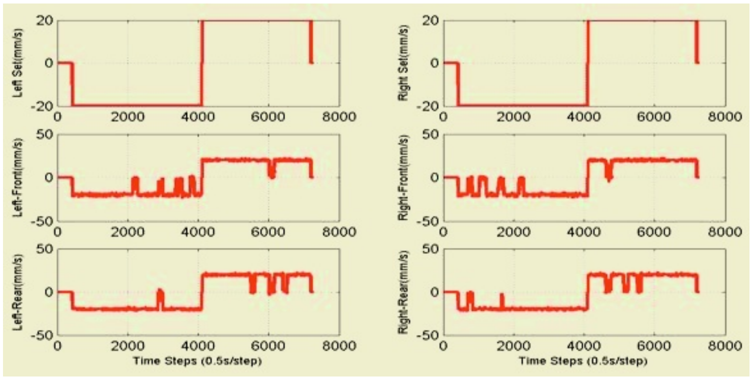


Fig. 3. The setting speed and measurements of four driving wheels with abnormal movement states.

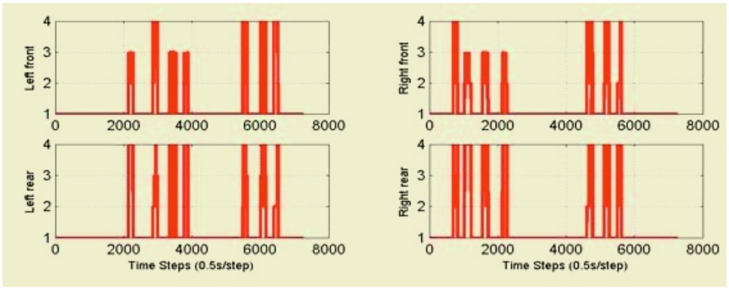


Fig. 4. The movement states of four wheels. (1: Normal, 2: Dead blocking, 3: Blocking, 4: Slipping)

We use samples of backward movement (from time step 425 to 4093) to training the network and use samples of forward movement (from time step 4121 to 7186) to simulate the network. Fig 4 illustrates the training samples and the simulation results. Fig 3 and fig 4 show that most abnormal movement states have been correctly identified.

5 Conclusions

This paper employs four probabilistic neural networks to handle the problem of abnormal movement states identification for a real mobile robot. The kinematics of normal and abnormal (including slipping, blocking, dead blocking) are established, and 8 kinds of residuals are exploited. Based on these, the PNNs classify the residual into 256(=4^4) kinds of movement state. Simulation experiment shows that the method proposed has a good performance.

References

1. Roumeliotis, S.I., Sukhatme, G.S., Bekey, G.A.: Sensor Fault Detection and Identification in a Mobile Robot. IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems (1998) 1383-1388
2. Goel, P., Dedeoglu, G., Roumeliotis, S.I., Sukhatme, G.S.: Fault Detection and Identification in a Mobile Robot Using Multiple Model Estimation And Neural Network. IEEE Int'l Conf. on Robotics & Automation (2000) 2302-2309
3. Washington, R.: On-board Real-time State and Fault Identification for Rovers. IEEE Int'l Conf. on Robotics & Automation (2000) 1175-1181
4. Verma, V., Gordon, G., Simmons, R.: Efficient Monitoring for Planetary Rovers. International Symposium on Artificial Intelligence and Robotics in Space (2003)
5. Kawabata, K., Okina, S., Fujii, T., Asama, H.: A System for Self-diagnosis of an Autonomous Mobile Robot Using an Internal State Sensory System: Fault Detection And Coping with The Internal Condition. Advanced Robotics, **9** (2003) 925-950
6. Dixon, W.E., Walker, I..D., Dawson, D.M.: Fault Detection for Wheeled Mobile Robots with Parametric Uncertainty. IEEE/ASME Int'l Conf. on Advanced Intelligent Mechatronics (2001) 1245-1250
7. Madani, K.: A Survey of Artificial Neural Networks Based Fault Detection and Fault Diagnosis Techniques. Proceedings of International Joint Conference on Neural Networks. Washington, DC, USA (1999) 3442-3446
8. West, B.P., May, S.R., Eastwood, J.E., Rossen, C.: Interactive Seismic Facies Classification Using Textural Attributes and Neural Networks. Leading Edge (Tulsa, OK), **10** (2002) 1042-1049
9. Cai, Z.X., Zou, X.B., Wang, L., Duan, Z.H., Yu, J.X.: A Research on Mobile Robot Navigation Control in Unknown Environment: Objectives, Design and Experiences. Proceedings of Korea-Sino Symposium on Intelligent Systems, Busan, Korea (2004) 57-63

A Neural Network Based Method for Shape Measurement in Steel Plate Forming Robot*

Hua Xu[1], Peifa Jia[1], and Xuegong Zhang[2]

[1]State Key Laboratory of Intelligent Technology and Systems, Tsinghua University, Beijing 100084, China
{Xu-h,dcstjpf}@tsinghua.edu.cn
[2]Department of Automation, Tsinghua University, Beijing 100084, China
zhangxg@tsinghua.edu.cn

Abstract. Shape measurement is one of the critical problems in manufacturing robot systems. The point coordinates that we get change distinctly, because different objects to be processed own various shape forms. So it is difficult for traditional methods to get original accurate shape information. It always affects the processing results of manufacturing robot systems. According to the shipbuilding requirements, this paper proposes a dynamic and intelligent shape measurement method, which is based on the fuzzy reasoning (FR) and neural network (NN) method. FR is used to judge the relation of measured points. As the input of the NN, the fitted coordinate and the possibility of the rim point can be got. It has been demonstrated effective in Dalian Shipbuilding manufacturing robot system.

1 Introduction

In shipbuilding industries, Steel Plate Forming by Line Cooling and Heating (SPFCH) is a critical technology in processing steel plates for shipbuilding requirements [1]. The processing work is conducted by industrial robots called SPFCH robots. In order to process one steel plate accurately, the robot system should firstly measure the steel plate shape correctly. According to its shape, the corresponding processing procedure can be completed correctly. However, the shape of different rolled steel plates are always various. So the shape measurement needs to be conducted in every processing procedure. In order to measure the steel plate shape correctly, laser range finders are always used in detecting rims of steel plates. Then the shape can be fitted according to the measurement results. However, because steel plate forms are different, it is always difficult for laser detection measurement to decide rims according to the measured data directly. The variation of steel plate form is always non-linear. So it needs a new method to decide the steel plate rim.

Artificial Neural Network (ANN) is an information processing structure which imitates human brain behavior [2]. It is a powerful tool for non-linear approximations

* This work was jointly supported by the National Nature Science Foundation for Youth Fund (Grant No: 60405011), the China Postdoctoral Foundation for China Postdoctoral Science Fund (Grant No: 20040350078) and the National 863 Project of China (Grant No: 2001AA421200-1).

and can be used when all other methods have failed [3]. Amongst the numerous ANN architectures available in the literature, the feed-forward networks with one or more hidden layers and the back propagation learning algorithm [4] are the most common in measurement applications[5], [6]. In order to decide the processing track correctly in shipbuilding, a feed-forward ANN has been used in measuring and detecting the steel plate rim. It can overcome the difficulty of anomalous change of plate rims. The measure accuracy can be improved effectively.

In this paper, a shape measurement system (SMS) using ANN is proposed. This paper is organized as the following. In Section 2, the structure of the system is expatiated in details. Section 3 presents our intelligent and dynamic shape measurement tool. Section 4 analyzes the experimental results. Finally, the conclusion and future work can be found in Section 5.

2 Architecture of Shape Measurement System (SMS)

SMS, designed to judge the steel plate rim according to the measured data, has several tasks to resolve as follows:

(1) Measuring the rim points and calculating its space location.
(2) Representing the measured points and the rim standards.
(3) Evaluating the similarity between the measured points and rim standards.
(4) Filtering and classifying process.
(5) Learning process.

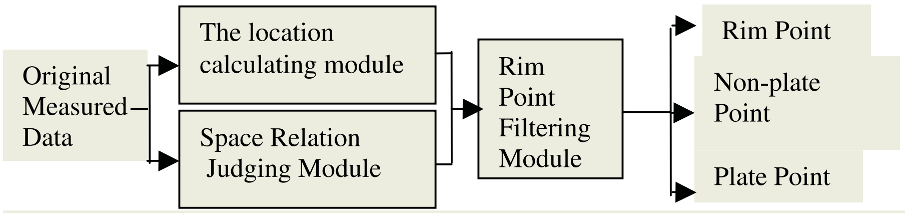

Original Measured Data
The location calculating module
Space Relation Judging Module
Rim Point Filtering Module
Rim Point
Non-plate Point
Plate Point

Fig. 1. To address those above mentioned problems, the architecture of SMS is composed of three modules: (1) the location calculating module; (2) the space relation between the current point and the former points retrieving module based on Fuzzy Reasoning (FR); (3) a rim point filtering module based on BP neural network.

3 Shape Measurement System Based on BP Neural Network

In SMS, original measured data should be input to the location calculating module firstly. Also should the original measured data be input to the space relation judging module. All of these calculating results are as the input to the rim point filtering module.

3.1 The Coordinate Calculating Module

In the approach, a location calculating module is designed to calculate the location coordinate relative to the system origin according to the measured data and the machine parameter. In order to calculate the location coordinate of one measured data given by laser location finder, we collected the measured data, the movement distance on every axis and the relative coordinate. Then the location coordinate of the measured point can be got.

Suppose the machine parameters of the measurement instrument are as the following. The angle coordinate θ is THETA. The coordinate of the laser location finder is (MEA_X, MEA_Y, MEA_Z) relative to θ axis. The coordinate of machining instrument is (PRO_X, PRO_Y, PRO_Z) relative to the origin of θ axis.

Suppose the measured location coordinate is (X_m, Y_m, Z_m). The distance measured by laser location finder is MEASURE. If the machining instrument moves from 0 to θ p on θ axis, then move the measured point. The coordinate of this time is (X_p, Y_p, Z_p). It is the coordinate required by machining instrument.

According to the method of the space coordinate calculation, the coordinate required by machining instrument can be calculated as the following:

$$\theta = THETA - \theta p. \tag{1}$$

$$Xp = Xm - MEA\_X + PRO\_X. \tag{2}$$

$$Yp = Ym - MEA\_Y - PRO\_Y * \cos\theta + PRO\_Z * \sin\theta. \tag{3}$$

$$Zp = Zm + MEASURE + MEA\_Z - PRO\_Y * \sin\theta - PRO\_Z * \cos\theta. \tag{4}$$

3.2 Space Relation Judging Module Based on Fuzzy Reasoning

Space relation judging module is used to analyze the relation between the current measured point and the former one. According to the change on every axis, one analysis result that whether the former point is a rim point or not can be got. However, in different situations, data change on different axis is also various, because the coordinate of angle axis is different. So in the space relation judging module, fuzzy reasoning methodology is used to analyze the relation between the current point and former one.

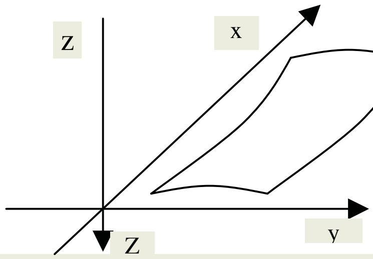


Fig. 2. Steel Plate in Space Coordinate

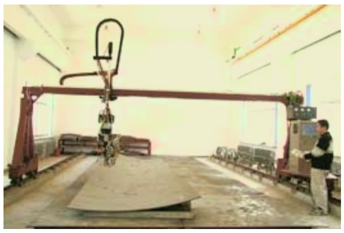


Fig. 3. Manufacturing Robot System

In this module, a simple fuzzy reasoning [8], [9], [10] is used to analyze whether one point is a rim point or not. From Fig.2, it can be known that if one point is a rim point actually, the difference of two neighbor points' Z coordinate in space coordinate changes obviously compared with those of X and Y coordinate. Because of the different curving plate of steel plates, the difference of X or Y coordinate may also change according to the space angle θ. So the fitness of one rim point P_n relates to the difference of its coordinate (X_n, Y_n, Z_n) can be represented as the following. To judge the rim point, the former measured point P_{n-1} and the next one P_{n+1} are required. Suppose their coordinates are (X_{n-1}, Y_{n-1}, Z_{n-1}) and (X_{n+1}, Y_{n+1}, Z_{n+1}).

$$\omega_Z + \omega_Y + \omega_X = 1 . \tag{5}$$

$$\Delta^1_Z = Z_n - Z_{n-1} . \tag{6}$$

$$\Delta^2_Z = Z_{n+1} - Z_n . \tag{7}$$

$$\Delta^1_Y = Y_n - Y_{n-1} . \tag{8}$$

$$\Delta^2_Y = Y_{n+1} - Y_n . \tag{9}$$

$$\Delta^1_X = X_n - X_{n-1} . \tag{10}$$

$$\Delta^2_X = X_{n+1} - X_n . \tag{11}$$

$$f(p) = \frac{\Delta^1_Z \times \omega_Z}{\Delta^2_Z} + \frac{\Delta^1_Y \times \omega_Y}{\Delta^2_Y} + \frac{\Delta^1_X \times \omega_X}{\Delta^2_Z} . \tag{12}$$

To use this approach, the first two measured points need to be preserved to be the points on the steel plate. This requirement can be preserved, because the measurement include plate measurement and rim measurement. Usually, the beginning points are on the plate. From formula (12), if the fitness value of the point P_n is close to 0, then the point P_n is a plate point. If it is close to 0.5, then it is a rim point. Otherwise, if it is near to 1, then the point is a point outside the steel plate. Then the fitness value only describes the relation between the measured point and the former points. In the next, the fitness value is input to the rim point filtering module. And the curve fitted the measured points can be got.

3.3 Rim Point Fitting Module Based on BP Neural Network

The BP network is one of the most widely applied ANN firstly introduced by D.E. Rumelfield and Mcclelland [4]. It has successfully been used in various application fields including signal processing, image recognition, model analysis and system control [11], [12]. Good approach character and excellent generalization are its distinct features. The BP's objective in rim fitting is to fit the rim curve according to the transcendental knowledge. The BP network is composed of multi-neuron layers: input layer, hidden layer and output layer.

In this paper, the inputs of BP network are the outputs of the coordinate calculating module –"point coordinate" and the outputs of the relation analysis module-"rim fitness value". There are three layers in the BP network. The three layers are fully interconnected since the fitness value is connected to all of the neurons in the input layer.

In Fig.4, the BP network topology is depicted. After training, three neurons in output layer represent the axis coordinate respectively and the other one neuron in output layer represent the type of points (plate point, rim point or outside point).

Suppose that the input x = (X, Y, Z, f) = (x_1, x_2, x_3, x_4), the weight of the q th neuron in the i th layer is $\omega_{ij}^{(q)}$. The adaptive step learning algorithm is used. In the learning algorithm, the weights are revised according to the following formula [11], [13]:

$$\omega_{ij}^{(q)}(k+1) = \omega_{ij}^{(q)}(k) + \alpha(k) D_{ij}^{(q)}(k) \cdot \tag{13}$$

$$\alpha(k) = 2^{\lambda} \alpha(k-1) \cdot \tag{14}$$

$$\lambda = \mathrm{sgn}[\, D_{ij}^{(q)}(k) D_{ij}^{(q)}(k-1)] \quad \cdot \tag{15}$$

where α (k) is the learning rate. sgn is the sigmond function. D(k) is the minus grads at the time k. For further details about the learning algorithm please refer to [11]. The output of the neural network includes the fitted coordinate of the point and its fitness. If the fitness is near to 0, then it is the point on the steel plate. However, if the fitness is near to 0.5, it represents the rim point. If the fitness is near to 1, it is the point outside the steel plate.

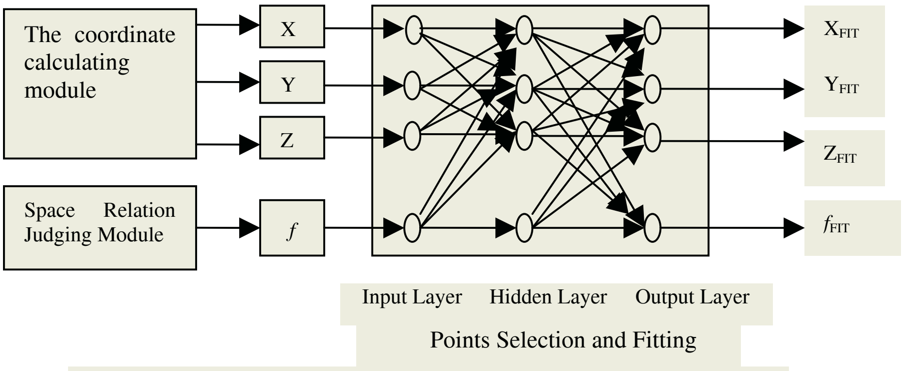

The coordinate calculating module
Space Relation Judging Module
X
Y
Z
f
X_FIT
Y_FIT
Z_FIT
f_FIT
Input Layer
Hidden Layer
Output Layer
Points Selection and Fitting

Fig. 4. The Neural Network in Filtering and Fitting Module

4 Experiments

The proposed scheme was actually applied on the Dalian Shipbuilding Steel Plate Forming Robot platform. Due to space limit, it does not present additional examples in this paper. We have measured 14738 points, including rim points, outside points and plate points. The collection is split into a training set of 4719 points and a testing set 10019 points. The average precision of the shape measurement is 97.99%. And the average precision of fuzzy classification is 92.09%. In comparison with Fuzzy classification, the above-mentioned scheme, based on a combination of the FR and SOFM, will increase the accuracy of shape measurement.

5 Conclusions

Anomalistic rim and curve steel plate are usual phenomena in Shipbuilding. In order to process steel plate accurately, shape measurement is a critical problem to be solved in the manufacturing robot system. This paper presents a fuzzy reasoning and BP neural network based method to complete the shape measurement. In this paper, fuzzy reasoning is firstly used to judge the relationship between the measured points and the points on the steel plate. A fitness value is used to describe this relation. Then, as the inputs, the fitness value and the point coordinate are processed in a BP neural network so as to get the fitted point coordinate according to the relation description. What's more, the possibility of whether the point is on the steel plate can be got. So the steel plate shape can be fitted accurately. The validity of this method has been demonstrated by using it in the manufacturing robot system of Dalian Shipbuilding Factory.

The optimization of the fitting neural network according to the shape measurement requirements will be studied in the future. Via the optimization, it is to realize the aim of improving the efficiency of the shape measurement. On the other, the method is designed for single point measurement. In real manufacturing robot systems, it always needs multi-points measurement and processing. So in the next, synchronous multi-point shape measurement will be studied on the base of this method.

References

1. Jinyu, Z.: Design of the Controller for Sample Machine of SPFCH Intelligent Robot. Master Thesis of Tsinghua University, Tsinghua University Beijing (2004)
2. Simpson, P.K.: Foundation of Neural Networks. IEEE Technology UPDATE SERIES Neural Networks Theory , Technology, and Applications , New York (1996) 1-22
3. Armitage, A.F.: Neural Networks in Measurement and Control. Measurement and Control, **25** (1995) 205-215
4. Rumelhart, D., Mecelelland, J.: Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Cambridge: Bredford Books, MIT Press, Massachusetts (1986)
5. Daponte, P.; Grimaldib, D.: Artificial Neural Networks in Measurements. Measurement, **23** (1998) 93-115
6. Eren, H.; Fung, C.C.; Wang, K.W.: An Application of Artificial Neural Network for Prediction of Densities and Particle Size Distributions in Mineral Processing Industry. Proceedings of IEEE Instrumentation and Measurement Technical Conference , **1** (1997) 1118-1121
7. Kohonen, T. (ed.): Self Organizing Maps (Second Edition). Springer-Verlag, Berlin Heidelberg New York (1997)
8. Klir, G.J., Folger, T.A.: Fuzzy Sets, Uncertainty, and Information**,** Prentice Hall, Upper Saddle River, NJ, USA (1988)
9. McBratney, Alex. B., Odeh, Inakwu O.A.: Application of Fuzzy Sets in Soil Science: Fuzzy Logic. Fuzzy Measurements and Fuzzy Decisions. Geoderma, **77** (1997) 85-113
10. Ramot, D., Milo, R.,Friedman, M., Kandel, A.:On Fuzzy Correlations**.** IEEE Transactions on Systems, Man and Cybernetics (Part B), **31**(2001) 381 - 390
11. Martin T. Hagan, Howard B. Demuth: Mark Beale, Neural Network Design**,** CITIC Publishing House, Beijing (2002)
12. Horvath, G.: Neural Networks from the Perspective of Measurement Systems. Proceedings of the 20th IEEE Instrumentation and Measurement Technology Conference, **2** (2003) 1102-1107
13. Sun, Z., Zaixing, Z.: Intelligent Control. Tsinghua University Press, Beijing (1996)

Recurrent Networks for Integrated Navigation

Jianguo Fu, Yingcai Wang, Jianhua Li, Zhenyu Zheng, and Xingbo Yin

Department of Navigation, Dalian Naval Academy, Dalian, Liaoning 116018 China
Fujianguo80@hotmail.com

Abstract. A novel neural-network-compensated Kalman filter for integrated navigation system was proposed. Based on the similarity of operation principle between Elman networks and non-linear ARMA model, the Elman network is employed as a compensating error estimator to improve accuracy of the Kalman filter. The proposed architecture is evaluated with the acquired data from a naval vessel. And the results show that the presented method can markedly attenuate the effect of interferes to Kalman filter, and improve the precision of the integrated navigation system.

1 Introduction

To provide accurate vehicle state estimation, most present-day vehicular positioning systems depend on Kalman filtering to integrate INS, GPS and other navigation equipment outputs. Although the Kalman filter is widely used and represents one of the best solutions for integrated navigation, it still has some drawbacks related to model error, immunity to noise effects, and observability [1]. Neural networks have been extensively applied in diverse areas [2],[3],[4]. Based on the similarity of operation principle between Elman networks and non-linear ARMA model[5],[6], This article describes a new multisensor integration method that uses neural networks compensating estimation error of the Kalman filter. The compensating value is estimated by fusing actual observation and filter estimation with the Elman networks. The proposed architecture can react on both external and inner interferes and noise of the system. The method is evaluated with recorded data from a naval vessel. The results show that the presented method improves obviously the accuracy of the system.

2 Neural Networks Error Estimator

A time series is an ordered sequence of observations made through time, whereas a time series prediction problem is the prediction of future data making use of past information. It involved the random model structure. At present, a lot of non-linear random model have been proposed. It is well-known that ANNs are universal function approximators and do not require knowledge on the process under consideration. ANNs for modeling time series generally have special structures that store temporal information either explicitly using time-delayed structures or implicitly using feedback structures. We adopt recurrent networks to approximate non-linear ARMA model.

2.1 Similarity Between Elman Network and Non-linear ARMA Model

The most general model which is used in prediction is a linear ARMA(p, q) model[7]

$$X_t = \sum_{i=1}^{p} \varphi_i X_{t-i} + \sum_{j=1}^{q} \theta_j e_{t-j} + e_t \, . \tag{1}$$

where e_t is assumed to be uncorrelated zero-mean sequences with $E(e_t | X_{t-1}, X_{t-2} ......) = 0$. A general extension of the linear ARMA models to non-linear case is:

$$X_t = h(X_{t-1}, X_{t-2}, X_{t-3}......, X_{t-p}, e_{t-1}, ...e_{t-q}) + e_t \, . \tag{2}$$

We call this model the non-linear auto regressive moving average NARMA(p, q) model . With finite observation, we can utilize following recurrent approach approximate condition mean predictor:

$$\begin{aligned} \hat{X}_t &= h(X_{t-1}, X_{t-2}, X_{t-3}......, X_{t-p}, e_{t-1}, ...e_{t-q}) \\ \hat{e}_j &= X_j - \hat{X}_j \end{aligned} \tag{3}$$

The approximate prediction model can be approached approximately by an external recurrent networks[5]. The external recurrent networks and NARMA (*p, q*) model are showed in figure1.

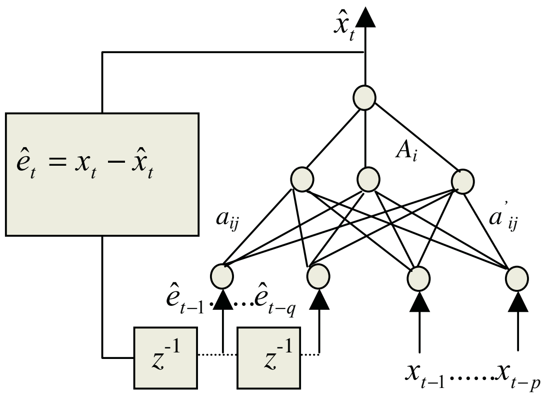


Fig. 1. External recurrent networks and NARMA(*p,q*) model

Where output estimation is:

$$\hat{X}_t = \sum_{i=1}^{H} A_i g[\sum_{j=1}^{p} a_{ij} X_{t-j} + \sum_{j=1}^{q} a'_{ij} (X_{t-j} - \hat{X}_{t-j}) + W_i] \, . \tag{4}$$

The advantage of external recurrent networks is that can directly use output as feedback, but this network must know the orders of p, q, which is very difficult to unknown character time series. Under the certain conditions, it has been proved that recurrent networks with inner delay time, shown by figure1, are the same to the external recurrent networks[6]. A typical inner recurrent networks is Elman networks. In this network, inputs are received from an external source, passed to a hidden layer, and then on to the output layer. The signal from the output layer is passed to an external source, as well as back to a state layer which then acts as an input layer (along with the actual input layer) to the hidden layer on the next pass. The networks include the dynamic characteristic of the system in the structure of the network directly and can directly learn the non-linear dynamic system without knowing *p* or *q*.

2.2 Recurrent Network Structure and Algorithm

The Elman network commonly is a two-layer network with feedback from the recurrent-layer output to the recurrent-layer input. This recurrent connection allows the Elman network to both detect and generate time-varying patterns. An Elman network is shown below in figure2.

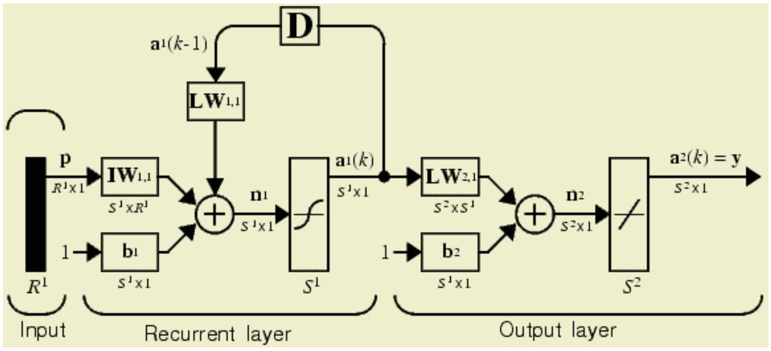


Fig. 2. The structure of inner recurrent network

The additional inner feedback channels of the networks enhance its ability of processing the dynamic information. Its algorithm is similar to the BP networks'. The difference lies in hidden layer. Its import vector $a^1(k)$ is described as:

$$a^1(k) = f(IW_{1.1}p + LW_{1.1}a^1(k-1) + b_1) \,. \tag{5}$$

The output of network is $a^2(k)$:

$$a^2(k) = h(LW_{2.1}a^1(k) + b_2) \,. \tag{6}$$

The Elman network has non-linear neurons in its hidden (recurrent) layer, and linear neurons in its output layer. This combination is special in two-layer networks. And with these transfer functions, it can approximate any function with arbitrary accuracy. The only requirement is that the hidden layer must have enough neurons. More hidden neurons are needed as the function being fit increases in complexity.

Note that the Elman network differs from conventional two-layer networks in that the first layer has a recurrent connection. The delay in this connection stores values from the previous time step, which can be used in the current time step.

3 The Filter of an Integrated Navigation System

Kalman filter is the most widely used estimator in integrated navigation systems. However, It only works well under certain predefined models and statistics characteristics. The ship's practical voyage is affected by the different environment elements. Its process noise and measurement noise are uncertain random process. If the filter is exposed to input data that do not completely fit the assumed property, it will not provide optimal even reliable estimates. In order to eliminate the errors created by external and inner interferes, we use the neural networks as an error estimator to compensate the estimation errors.

3.1 Filter Structure

The structure of the proposed filter is shown in figure 3. NN is the neural networks error estimator, which estimates the error of the Kalman filter by fusing the measurement and state estimation of the kalman filter.

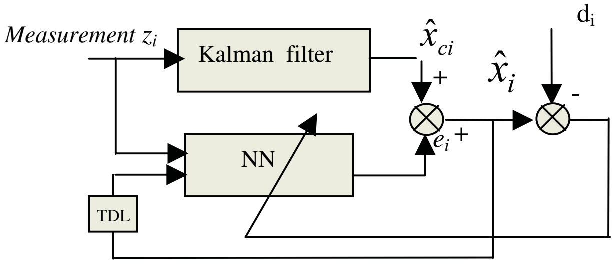


Fig. 3. The filter structure. NN is the neural networks error estimator

3.2 State Equation

The random system is regarded as to satisfy with the situation of the unanimously controlled and unanimously observed linear time invariant system, that is:

$$X_{(k)} = \Phi X_{(K-1)} + \Gamma\, W_{(k-1)}) \quad . \tag{7}$$

$$y_{(k)} = CX_{(k)} + V_{(k)} \quad . \tag{8}$$

Where the state $X_{(k)} \in R^n$, the measurement $y_{(k)} \in R^m$, W (K) and V(K) are the process noise and measurement noise . It is assumed that the noise vectors W(K) and V(K) are zero-mean Gauss white noise vectors and with covariance Q, R respectively. Φ,Γ,C are constant system matrixes. The 6×1 state vector is: X=(φ(Latitude), λ(Longitude), VN(Ocean current northern component), VE (Ocean current eastern component), G(Course), V(Velocity))^T. The 4×1 measurement vector is: Z(k)= (φ(Latitude), λ(Longitude), G(Course), V(Velocity))^T.

The nominal vector of the filter, $\hat{X}_{n(k)}$,is

$$\hat{X}_{n(k)} = \begin{cases} \hat{\varphi}_n(k) = \hat{\varphi}_n(k-1) + TV(K-1)\cos\hat{G}(K-1) \\ \hat{\lambda}_n(k) = \hat{\lambda}_n(k-1) + [TV(K-1)\sin\hat{G}(K-1)]/\cos\varphi(K) \\ \hat{G}_n(k) = \hat{G}(k-1) \\ \hat{V}_n(k) = \hat{V}(k-1) \end{cases} \tag{9}$$

3.3 Error Estimator

Using the neural network as the error estimator is to utilize the neural network's self-learning capability to compensate the Kalman filter's estimation errors. We adopt Elman network , and on the basis of the different measurement amount, uses four neural network to run parallel, and estimate the longitude, latitude, navigation and speed of a ship separately. The input of the network is current measurement and the state estimation at time *k*-1 and *k*-2. The output of the network is the estimation error at time *k*.

4 Experiment Result

The data for evaluating are acquired from a naval vessel. The measurement sensors are: Loran C receiver, DGPS receiver, gyroscope compass, and electromagnetic log. Using the measurement of DGPS as the real value, the recorded data are post-processed with MATLAB. The results are show in figure 4. Figure 4a(left) is the error curve of neural network output. Figure 4b(right) shows the curves of real position of the ship, the position estimation of Kalman filter and the position estimation with neural network error compensation. The figure 4 shows that, Owing to the error compensation with the neural networks, the accuracy is improved obviously.

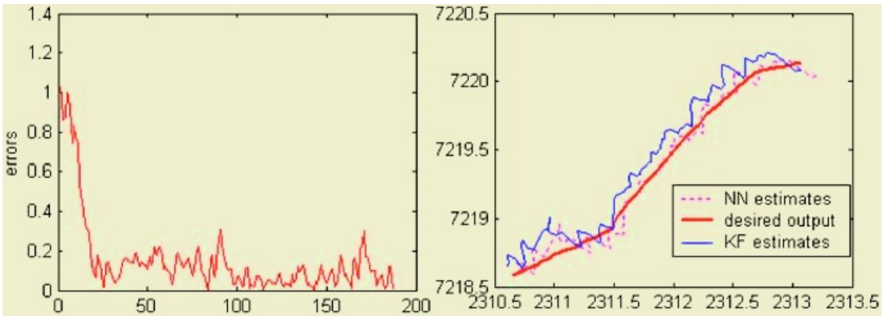


Fig. 4. Experiment results. Left is the compensating error curve , right is the position curve.

5 Conclusions

Recurrent networks have enhanced ability for dynamic information processing via the feedback channel of the hidden layer, which makes it having superior performance in predicting. Utilizing recurrent network to compensate the estimation errors of Kalman filter is an efficient method to improve the performance of Kalman filter. Experimental results show that the effect of inner and external interferes to Kalman filter can be markedly attenuated with the recurrent networks error compensator.

References

1. Haykin, S.: Kalman Filtering and Neural Networks. John Wiley & Sons (2001)
2. Haykin, S.: Neural Networks. 2nd Edition, Prentice Hall (1999)
3. Yang, M.H., Kriegman, D.J., and Ahuja, N.: Detecting Faces in Images: A Survey. IEEE Trans. on Pattern Analysis and Machine Intelligence, **24** (2002) 34-58
4. Igel, C., and Kreutz, M.: Operator Adaptation in Evolutionary Computation and Its Application to Structure Optimization of Neural Networks. Neurocomputing, **55** (2003) 347-361
5. Elman, J.L.: Finding Structure in Time. Cognitive Science, **14** (1990) 179-211
6. Connor, J. T., Martin, D., Atlas, L E.: Recurrent Neural Networks and Robust Time Series Prediction. IEEE Transactions on Neural networks , **5** (1994) 240-254
7. Gilde, C.: Time Series Analysis and Prediction Using Recurrent Gated Experts. Master's Thesis, Department Of Computer Science, University of Skövde, Sweden (1996)

Application of Different Basis and Neural Network Turbo Decoding Algorithm in Multicarrier Modulation System over Time-Variant Channels*

Yupeng Jia[1], Dongfeng Yuan[1,2], Haixia Zhang[1], and Xinying Gao[1]

[1] School of Information Science and Engineering, Shandong Univ., Jinan, Shandong 250100, China
[2] State Key Lab, on Mobile Communications, Southeast Univ., Nanjing, Jiangsu 210096, China
yupengjia@sdu.edu.cn

Abstract. In this paper filter bank generated by different orthogonal Daubechies wavelets and bi-orthogonal wavelets are tested. Due to the high spectral containment of wavelet filters, the wavelet packet Multicarrier moduclation (WP-MCM) system performs better than Fourier based MCM system over time-variant channels. Turbo code adopting neural network (NN) decoding algorithm is applied in WP-MCM system, the BER performance improves greatly over time-variant channels with few iterations in the regions of low signal to noise ratio and is very close to that adopting the maximum a posteriori (MAP) decoding algorithm.

1 Introduction

One contemporary implementation of Multicarrier modulation is the Orthogonal Frequency Division Multiplexing (OFDM) scheme, in which the transmission bandwidth is divided into many narrow subchannels, which are transmitted in parallel. Therefore, the symbol duration is increased and the intersymbol interference (ISI) caused by time-dispersive fading environment is mitigated. However, with longer symbol duration, the interchannel interference (ICI) caused by doppler spread in mobile wireless channels will be increased. As indicated in [1], the ICI degrades the performance of OFDM systems.

In [2], B.G. Negash introduces orthogonal wavelet basis to replace Fourier basis in MCM system. Due to very high spectral containment properties of wavelet filters [3], WP-MCM scheme can better combat narrowband interference and is inherently more robust to ICI than Fourier filters. The classic notion of cyclic prefix (CP) does not apply for WP-OFDM, hence data rates can surpass those of FFT implementations [4]. Another attractiveness is the flexible time-frequency tiling by arranging filter bank constructions, accordingly flexible channel structure is easy to obtain.

We adopt turbo code in WP-MCM to improve system performance and a multilayer perceptron neural network, trained with error back propagation algorithm, is used for

* Supported by National Scientific Foundation of China (No. 60372030).
Supported by China Education Ministry Foundation for Visiting Scholar (No. [2003]406).
Supported by Key Project of Provincial Scientific Foundation of Shandong (No. Z2003G02).

the decoding of turbo code. The motivation of our adopting NN decoding algorithm lies in its ability to learn and approximate arbitrarily complex nonlinear functions. In this case the decoding map has to be learned, thus bypassing the calculation and comparison of metric paths of trellises. NN has further advantages such as the regular and parallelizable nature of the NN architecture, its decoding speed, since after the off-line training phase it is simply a feed forward filter. Here a suitable training set scheme [5] suitable to achieve performance close to MAP is used.

The remainder of this paper is organized as follow: Section 2 gives theoretical analysis of fourier and WP-MCM system. Section 3 describes the turbo coded WP-MCM and NN turbo decoding algorithm. Simulation results over time-variant channels are presented in Section 4, then conclusions follow in Section 5.

2 System Analysis

Normally the fourier transform is used in OFDM scheme. Data is transmitted via the IFFT operation

$$y_F(t) = \sum_{k=0}^{N-1} d_k f_k(t) \tag{1}$$

Where $f_k(t)'s$ are complex exponentials used in the IFFT operation, and $d_k's$ are the data projected on each subcarrier. At the receiver the FFT operation is performed, to determine the data in subchannel j, we match with the waveform for carrier j

$$\langle y_F(t), f_j(t)\rangle = \sum_{k=0}^{N-1} d_k \langle f_k(t), f_j(t)\rangle \tag{2}$$

Where $\langle f_i(t), f_i(t)\rangle = 1$ and $\langle f_i(t), f_j(t)\rangle = 0$.

The channel model we used is that of a Finite Impulse Response (FIR) filter with time-variant tap value. Here we adopt the simplest environment whose impulse response consists of only one fading path. The received signal can be written as

$$r_F(t) = \sum_{k=0}^{N-1} d_k \tilde{f}_k(t) + n(t) \tag{3}$$

Where $\tilde{f}_k(t) = f_k(t) \otimes h(t)$ and $n(t)$ is additive white Gaussian noise.

The channel is of course not ideal. CP can be inserted to overcome ISI. When however this CP is chosen too short, performance degrades since orthogonality is not restored. However predicting the channel impulse response length is a tedious task, and moreover, this length can be so big that the performance loss due to the insertion of a long CP becomes unacceptable. The wavelet filters is just what we need and effective solution to replace the fourier filters. Its longer basis function allows more flexibility in the design of the waveforms, and can offer a higher degree of sidelobe suppression. This result can be explained by examining the filtering operation.

In the fourier scheme, to match for the data projected on carrier j, the receiver must match filter with the waveform for carrier j.

$$\langle r_F(t), f_j(t)\rangle = d_I \rho_{j,j}(0) + \sum_{k=0, k\neq j}^{N-1} d_k \rho_{k,j}(0) + \tilde{n}(t) \tag{4}$$

The effective noise term, $\tilde{n}(t)$ is uncorrelated Gaussian noise. Due to channel effects, the distorted filters are no longer orthogonal to one another and a correlation terms, $\rho_{k,j}(0)$, appear due to the matched filtering operation.

For the WP-MCM system, a similar situation arises, the overlapped nature of wavelet filters requires g symbol periods, for a genus g system, to decode one data vector. we have contributions from the wavelet transforms of g-1 other data vectors

$$y_w(t) = \sum_{k=0}^{N} d_k w_k(t) + \sum_{l=1}^{g-1}\sum_{k=0}^{K-1} d_{k,l} w_k(t - lN) \tag{5}$$

To get data in subchannel j, we match the transmitted waveform with carrier j

$$\left\langle y_w(t), w_j(t)\right\rangle = \sum_{k=0}^{N-1} d_k \left\langle w_k(t), w_j(t)\right\rangle + \sum_{l=1}^{g-1}\sum_{k=0}^{N-1} d_{k,l} \left\langle w_k(t - lN), w_k(t)\right\rangle \tag{6}$$

Where $\left\langle w_i(t), w_i(t)\right\rangle = 1$ and $\left\langle w_i(t-aT), w_j(t-bT)\right\rangle = 0$. Considering equation (3) when matching for the data on carrier j we have

$$\begin{aligned}
&\left\langle r_w(t), w_j(t)\right\rangle \\
&= \sum_{k=0}^{N-1} d_k \left\langle \tilde{w}(t), w_j(t)\right\rangle + \sum_{l=1}^{g-1}\sum_{k=0}^{N-1} d_{k,l} \left\langle \tilde{w}_k(t - lN), w_j(t)\right\rangle + \left\langle n(t), w_j(t)\right\rangle \\
&= d_l \rho_{j,j}(0) + \sum_{k=0, k\neq j}^{N-1} d_k \rho_{k,j}(0) + \sum_{l=1}^{g-1}\sum_{k=0}^{N-1} d_{k,l} \rho_{k,j}(l) + \tilde{n}(t)
\end{aligned} \tag{7}$$

The second term of equation (4) and (7) is a binomially distributed interference term, which is resulted from the loss of orthogonality between subcarriers. Due to the superior spectral containment of wavelet filters, we find this term is much smaller for wavelet scheme. However, as a consequence of the overlapped nature, in the time domain, of wavelet scheme, there are more interference terms compared to Fourier scheme. In the Fourier case, the output terms in equation (4) include the data corresponding to that subchannel, Gaussian noise term and N-1 interference terms. In the wavelet case, the number of interference terms rises to N-1+N(g-1), significantly more than the Fourier case. While the ICI terms for Fourier filters are much larger than their wavelet counterparts, the sheer number of interference terms (both ISI and ICI) offset those gains, when viewed in this light we have found a great deal of channel dependence. In this paper we will prove the efficiency of WP-MCM to overcome ICI and better BER performance than Fourier based MCM over time-variant channels.

3 Turbo Coded WP-MCM System and Neural Network Turbo Decoding Algorithm

Turbo code was first proposed by C. Berrou etc.[6] in 1993. The two fundamental concepts on which they are based are concatenated coding and iterative decoding. The turbo coded WP-MCM system model is given in Fig. 1.

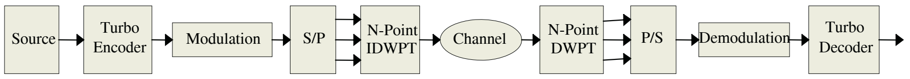


Fig. 1. Turbo coded WP-MCM system model

Turbo encoding is virtually parallel concatenated recursive systematic convolutional (RSC) encoding. While at the receiver, the demodulated bits from received WP-MCM signals are decoded with turbo decoder, which normally adopt either MAP, SOVA or

Log-MAP algorithm, etc. The MAP algorithm consists in finding the most probable bit knowing the received noisy sequence by computing the a posteriori probabilities from the noisy received sample.

Parallel computing capabilities of NN makes them suitable for decoding. In [5] a multilayer perceptron has been trained using the standard back error propagation (EPB) algorithm as shown in Fig.2.

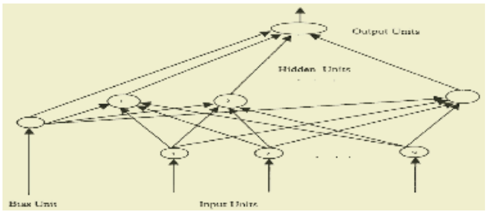


Fig. 2. Multilayer Perceptron Architecture

The pattern set consists of a generated codeword C_k obtained by using a moving time window about a reference bit of the code word as shown in Fig.3. The generated code word C_k is given by

$$C_K = [B_{k-w/2},\dots, B_k \dots, B_{k+2/2-1}] \tag{8}$$

The NN processes the received code word R_k (Fig. 4) sequentially, obtaining the information bit b_k associated with the code word B_k. Then, the time window is shifted by one bit position, including a new code word $B_{k+w/2}$ and discarding the code word $B_{k-w/2}$ and then the next code word C_{k+1} is processed.

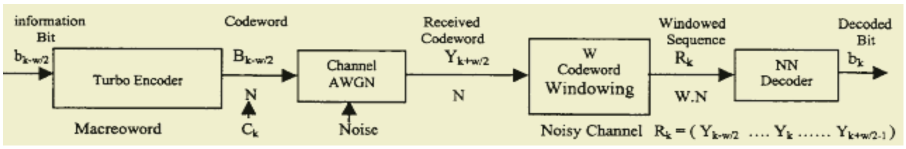


Fig. 3. Block scheme of the communication system with a NN decoder

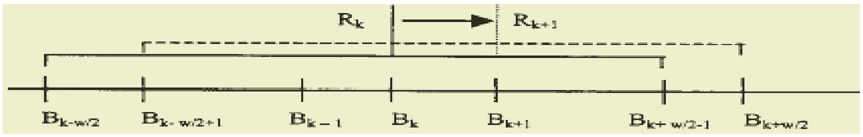


Fig. 4. Structure of received codeword

4 Simulation Results

We can get from Fig.5 that the BER performance of WP-MCM is obviously better than fourier based MCM. As QMF length increasing with 2N, BER performance of dbN case is getting better, which indicates that the improvement of reduced ICI exceeds the degradation of increased ISI over time-variant channels. For biorNr.Nd WP-MCM systems, if the frequency of decomposition wavelet is close to the reconstruction wavelet, namely more similar the wavelet pair are, more robust the system is to channel distortion and the performance loss is much less, such as bior6.8. If filter length of dbN and biorNr.Nd are the same, as shown in Fig.5, BER performance of db6 is better than bior1.5. It's reasonable because of non-orthogonality characteristic of low pass and high pass filters generated by bi-orthogonal wavelet.

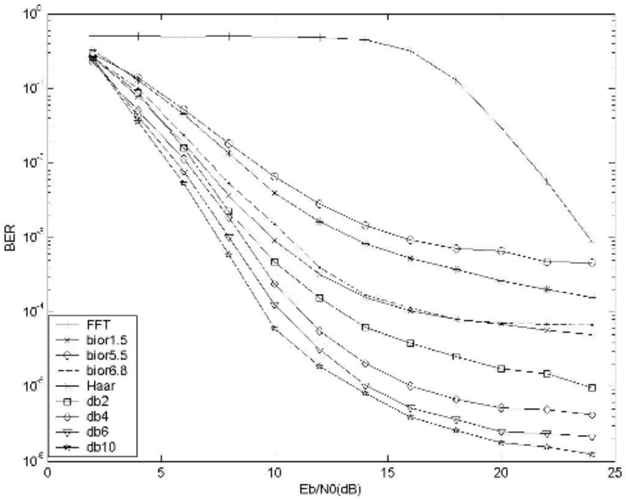


Fig. 5. BER performance of fourier based and some different wavelet packet MCM systems

Performance curves for neural network decoding algorithm and MAP decoding algorithm applied in turbo coded WP-MCM system were obtained in Fig.6, the NN decoding algorithm performs well in low SNR region and achieved comparable performance to that using MAP decoding algorithm over time-variant channels.

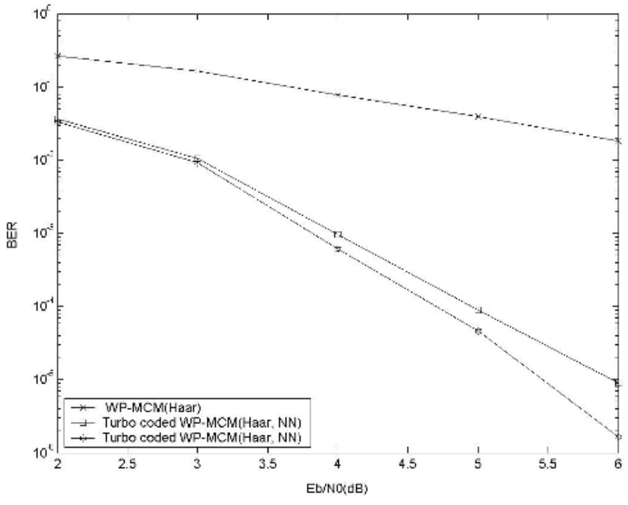


Fig. 6. BER performance for turbo coded WP-MCM (Haar) system

5 Conclusions

After analysis above, some meaningful conclusions can be drawn as following:

1. The BER performance of WP-MCM is better than fourier scheme over time-variant channels. For practical application of orthogonal wavelet basis, a tradeoff should be made between BER performance and computational complexity. The performance of bi-orthogonal WP-MCM deteriorates for the nature of non-orthogonality.
2. Simulation results reveal that the performance of NN decoding algorithm is slightly inferior to the MAP algorithm, however it has the inherent advantages of being fast and less complexity than other turbo decoding schemes.

References

1. Russel, M. and Stuber, G. L.: Interchannel interference Analysis of OFDM in A Mobile Environment, in Proc. VTC (1995) 820-824
2. Negash, B.G. and Nikookar, H.: Wavelet Based OFDM for Wireless Channels, in VTC 2001 Spring. IEEE VTS 53rd , **1** (2001)
3. Daubechies, i.: Ten Lectures on Wavelets, Society for Industrial Applied Mathematics.
4. Miller and Nasserbat,: RM Wavelet Based (WOFDM) PHY Proposal for 802.16.3, Rev.0.0 http://ieee802.org/16
5. Annauth, R., Rughooputh, H.C.S.: Neural Network Decoding of Turbo Codes, IJCNN '99. International Joint Conference on Neural Networks (1999)
6. Berrou, C., Glavieux, A., Thitimajshima, P.: Near Shannon Limit Error-correcting Coding and Decoding. Turbo Codes, in Proc. Int. Conf. Communications (1993) 1064-1070

Blind Detection of Orthogonal Space-Time Block Coding Based on ICA Schemes*

Ju Liu[1,2], Bo Gu[1], Hongji Xu[1], and Jianping Qiao[1]

[1] School of Information Science and Engineering, Shandong University,
Jinan 250100, China
{juliu,gubo,hongjixu,jpqiao}@sdu.edu.cn
[2] State Key Lab. of Integrated Services Networks, Xidian University
Xi'an 710071, China

Abstract. Space-time block coding (STBC) can achieve transmit diversity gain and obtain higher coding gain without sacrifice of bandwidth. But the decoding requires accurate channel state information (CSI), which strongly determines the system performance. Independent component analysis (ICA) technique can be used to detect the transmitted signals without channel estimation. In this paper, we study two schemes based on ICA blind detection by exploiting the orthogonality of orthogonal space-time block coding (OSTBC). What is more, some blind algorithms based on channel estimation are used for performance evaluation. Simulation results for Rayleigh fading channels demonstrate that the two proposed schemes achieve significant bit error rate (BER) performance and indicate the optimal separation algorithms suitable for OSTBC system.

1 Introduction

The next generation wireless communication systems are required to have better quality and coverage, be more power and bandwidth efficient, and be deployed in diverse environments. Multiple antennas can be used at the transmitter and the receiver to enhance the spectral efficiency and system capacity. Orthogonal Space-Time Block Coding (OSTBC), proposed by Alamouti and extended by Tarokh to higher dimension, is a well-known transmit diversity scheme [1], [2]. It can provide maximal diversity order without bandwidth expansion.

For conventional Space-Time Block Coding (STBC), the performance strongly depends on the channel estimation, so powerful detection algorithms have been developed. But if the system is in a rapidly changing environment or there is an inherent estimation error caused by the estimation methods, the system performance will be degraded seriously. In most cases, the channel estimation is implemented by including a long pilot sequence in the data stream. However, it will lead to the loss of spectral efficiency and the decrease of payload rate that can be transmitted.

In recent years, independent component analysis (ICA), one kind of blind source separation (BSS) techniques, has attracted special attentions in the fields of wireless communication. Even without any information of the transmission channel, ICA can recover the transmitted symbols only from the observations.

* This work was supported by National Natural Science Foundation of China (No. 60372031) and Open Foundation of State Key Lab. of Integrated Services Networks.

In this paper, two methods combining OSTBC with ICA based blind detection are proposed. The orthogonality of OSTBC is used to design schemes, which will ensure the orthogonality of the separating matrix. What is more, some other blind detection algorithms are used for performance evaluation.

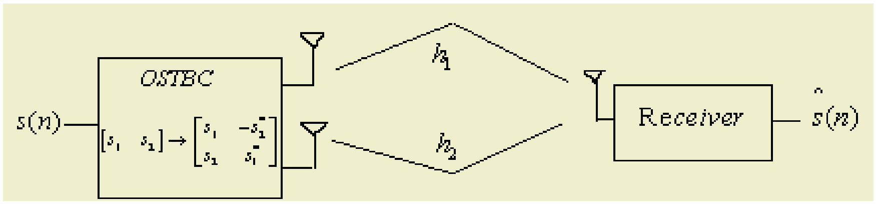

OSTBC
s(n)
$[s_1 \;\; s_2] \rightarrow \begin{bmatrix} s_1 & -s_2^* \\ s_2 & s_1^* \end{bmatrix}$
h_1
h_2
Receiver
$\hat{s}(n)$

Fig. 1. OSTBC system model

2 OSTBC System Model

Considering the simplest OSTBC system with two transmit antennas and a single receive antenna (see Fig.1.), input symbols are grouped in twos, and transmitted from two antennas over two consecutive symbol periods. Let $\boldsymbol{s}(k)=[s_1(k) \quad s_2(k)]^T$ denote the kth block of two symbols to be transmitted, which is encoded by the OSTBC encoder and mapped onto a 2x2 code matrix $\boldsymbol{C}(k)=[\boldsymbol{c}_1(k) \quad \boldsymbol{c}_2(k)]$:

$$\boldsymbol{C}(k)=\sum_{n=1}^{N}\boldsymbol{A}_n s_n(k)=\begin{bmatrix} s_1(k) & -s_2^*(k) \\ s_2(k) & s_1^*(k) \end{bmatrix} \tag{1}$$

where $\boldsymbol{A}_n$ is the orthogonal coding matrix corresponding to the nth symbol $s_n(k)$, and $\boldsymbol{c}_n(k)$ is the 2x1 coded vector at the nth time instant of the kth block. Assume that channel $\boldsymbol{h}=[h_1 \quad h_2]$ is constant over consecutive symbol periods, and the received signal $y_n(k)$ can be expressed as

$$[y_1(k) \quad y_2(k)]=[h_1 \quad h_2]\begin{bmatrix} s_1(k) & -s_2^*(k) \\ s_2(k) & s_1^*(k) \end{bmatrix}+[n_1(k) \quad n_2(k)] \tag{2}$$

where $n(k)$ is Gaussian noise with zero mean and variance σ^2.

OSTBC is a linear STBC that has the following unitary property:

$$\boldsymbol{C}(k)\boldsymbol{C}(k)^H=\sum_{n=1}^{N}|s_n|^2\cdot\boldsymbol{I} \tag{3}$$

So, (2) can be written as

$$\begin{bmatrix} y_1(k) \\ y_2^*(k) \end{bmatrix}=\begin{bmatrix} h_1 & h_2 \\ h_2^* & -h_1^* \end{bmatrix}\begin{bmatrix} s_1(k) \\ s_2(k) \end{bmatrix}+\begin{bmatrix} n_1(k) \\ n_2^*(k) \end{bmatrix} \tag{4}$$

name the matrix $\begin{bmatrix} h_1 & h_2 \\ h_2^* & -h_1^* \end{bmatrix}$ as the equivalent channel coded matrix, denoted by $\boldsymbol{H}$. since $\boldsymbol{H}^H\boldsymbol{H}=\rho\boldsymbol{I}$ where $\rho=|h_1|^2+|h_2|^2$, we know that $\boldsymbol{H}$ is orthogonal, which is very useful in the following designing of detecting schemes.

For the detection of the conventional OSTBC, we should first estimate the CSI, and the existing algorithms can be classified into three categories: trained, blind, and semi-blind detection [3]. Most of them need long pilot sequence to estimate the channel, which would cause a loss of spectral efficiency while spectral resource is very tensional nowadays.

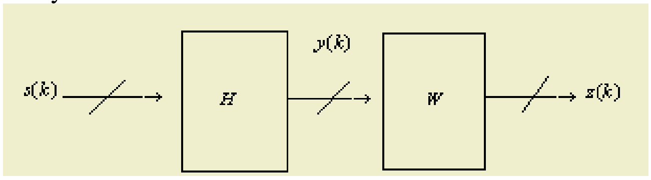


Fig. 2. ICA mathematical model (linear mixture and blind separation)

3 Blind Detection Schemes Based on ICA

3.1 ICA System Model

ICA is a signal processing and data analysis method within the family of BSS techniques, which can recover the transmitted signals only from the observations according to the stochastic independency property of the input signals.

In ICA problems, the linear mixing model (Fig.2) is

$$\boldsymbol{y}(k) = \boldsymbol{H}(k)\boldsymbol{s}(k) + \boldsymbol{n}(k) \tag{5}$$

where $\boldsymbol{s}(k)$ is independent source component, $\boldsymbol{y}(k)$ is the received linear mixing signal, and $\boldsymbol{H}(k)$ is the linear mixing matrix.

The source signal $\boldsymbol{s}(k)$ is recovered by using the separating matrix $\boldsymbol{W}$

$$\boldsymbol{z}(k) = \boldsymbol{W}^T \boldsymbol{y}(k) = \hat{\boldsymbol{s}}(k) \tag{6}$$

Compared with the separating matrix in conventional ICA methods, the separating matrix $\boldsymbol{W}$ used in the OSTBC system is orthogonal, we can give a simple proof: (Here, the case without noise is considered)

$$\boldsymbol{z} = \boldsymbol{W}^T \boldsymbol{y} = \boldsymbol{W}^T \boldsymbol{H}\boldsymbol{s} = \boldsymbol{G}^T \boldsymbol{s} \tag{7}$$

If we want to regard the separated signal $\boldsymbol{z}$ as the source signal $\boldsymbol{s}$, the matrix $\boldsymbol{G}$ should subject to:

$$\boldsymbol{G}^H \boldsymbol{G} = \boldsymbol{W}^H \boldsymbol{H}^* \boldsymbol{H}^T \boldsymbol{W} = \rho \boldsymbol{W}^H \boldsymbol{W} = \boldsymbol{I} \tag{8}$$

Then constraint becomes:

$$\boldsymbol{W}^H \boldsymbol{W} = \frac{1}{\rho} \boldsymbol{I} \tag{9}$$

3.2 Two ICA Detecting Schemes for Recovering Source Signals

There have been several detecting schemes based on ICA [4],[5], but all of them use the conventional ICA algorithms to obtain the separating matrix $\boldsymbol{W}$. Since the conventional ones didn't have the universality of orthogonality for $\boldsymbol{W}$**,** when used in decoding of OSTBC, they also ignore the special orthogonality of $\boldsymbol{W}$, which is very useful information in blind signal recovery. In this part, two schemes using this property are established to estimate $\boldsymbol{W}$.

3.2.1 The first is the improved EASI algorithm, named orthogonal EASI detection. For conventional EASI algorithm [6], the training formulation of the separating matrix is as follows:

$$\boldsymbol{W}(k+1)=\boldsymbol{W}(k)-\mu F(\boldsymbol{z}(k))\boldsymbol{W}(k) \tag{10}$$

where $F(\boldsymbol{z})=\boldsymbol{z}\boldsymbol{z}^T-\boldsymbol{I}+g(\boldsymbol{z})\boldsymbol{z}^T-\boldsymbol{z}g^T(\boldsymbol{z})$ is a function of the nonlinear function $g(\boldsymbol{z})$, and $g(\boldsymbol{z})=\boldsymbol{z}^3$.

In order to use the orthogonality of $\boldsymbol{W}$**,** we adjust the training formulation in (10) and adding one orthogonal term to it, then the training formulation becomes:

$$\boldsymbol{W}(k+1)=\boldsymbol{W}(k)-\mu F(\boldsymbol{z}(k))\boldsymbol{W}(k)+\gamma\boldsymbol{W}(k)(\boldsymbol{I}-\rho\boldsymbol{W}^T(k)\boldsymbol{W}(k)) \tag{11}$$

where μ is the step for training, γ is another gain parameter ranging from 0.5 to 1, and ρ could be estimated by :

$$\hat{\rho}(k+1)=(1-\lambda)\hat{\rho}(k)+\lambda\frac{\boldsymbol{y}^T(k)\boldsymbol{y}(k)}{\sigma_s^2} \tag{12}$$

where λ is a forgetting factor.

In formula (11), as $\boldsymbol{W}$ is convergent, the last term should be equal to zero, which means the separating matrix is orthogonal already. If not, then we can say that the $\boldsymbol{W}$ got here is not optimal.

3.2.2 The second method is proposed from matrix structure designing perspective, named orthogonal J matrix detection. In order to achieve the orthogonality, the scheme (under 2x1 system) is designed as follows:

$$\boldsymbol{W}=[\boldsymbol{w}_1,\boldsymbol{w}_2] \qquad \text{where } \boldsymbol{w}_2=\boldsymbol{J}\boldsymbol{w}_1^* \qquad \text{The matrix } \boldsymbol{J}=\begin{bmatrix}0 & 1\\ -1 & 0\end{bmatrix},$$

$$\text{so that,} \quad \boldsymbol{W}^H\boldsymbol{W}=\alpha\boldsymbol{I} \quad \text{where} \quad \alpha=\|\boldsymbol{w}_1\|^2.$$

Then the separated signals

$$\boldsymbol{z}_1(k)=\boldsymbol{w}_1^T(k)\boldsymbol{y}(k) \qquad \boldsymbol{z}_2(k)=\boldsymbol{w}_2^T(k)\boldsymbol{y}(k) \tag{13}$$

and the training formulation of the separating matrix is :

$$\boldsymbol{w}_1'(k+1)=\boldsymbol{w}_1(k)-\mu\nabla_{\boldsymbol{w}_1}J_{CMA}(k) \tag{14}$$

where

$$J_{CMA}=\sum_{i=1}^{M}E(|\boldsymbol{z}_i|^2-1)^2+\frac{1}{\rho}\sum_{\substack{i,j=1\\ i\neq j}}^{M}\left|E(\boldsymbol{z}_i\boldsymbol{z}_j^*)\right|^2 \tag{15}$$

is constant modulus algorithm(CMA) cost function [7].

Normalize $\boldsymbol{w}_1$ such that $\alpha = \dfrac{1}{\rho}$

$$\boldsymbol{w}_1(k+1) = \frac{1}{\sqrt{\hat{\rho}}} \frac{\boldsymbol{w}_1^{'}(k+1)}{\left\| \boldsymbol{w}_1^{'}(k+1) \right\|} \tag{16}$$

where $\hat{\rho}$ is obtained through (12).

And

$$\boldsymbol{w}_2(k+1) = \boldsymbol{J}\boldsymbol{w}_1^{*}(k+1) \tag{17}$$

4 Simulation Results

This section investigates the bit error rate (BER) performance of the methods proposed in this paper. Simulations are achieved under OSTBC 2x1 system. The quasi-static flat Rayleigh fading channel and the BPSK modulation are assumed. Some other detecting methods such as blind, trained, semi-blind algorithms are also simulated for comparison [3].

Fig.3 shows the BER performance comparison of the EASI and the orthogonal EASI schemes. In the simulation, we assume μ =0.02, γ=0.9. As can be seen, the performance is improved more evident along with the increasing of SNR. When BER is at 10^{-4}, the orthogonal EASI scheme got approximately 4 *db* gain.

Fig.4 shows the BER performance comparison of the orthogonal J matrix scheme and the conventional channel estimation schemes. It can be seen this scheme has got better performance than trained and semi-blind schemes. Compared with semi-blind scheme, the orthogonal J matrix scheme got approximately 6 *db* gain.

Fig.5 shows another simulation result for the orthogonal J matrix scheme of the signals separation on constellation. Assume $SNR = 20\ db$, $\mu = 0.02$ and QPSK is used. (a) is the received mixture signals, and (b) is the separated signal. Results show, after the ICA detection, signals are separated successfully.

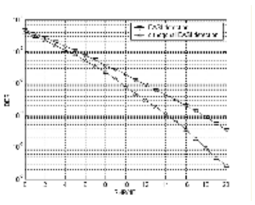


Fig. 3. BER performance of EASI and the orthogonal EASI scheme

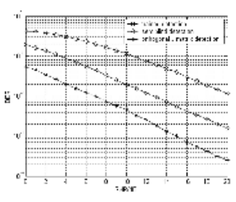


Fig. 4. BER performance comparison of different schemes

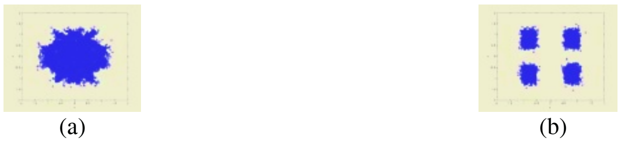


Fig. 5. Signals separation on constellation

5 Conclusions

In this paper, two methods based on ICA have been presented, which can detect the symbols transmitted by a system using OSTBC. The main advantage of ICA technique is that it does not require the channel estimation, therefore, no long training sequences are required, only a very short pilot sequence is used in order to estimate the correct order of the parallely transmitted sequences. Meanwhile, the spectral efficiency and the useful data rate are increased.

As we can see, the two proposed methods have notable performance. But the biggest challenge of these methods is that when there are more transmit antennas, the system performance will deteriorate quickly. Also future work will focus on how to decrease the length of the data block used for calculating the separating matrix, so that future schemes can cope with fast time-variant channel. What's more, how to generalize the schemes to MIMO system will be considered.

References

1. Alamouti, S. M.: A Simple Transmit Diversity Technique for Wireless Communications. IEEE Journal on Selec. Areas in Comm, **16** (1998) 1451-1458
2. Tarokh, V.: Space-Time Block Codes from Orthogonal Designs. IEEE Trans. Inform. Theory, **45** (1999) 1456-1467
3. Stoica, P., Ganesan, G.: Space-time Block Codes: Trained, Blind and Semi-blind Detection. Proc. of IEEE ICASSP (2002) 1609-1612
4. Xu, H., Liu, J., Hu, H.: Blind Detection Based Space-time Block Coding with Antenna Subset Selection. proc. the Seventh International Conference on Signal Processing (ICSP04), Beijing, China, **2** (2004) 1731-1734
5. Liu, J., Iserte, A. P., Lagunas, M. A.: Blind Separation of OSTBC Signals Using ICA Neural Networks. IEEE International Symposium on Signal Processing and Information Technology, Darmstadt, Germany (2003)
6. Cardoso, J. F., Laheld, B. H.: Equivariant Adaptive Source Separation, IEEE Trans. Signal Proc, **44** (1996) 3017–3030
7. Rinas, J., Kammeyer, K.D.: Comparison of Blind Source Separation Methods Based on Iterative Algorithms, 5th International ITG Conference on Source and Channel Coding (SCC04), Erlangen, Germany (2004)

Improvement of Borrowing Channel Assignment by Using Cellular Probabilistic Self-organizing Map

Sitao Wu and Xiaohong Wang

School of Electrical Engineering, Southwest Jiaotong University,
Chengdu, Sichuan 610031, China
wusitao@yahoo.com.cn

Abstract. Borrowing channel assignment (BCA) is used for efficient spectrum allocation in cellular systems. However, when many cells with heavy traffic cluster together and there are still some cells with light or medium traffic, i.e., patterned traffic load, BCA is not effective for spectrum allocation. In this paper, an improvement of BCA for patterned traffic load, based on traffic prediction by cellular probabilistic self-organizing map (CPSOM), is proposed. Upon the prediction of traffic that does not vary fast, the proposed method has better performance for patterned traffic load than the traditional BCA methods. Experimental results corroborate that the proposed method achieves higher satisfaction probability than the traditional FCA and BCA techniques.

1 Introduction

Research on how to efficiently utilize the scarce radio spectrum resource to satisfy the increasing users has become an exigent issue. Borrowing channel assignment (BCA) methods [1-6] are solutions to this problem. However, for BCA in certain situation, many cells with heavy traffic cluster together and the inner cells in the clusters cannot borrow channels since all the nominal channels in the nearby cells are used. Meanwhile, there are still unused nominal channels available in other cells with light or medium traffic. This situation is called patterned traffic load. The channel utilization is not fully optimized due to the unused nominal channels. In this paper, a BCA method based on cellular probabilistic self-organizing map (CPSOM) [7], is proposed to overcome this problem. The proposed method periodically predicts the current traffic load and re-assigns all the nominal channels for all cells. If the traffic load does not vary fast and is predicted with an acceptable level, heavy-traffic cells can obtain more nominal channels while the numbers of the nominal channels in other light-or-medium-traffic cells are decreased. Therefore the channel utilization by the proposed method for patterned traffic load is better than that by the traditional BCA methods. For other situations, the performance by the proposed method is similar to that by the traditional BCA methods. It is worth noting that the proposed method is not expected to increase the satisfaction probability too much because CPSOM in the proposed method is directly used for traffic prediction, but not directly used for optimal channel optimization. CPSOM in the proposed method can be utilized in any one of the BCA methods.

2 The CPSOM-Based BCA Method

If traffic load can be roughly predicted, all cells can be re-assigned such that heavy-traffic cells are given more nominal channels, and light-or-medium-traffic cells are given less nominal channels. The traffic-prediction technique adopted in this paper is the online CPSOM algorithm [7], which is an improvement of the traditional SOM algorithm [8]. The computation complexity of CPSOM [7] is not heavy if the number of neurons is not very large (<10). After traffic is predicted by CPSOM and patterned traffic load happens at that time, all nominal channels are re-assigned by a FCA method to give a better channel utilization. The total number of all the nominal channels after re-assignment is the same as that before re-assignment, e.g., example, the number of all the nominal channels in a cellular system with N cells is M. An item $x_i(t)$ in the input vector $[x_1(t),...,x_N(t)]^T$ in a cellular system with N cells at time t is the new calls for cell i during the latest one minute before time t. The weight vector W_i of neuron i in CPSOM is represented by $W_i = \left[w_{i1},...,w_{iN} \right]^T$. A nearest neuron, e.g., neurons i, to the current input $x(t)$ is found. The weight vector W_i of neuron i records one of the average traffics in the past and can be used to predict the future traffic trend if the traffic does not vary fast. The weight vector W_i is then transformed such that the sum of all the elements of the transformed W_i is M and W_i reflects the past traffic. During the re-assignment, some accepted calls may be dropped out. If only blocking probability is considered, the dropped calls are not considered for the proposed method. Therefore the later comparison among the proposed method and other methods is based on satisfaction probability, which is defined as the ratio of the number of all accepted calls without dropping to that of all calls. The higher the satisfaction probability is, the better the system is.

How do we detect a patterned traffic load? Firstly, the satisfaction probability in the past is between about 60% and 95% in practice. Secondly, when more than 60% of all cells have heavy traffic and more than 40% heavy-traffic cells have no neighboring cells with light or medium traffic.

Finally, the whole CPSOM-based BCA method is describes as follows.

Step 1) Initially use a FCA method to allocate nominal channels for each cell. Initialize the CPSOM network. Set current time t=0 second.

Step 2) Allocate channels for new calls at time t and borrow channels by a BCA method if possible.

Step 3) If the current iteration t is an integer multiple of 60, i.e., one minute later, an input vector *x(t)* is formed and is fed into the CPSOM network and train the CPSOM network online.

Step 4) If the current iteration t is an integer multiple of a parameter τ ($\tau \gg 60$ seconds), the nearest neuron i from *x(t)* is found. After W_i is transformed and patterned traffic load is detected, use a FCA method to allocate new nominal channels in background. Otherwise, go to *step 6.*

Step 5) If FCA is finished, re-assign the current accepted and new calls to the new nominal channels for all cells.

Step 6) $t = t + 1$, go to *step 2.*

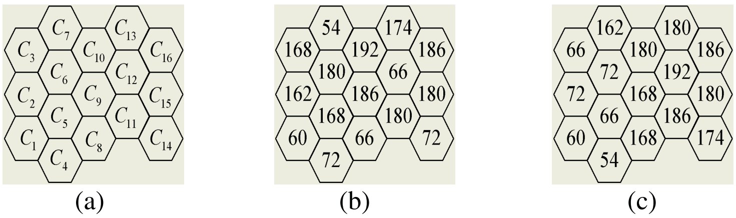


Fig. 1. The cellular system used in the experiments. (a) The symbol C_i in a cell denotes the *i*th cell; (b) Pattern 1 of nonuniform traffic distribution (calls/hour); (c) Pattern 2 (patterned traffic load) of nonuniform traffic distribution (calls/hour)

3 Experimental Results

The experiments were done on a cellular system shown in Fig. 1 (a). There are totally 16 cells and 80 channels available in the system. For simplicity, we do not consider adjacent channel interference. The reuse distance to avoid co-channel interference is set to three cell units. For unknown traffic load in advance, each cell is uniformly allocated with 10 channels. The BCA method used in our proposed method is SB. Three examples of different types of traffic loads were used in the experiments. Note that the three traffic loads are temporally stationary. All the experiments were simulated for 5 hours. For the CPSOM algorithm, a 3×3 network is used. The inputs to the CPSOM network are the new calls during the latest one minute before the current time and are normalized by dividing them over a constant 10. In the proposed method, the parameter τ is set such that $\tau = 3600$ seconds. In order to demonstrate the usefulness of the proposed method, the re-assignment was always performed in *step 4*, regardless of the detection of patterned traffic load.

The first experiment was done by using spatially uniform traffic in the system. The mean arrival rate λ is 1.5 calls/min. The call duration $\overline{x}$ is 3 minutes. The FCA and SB methods were used in the allocation. Since the chance of patterned traffic load is little, patterned traffic load was not detected in *step 4* of the proposed method. The increased (20%, 40%, 60%, 80% and 100%) and decreased (-20%, -40%) traffic loads were also considered. The satisfaction probability curves versus traffic loads for the three methods are plotted in Fig. 2 (a), where our proposed method and SB have very close performance and both are much better than FCA. This is reasonable and expected since underlying traffic is uniform (not patterned traffic load), and the re-assignment does not change the number of nominal cells in a cell too much.

The second experiment was done by using nonuniform traffic (pattern 1). The mean arrival rate is shown in Fig. 1 (b) and the call duration $\overline{x}$ is still 3 minutes. The light and heavy traffics differ dramatically. However, a heavy-traffic cell can still borrow a channel from its neighboring light-traffic cells if their nominal channels are not all used. Since patterned traffic load can seldom happen in this experiment, it was not detected in *step 4* of the proposed method. Fig. 2 (b) shows the performances among the FCA, SB and proposed method under different changes of traffic load. As expected, the proposed method and SB are much better than FCA. The performance by the proposed method and SB is very close to each other. Since the traffic in the second experiment is not patterned traffic load, the advantage of the proposed method cannot be reflected.

The third experiment was done by using nonuniform traffic (pattern 2). The mean arrival rate is shown in Fig. 1 (c) and the call duration $\overline{x}$ is still 3 minutes. The light and heavy traffic also differ dramatically. However, in this experiment some heavy-traffic cells cannot borrow channels since there are no nearby light-traffic cells. The number of heavy-traffic cells can occupy about 62% of total 16 cells. And about 60% heavy-traffic cells can have no light-traffic neighboring cell. So patterned traffic load is often detected in this experiment. Fig. 2 (c) shows the performance among the FCA, SB and the proposed method under different changes of traffic load. As expected, the proposed method and SB are much better than FCA. The advantage of the proposed method can be reflected in this traffic load such that the proposed method is better than SB by about 2% average increase of satisfaction probability for the traffic increases of 0%, 20%, 40% and 60%. Note that when the traffic is not very heavy, i.e., satisfaction probability is above 95% by the proposed method (98.3% for traffic decrease by 40% and 97.0% for traffic decrease by 20%), the performance of the

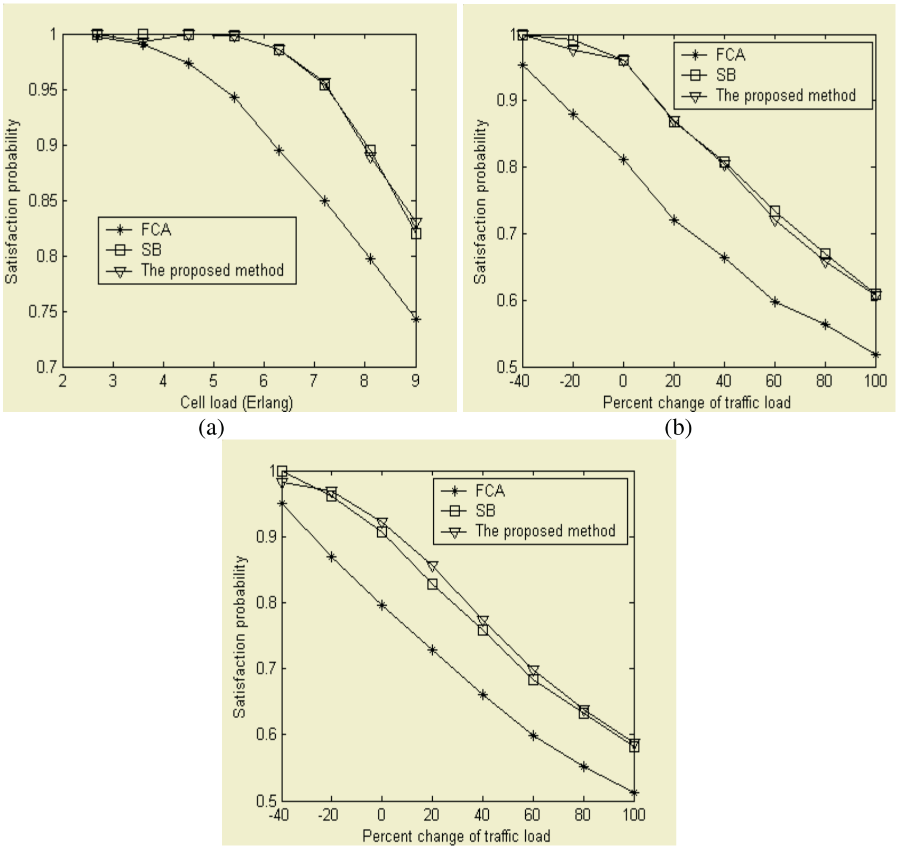


Fig. 2. Performance comparison among the three methods. (a) Uniform traffic; (b) Nonuniform traffic (pattern 1); (c) Nonuniform traffic (pattern 2)

proposed method is close to that by SB. This is because patterned traffic load can hardly happen when all cells have light or medium traffic. On the other extreme (satisfaction probability below 60%), the performance of the proposed method and SB is also close to each other when the traffic is increased to a large value. This is also expected since the traffic in each cell is heavy and there is no patterned traffic load. So no channels can be borrowed for the proposed method or SB such that they perform in a similar way.

In all the three experiments, the dropping probability is below 0.05% that is an acceptable small number.

4 Conclusions

In this paper, if we can roughly predict the short-term traffic load by CPSOM, the patterned traffic load can be detected. After detection, we can re-assign the nominal channels to all cells by using a FCA method. Experiments on three temporally stationary traffic loads demonstrate that our proposed method can have better performance than SB for patterned traffic load and much better performance than FCA. If other BCA methods are used with CPSOM-based prediction, the results will be similar. When the traffic load is temporally non-stationary and does not vary fast, like the stationary traffic, the proposed method will be better than the traditional FCA and BCA methods if patterned traffic load frequently occurs in the system.

References

1. Engel, J.S., Peritsky, M. M.: Statistically-Optimum Dynamic Sever Assignment in System with Interfering Severs. IEEE Trans. Vehicular Technology, **22** (1973) 203-209
2. Kahwa, T.J., Georganas, N.D.: A Hybrid Channel Assignment Scheme in Large-Scale Cellular-Structured Mobile Communication Systems. IEEE Trans. Communications, **26** (1978) 432–438
3. Elnoubi, S.M., Singh, R., Gupta, S.C.: A New Frequency Channel Assignment Algorithm in High Capacity Mobile Communication Systems. IEEE Trans. Vehicular Technology, **31** (1982) 125-131
4. Zhang, M., Yum, T.-K.P.: Comparison of Channel-Assignment Strategies in Cellular Mobile Telephone System. IEEE Trans. Vehicular Technology, **38** (1989) 211-215
5. Chang, K.-N., Kim, J.-T., Yim, C.-S., Kim, S.: An Efficient Borrowing Channel Assignment Scheme for Cellular Mobile Systems. IEEE Trans. Vehicular Technology, **47** (1998) 602-608
6. Sandalidis, H.G., Stavroulakis, P.P., Rodriguez-Tellez, J.: Borrowing Channel Assignment Strategies Based on Heuristic Techniques for Cellular Systems. IEEE Trans. neural networks, **10** (1999) 176-181
7. Chow, T.W.S. and Wu, S.: An Online Cellular Probabilistic Self-Organizing Map for Static and Dynamic Data Sets. IEEE Trans. Circuit and System I, **51** (2004) 732-747
8. Kohonen, T.: Self-Organizing Maps. Springer, Berlin (1997)

FPGA Realization of a Radial Basis Function Based Nonlinear Channel Equalizer*

Poyueh Chen[1], Hungming Tsai[2], ChengJian Lin[3,**], and ChiYung Lee[3]

[1] Department of Computer Science and Information Engineering
[2] Institute of Networking and Communication Engineering
Chaoyang University of Technology
[3] Department of Computer Science and Information Engineering
Nan Kai Institute of Technology
168 Gifeng E. Rd., Wufeng Taichung County, 413 Taiwan, China
568 Chung Cheng Rd., Tsao Tun, 542, Nan Tou County, Taiwan, China
cjlin@mail.cyut.edu.tw

Abstract. In this paper we propose a radial basis function (RBF) neural network for nonlinear time-invariant channel equalizer. The RBF network model has a three-layer structure which is comprised of an input layer, a hidden layer and an output layer. The learning algorithm consists of unsupervised learning and supervised learning. The unsupervised learning mainly adjusts the weight among input layer and hidden layer. The supervised learning adjusts the weight among output layer and hidden layer. We will implement RBF by using FPGA. Computer simulation results show that the bit error rates of the RBF equalize using software and hardware implements are close to that of the optimal equalizer.

1 Introduction

During the past few years, applications of high-speed communication are required and fast increasing. Nonlinear distortion becomes a major fact or which limits the performance of a communication system. High speed communications channels are often impaired by the channel inter-symbol interference (ISI), the additive white Gaussian noise (AWGN), and co-channel interference (CCI) [1]. All these effects are nonlinear and complex problems. Nevertheless, adaptive equalizers are used in digital communication system receivers to mitigate the effects of non-ideal channel characteristics and to obtain reliable data transmission.

We will adopt the radial basis function (RBF) neural network [2] suggested by Moody and Darken. Radial basis function (RBF) neural networks provide an attractive alternative to MLP for adaptive channel equalization problems because the structure of the RBF network has a close relationship to Bayesian methods for channel equalization and interference rejection problems. The main benefit was using linear algebra's basis

* This research is supported by the National Science Council under grant NSC 92-2213-E-252-006.

** Corresponding author.

mathematic, relatively reduced the computation load. Usually the computation quantity problem takes much more time to simulate through software. The solution is the utilization of hardware simulation to obtain faster computational effectiveness.

Development of digital integrated circuit Field Programmable Gate Array (FPGA) [3]-[5] digitizes the hardware making process. Recently programmable logic element increases the number of the logic, velocity and memory. And add a lot of extra function. In addition, use Very High Speed Integrated Circuit Hardware Description Language (VHDL), enable the complicated circuit to form the way through the circuit that VHDL compile, reach the specification designed easily and fast. Hence it holds lots of benefits like high capacity, speedy, duplicate design, cheaper price, and cost lower. Finally we will achieve this adapted, radiation basis function neural network equalizer by using hardware FPGA.

2 The Structure of RBF

The structure of the RBF neural network model is shown in Fig.1. The input data in the input layer of the network is $x = [x_1, x_2, x_3, \ldots, x_n]$, where n is the number of dimensions. The hidden layer consists of *m* computing units $(\phi_1 \ to \ \phi_m)$, which are connected to the output by m connection weights $(w_1 \ to \ w_m)$.

The output of the network used by this algorithm has the following form:

$$Y(x) = f(x) = \sum_{j=1}^{m} w_j \phi_j(x) \tag{1}$$

where ϕ_j is the response of the *j*th hidden neuron to the input x, is the weight connecting the *j*th hidden unit to the output unit. Here, *m* represents the number of hidden neurons in the network, and ϕ_j is a Gaussian function given by

$$\phi_j = exp(-\frac{\|x - c_j\|^2}{\sigma_j^2}) \tag{2}$$

where c_j is the center and σ_j is the width of the Gaussian. $\|\|$ denotes the Euclidean norm.

3 The Learning Algorithm for RBF

The learning algorithm consists of unsupervised learning and supervised learning. The unsupervised learning mainly adjusts the weight among input layer and hidden layer. The supervised learning adjusts the weight among output layer and hidden layer.

3.1 Unsupervised Learning

The unsupervised k-means clustering procedure is often employed as a part of the general learning algorithm to adjust RBF centers. This involves computing the squared distance between the centers and the network input vector, selecting minimum squared

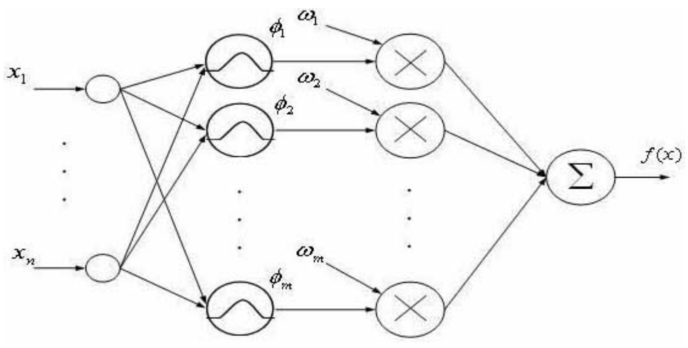


Fig. 1. RBF neural network structure.

distance and moving the corresponding center closer to the input vector. The computational procedure of this unsupervised clustering is as follows:

$$d_j(s) = \|x(s) - c_j(s-1)\|^2, \quad 1 \le j \le n \tag{3}$$

$$l = arg[min\{d_j(s), \quad 1 \le j \le n\}] \tag{4}$$

$$c_l(s) = c_l(s-1) + \alpha_c(x(s) - c_l(s-1)) \tag{5}$$

$$c_j(s) = c_j(s-1), \quad 1 \le j \le n \ \ and \ j \ne l \tag{6}$$

3.2 Supervised Learning

The supervised algorithm is very simple and robust. It is advisable to adjust the weights of the network so that the network can learn the general equalizer solution. The adaptation of the weights is achieved using the following supervised algorithm:

$$\phi_j(s) = exp(-\frac{\|x(s) - c_j(s)\|^2}{\sigma^2}), \quad 1 \le j \le n \tag{7}$$

$$\varepsilon(s) = t(s-\tau) - \sum_{j=1}^{n} w_j(s-1)\phi_j(s) \tag{8}$$

$$w_j(s) = w_j(s-1) + \alpha_w \varepsilon(s)\phi_j(s), \quad 1 \le j \le n \tag{9}$$

4 Hardware Implementation

We can plan FPGA into the function that a user wants through the design of CLB and connection of the connecting wire. As for we design FPGA, the assisting of software is needed. So we utilize ISE 6.2i software that Xilinx Company produces to do logic design.

The main part of the network structure of RBF neural network is design of hidden layer. The input layer can direct output of input value to input of hidden layer. The process does not pass through any operation. The output layer only needs to add up all outputs which hide layer with the adder. The part of hidden layer is to consist of gaussian function. The structure of gaussian function is made up by a subtraction, divider, multiplier and exponential function. The component shows to Fig.2. The part of the multiplier includes the adder and counter. The concept implementation use bit shift and add up it. The exponential function part can rely on Taylor's expansion approximatively. Taylor's expansion is as follows:

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots + \frac{x^m}{m!} \tag{10}$$

where *m* is number of order, x is input. The larger *m* is the more perfect, but it will be many logic gates and bit numbers used. We choose *m*=4 to make the realization of the hardware. Fig.3 shows the block diagram that its hardware implementation. It used three multipliers, three dividers and an adder. The adder adds the outputs of every order.

Finally, the hardware implementation of channel equalizer using RBF neural network structure, show in Fig.4. The chip uses Virtex-2 series 3 million logic gates that Xilinx Company produce. The RBF network structure is two input nodes, eight hidden nodes and an output node.

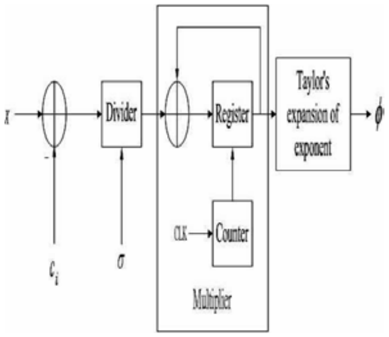


Fig. 2. The structure of gaussian function.

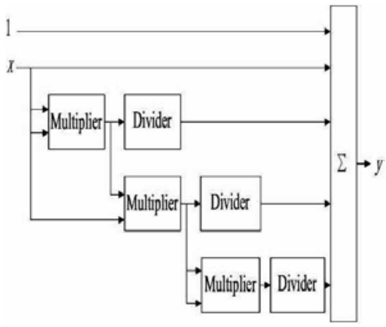


Fig. 3. Taylor expansion of exponential function.

5 Illustrative Examples

A discrete time model of a digital communication system is depicted in Fig.5. A random sequence x_i is passed through a dispersive channel of finite impulse response (FIR), to produce a sequence of outputs $\hat{y}_i$. A term, e_i, which represents additive noise in the system, is then added to each $\hat{y}_i$ to produce an observation sequence y_i. The problem to be considered is that of utilizing the information represented by the observed channel outputs $y_i, y_{i-1}, \ldots, y_{i-m+1}$ to produce an estimate of the input symbol x_{i-d} A device which performs this function is known as an equalizer. The integer's *m* and *d* are known as the order and the delay of the equalizer, respectively. Throughout, the input samples are chosen from $\{-1, 1\}$ with equal probability and assumed to be independent of one another.

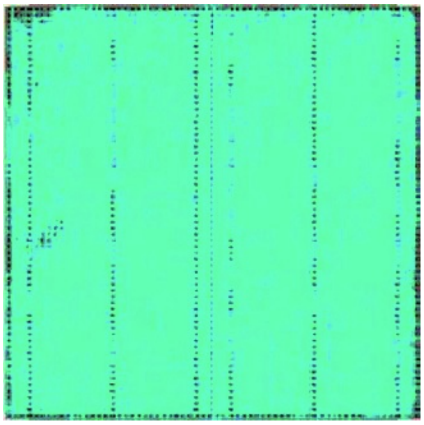


Fig. 4. RBFNN is realized on xc2v3000-5fg676.

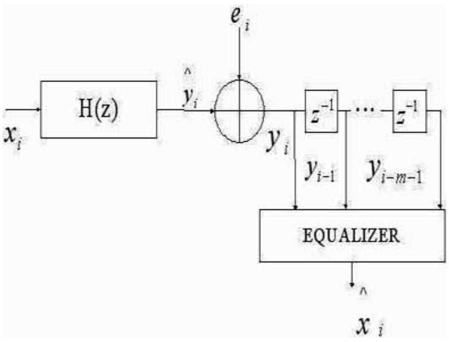


Fig. 5. Schematic of the data transmission system.

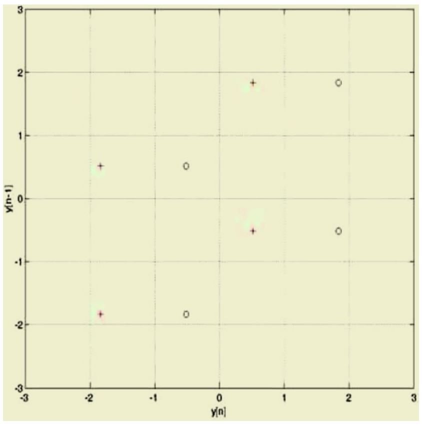


Fig. 6. Desired channel states with no noise.

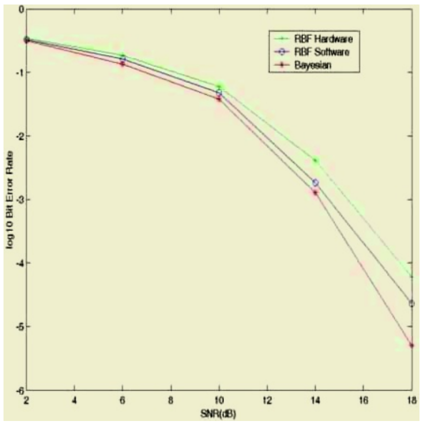


Fig. 7. Comparison of bit error rate curves.

The equalizer performance is described by the probability of misclassification with respect to the signal-to-noise ratio (SNR). With the assumption of independent identically distributed (i.i.d.) sequence the SNR can be defined as

$$SNR = 10log_{10}\frac{\sigma_s^2}{\sigma_e^2} \tag{11}$$

where σ_s^2 represents the signal power and σ_e^2 s the variance of the Gaussian noise.

The equalizer order is chosen as *m*=2. Let the channel transfer function be

$$\hat{y}(n) = (0.5x(n) + x(n-1)) + 0.1 \times (0.5x(n) + x(n-1))^3 \tag{12}$$

All the combinations of *x(n)* and the desired channel states are showed in Fig.6. To see the actual bit error rate (BER), a realization of 10^6 points of sequence *x(n)* and *e(k)* are used to test the BER of trained RBF neural network equalizer. It also tests the BER of hardware implementation of the RBF neural network equalizer. The resulting BER curve of the RBF neural network equalizer with software simulation and hardware implementation under the different SNR is show in Fig. 7.

We now compare the performance with the RBF neural network equalizer of software simulation, the RBF neural network equalizer of hardware implementation and Bayesian equalizer. The Bayesian equalizer is near optimal method for communication channel equalizer. Computer simulation results show that the bit error rate of the RBF neural network equalizer is close to the optimal equalizer. But we can see, some BER of hardware implementation is worse. Because the value of the hardware is expressed with binary, so there will be difference in value. If we want to close the BER of original RBF neural network structure, it is necessary to increase the number of bit.

6 Conclusion

The paper describes the RBF neural network structure as channel equalizer. The learning algorithm consists of unsupervised learning and supervised learning, in order to reach a better and a faster training method. And realize its structure of network using FPGA can obtain faster operation efficiency.

In the experiment of time-invariant channel, the result of simulation can be found out, the BER is close to Bayesian equalizer. If we make the BER of hardware simulation closer to original RBF network structure, it can be very easy to increase number of bit.

References

1. Khalid A. Al-Mashouq: The Use of Neural Nets to Combine Equalization with Decoding for Severe Intersymbol Interference Channels. IEEE Transaction on Neural Networks, **5** (1994)
2. S. Chen, Bernard Mulgrew, Peter M. Grant: A Clustering Technique for Digital Communications Channel Equalization Using Radial Basis Function Networks. IEEE Transaction on Neural Networks, **4** (1993)
3. Yutaka, Maeda, Toshiki, Tada: FPGA Implementation of a Pulse Density Neural Network With Learning Ability Using Simultaneous Perturbation. IEEE Transaction on Neural Networks. **14** (2003)
4. Chin Tsu Yen, Wan-de Weng, Yen Tsun Lin: FPGA Realization of a Neural-Network-Based Nonlinear Channel Equalizer. IEEE Transaction on Neural Networks, **51** (2004)
5. J.J. Blake, L.P. Maguire, T.M. McGinnity, B. Roche, L.J. McDaid: The Implementation of Fuzzy Systems, Neural Networks and Fuzzy Neural Networks Using FPGAs. Information Sciences (1998)

Varying Scales Wavelet Neural Network Based on Entropy Function and Its Application in Channel Equalization

Mingyan Jiang[1], Dongfeng Yuan[1,2], and Shouliang Sun[1]

[1] School of Information Science and Engineering, Shandong University, Jinan,Shandong 250100, China
{jiangmingyan,dfyuan,ssl}@sdu.edu.cn

[2] State Key Lab. on Mobile Communications, Southeast University, Nanjing, Jiangsu 210096, China

Abstract. This paper proposes a new kind of neural network named varying scales wavelet neural network to reduce wavelet-neuron number and simplify network structure. In order to avoid the local minima, entropy function is used as penalty function. The new network is applied to channel equalization, simulation results demonstrate that this network has less wavelet-neurons and recursive steps and can converge to the global minimum.

1 Introduction

Wavelet neural network (WNN)[1] becomes a research hotspot as a novel approximate tool and it has good performance in function approximation and pattern recognition applications[1].This paper proposes varying scales wavelet neural network (VSWNN) to channel equalization named VSWNNE and compares it with the linear transversal equalizer based LMS algorithm (LMS-LTE[2]), the linear transversal equalizer based RLS algorithm (RLS-LTE[2]), and the equalizer based wavelet neural network (WNNE[3]) in nonlinear channel.

2 Wavelet Neural Network Structure

According to wavelet analysis theory, we can use it to express any functions[4]. If scale m is big enough, we can approximate any function $g(x)$ by any precision. That is:

$$g(x)=\sum_{m,t}<g,\varphi_{m,t}>\varphi_{m,t}(x) \tag{1}$$

$$\varphi_{m,t}(x)=2^{m/2}\varphi(2^m x-t) \tag{2}$$

[1] This work is supported by the NSF of China (No. 60372030),the Key Project of Provincial Scientific Foundation of Shandong (No. Z2003G02).

Given scale m and t, due to equation (1) and (2), we have

$$g(x) = \sum_{i=1}^{N} w_i \varphi_{m,t_i}(x) + \overline{g} \tag{3}$$

Equation (3) is the mathematic expression of WNN where w_i is the weight, $\varphi(x)$ is the scale function, $\overline{g}$ is a constant to deal with nonzero mean functions, m is scale value, t_i is the shifting parameter and N is wavelet-neuron number. In multidimensional networks, the rotation matrix R is used[1].

3 Network Initialization and Learning Algorithm

Zhang[4] pointed out that the selection of wavelet scales in WNN has an influence on wavelet-neuron number and network performance. We propose VSWNN to get the precise scale and wavelet-neuron number and it can increase speed of network convergence.

Usual wavelet neural networks need some constraint conditions to prevent the network from dropping local minima[1]. Sun[5] proved that if entropy function $E(d,y) = d \bullet \ln y + (1-d) \bullet \ln(1-y)$, where d is the ideal network output and y is the real network output, is used as penalty function the network would jump out of the local minima and increase speed of convergence. So we replace mean square error (MSE) function with entropy function in VSWNN. Assume there is a nonlinear function expressed as $y = f(x_1, x_2, ..., x_n)$, where $x_1, x_2, ..., x_n \in (0,1)$, we adopt the following steps to initialize network and obtain the scale value:

(1) Initialize the network parameters: estimate the mean of the function $y = f(x_1, x_2, ..., x_n)$, set $\overline{g}$ to this mean: simply set w to random numbers in the range $(0,1)$: the rotations (R) are initialized to identity matrix[1]: in order to train network in roughest scale, set scale m to a small number:

(2) Set a certain initial iteration value and begin to train the network. Calculate the increment of entropy function $\Delta E = (E_k - E_{k-1}) / E_{k-1}$ per unit, which is defined as 20 iterations to deal with the case that network error increases by accident. E_k is the k-th unit E value. If: $\Delta E > \xi$, cancel current unit update of the weights, multiply m by η which is bigger than 1; $0 < \Delta E \le \xi$, current unit update of the weights is accepted, m and N need not to be changed: $\Delta E \le 0$, current unit update of the weights is accepted, multiply m by ρ which is smaller than 1.The threshold ξ is equal to 1%~5% commonly.

(3) We can find a suitable scale m from the step (2). Set the new wavelet-neuron number to $N_1 = 2^m + 1/2$ and calculate ΔE per unit while training network. If: $\Delta E > \xi$, cancel current unit update of the weights, set the new wavelet-neuron

number to $N_2 = N_1 = N_1/2$: $0 < \Delta E \le \xi$, current unit update of the weights is accepted, wavelet-neuron number need not to be changed: $\Delta E \le 0$, current unit update of the weights is accepted, set the new number of wavelet-neurons to $N_2 = N_1/2$:

(4) Choose suitable scale and wavelet-neuron number, then train the network till network error is lower than a given value.

(5) Test network by new samples, delete the nodes whose weights are very small and then optimize the network.

Suppose the network input and ideal output pairs are $(x(n), d(n))$, $y(n)$ is the real network output, the modifications of parameters in the step (4) are:

$$\Delta\bar{g}(k) = \gamma\Delta\bar{g}(k-1) - (1-\gamma)\alpha\mathrm{O}(n) \tag{4}$$

$$\Delta w_i(k) = \gamma\Delta w_i(k-1) - (1-\gamma)\alpha\mathrm{O}(n)2^{m/2}\varphi(z_i) \tag{5}$$

$$\Delta t_i(k) = \gamma\Delta t_i(k-1) + (1-\gamma)\alpha\mathrm{O}(n)2^{m/2}w_i R_i\varphi'(z_i) \tag{6}$$

$$\Delta R_i(k) = \gamma\Delta R_i(k-1) - (1-\gamma)\alpha\mathrm{O}(n)2^{m/2}w_i\varphi'(z_i)(2^m x - t_i)^T \tag{7}$$

Where $\frac{\partial E}{\partial \bar{g}} = \mathrm{O}(n)$, $z(i) = 2^m R_i(x - 2^{-m}t_i)$, $\mathrm{O}(n) = \frac{\partial E}{\partial y(n)} = \frac{d(n) - y(n)}{y(n)(1-y(n))}$, $\varphi' = d\varphi(x(n))/d(x(n))$. m is the scale value, N is wavelet-neuron number, $z(i)$ is the input of wavelet functions, α is learning rate, γ is momentum coefficient.

4 Varying Scales Wavelet Neural Network Equalizer and Simulation Results

Equalizer plays an important role in modern communication system when information transmission rate increases and channel distortions are severer. Traditional equalizers are FIR filters and perform badly since the real channels are nonlinear. Neural network equalizer becomes a research hotspot because neural network has very strong mapping ability. Wavelet neural network has faster speed of convergence and simpler structure in contrast to BP and radial basis function (RBF) networks.

The diagram of VSWNN equalizer is shown in Fig.1. The network adopts Gaussian-derivative function $\psi_s(x) = -xe^{-x^2/2}$ as transfer function of wavelet-neuron. Using noisy input/output measurements, we can make the size of network and modify every parameter based on network error. When the value of entropy function is lower than a fixed value, stop training network. In training process, the error is the difference of real network output and ideal network output. In transmitting process, the error is the difference of real network output and judging output.

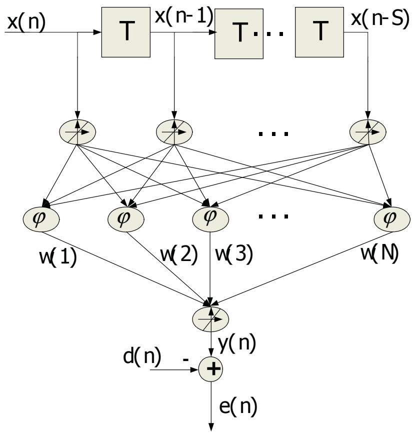


Fig. 1. VSWNNE structure diagram

The nonlinear channel is given as $H(Z) = 1 + 0.5z^{-1} + 0.5z^{-2}$. Channel input $x(n)$ is sequence of 0 and 1, Channel output is $y(n) = s(n) - 0.9[s(n)]^r$, where r is the order of channel and $S(Z) = H(Z) \bullet X(Z)$. Using dichotomy method to set less initial time, η and ρ are 2 and 1/2 respectively. Set initial iterations and threshold ξ to 200 and 4% respectively. LTE tap number and VSWNNE input layer neuron number are all 4 for impartiality. We use 5000 input/output data to train our network and then transfer 10^6 symbols to calculate bit error ratio (BER) with different signal noise ratio (SNR). The simulation results are as follows:

(1) When r=2, the performances of 3 equalizers are shown in Fig.2.The VSWNNE and WNNE clearly outperform the LMS-LTE and the RLS-LTE while VSWNNE performs a little better than WNNE.

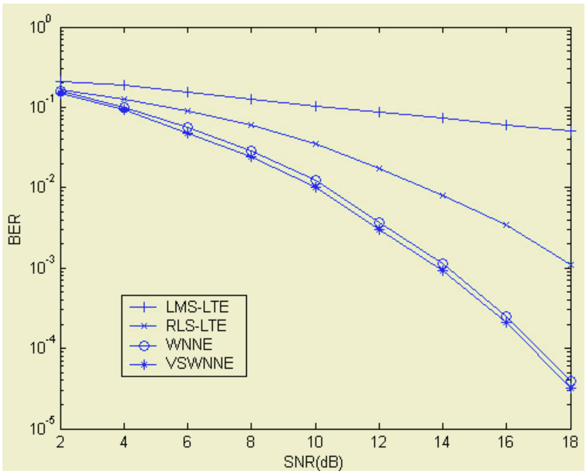


Fig. 2. The performances of four methods when r=2

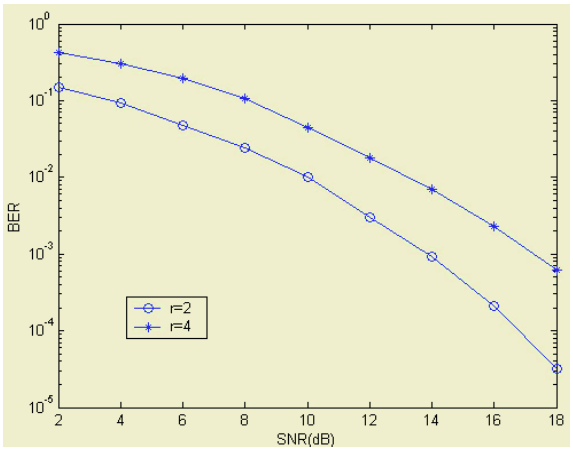


Fig. 3. The performances of VSWNNE when r=2,4

(2) When *r*=2 and 4, the performances of VSWNNE are shown in Fig.3.

(3) VSWNNE parameters in different scales are shown in Table 1. We can see that wavelet-neuron number N is increasing and network error E is descending when increasing scale m. We choose scale m=8 to overcome severe channel distortions while selecting scale m=4 can deal with slight distortions.

(4) VSWNNE size compared with WNNE size is shown in Table 2. We can see VSWNNE has a less size and a faster speed of convergence than WNNE with the almost same network error.

Table 1. VSWNNE parameters in different scales

m	1	4	8
E	0.295	0.0726	0.0486
N	5	14	26

Table 2. Comparison of VSWNNE and WNNE

Method	N	Iterations	Network error
WNNE	41	10000	0.0493
VSWNNE	26	5836	0.0486

5 Conclusions

This paper proposes a varying scales wavelet neural network and its learning algorithm. We apply the network to communication channel equalization. The constraint conditions needed by usual wavelet neural network have been removed, and the speed of convergence has been increased. We have also adopted initial iteration method to reduce the size of the network. Simulation results show that VSWNNE, which performs much better than LMS-LTE and RLS-LTE, can reduce size of WNN and has faster speed of convergence than WNNE.

References

1. Zhang, Q., Benveniste, A.: Wavelet Networks. [J] IEEE Trans on NN, **3** (1992) 889-898
2. Proakis, J.G.: Digital Communications(Fourth Edition). Publishing House of Electronics Industry (2003)
3. Chang, P.-R., Yeh, B.-F.: Nonlinear Communication Channel Equalization Using Wavelet Neural Networks. IEEE World Congress on Computational Intelligence, **6** (1994) 3605-3610
4. Zhang, J., Walter, G.G., Miao, Y., Wan, N., Lee, W.: Wavelet Neural Networks for Function Learning. IEEE Transactions on Signal Processing, **43** (1995) 1485-1497
5. Sun, Z., Hu, S., Zhang, S., Hou, W.: The Entropy Function Criterion for BP Algorithm. Journal of Northern Jiaotong University, **21** (1997)

Robust Direction of Arrival (DOA) Estimation Using RBF Neural Network in Impulsive Noise Environment*

Hong Tang, Tianshuang Qiu, Sen Li, Ying Guo, and Wenrong Zhang

School of Electronic and Information Engineering, Dalian University of Technology, Dalian, Liaoning 116024, China
tanghongdlut@yahoo.com.cn

Abstract. The DOA problem in impulsive noise environment is approached as a mapping which can be modeled using a radial-basis function neural network (RBFNN). To improve the robustness, the input pairs are preprocessed by Fractional Low-Order Statistics (FLOS) technique. The performance of this network is compared to that of the FLOM-MUSIC for both uncorrelated and correlated source. Numerical results show the good performance of the RBFNN-based DOA estimation.

1 Introduction

In sensor array processing, one of the critical problem has been the estimation of DOA of the source signals. Many high-resolution DOA estimation algorithms have been proposed. But, a common problem must be noted that the additive noise is assumed to be Gaussian distributed in these algorithms. However, this assumption is not always true in practice. There are always numerous man-made or natural electromagnetic disturbances [1] in wireless channel. The non-Gaussian impulsive noise is also be proved by experiments [2]. In open documents, FLOM-MUIC [3] is notable. But it still involve mass computational complexity.

Neural networks are candidates to carry on the computation tasks required in several array processing [4]. The estimation of DOA in impulsive noise environment is treated as a nonlinear mapping from the input pairs to the output pairs.

2 The Distribution of Impulsive Noise

In this paper, the alpha-stable distribution is introduced as the impulsive noise model in wireless channel. Alpha-stable distribution does not have closed form of united probability density function. It can be conveniently described by its characteristic function as $\varphi(t)=e^{\left\{ jat-\gamma|t|^{\alpha}[1+j\beta \operatorname{sgn}(t)\omega(t,\alpha)]\right\}}$. Where $\omega(t,\alpha)=\tan\frac{\alpha\pi}{2}$, if $\alpha\neq 1$,

* This work is supported by National Science Foundation of China under Grants 30170259, 60372081, and the Science Foundation of Liaoning Province under Grant 2001101057.

$\omega(t,\alpha) = \frac{2}{\pi}\log|t|$, if $\alpha = 1$, and $\mathrm{sgn}(\cdot)$ is a sign function. α is the characteristic exponent restricted in $0 < \alpha \le 2$. It controls the thickness of the tail in the distribution. γ $(\gamma > 0)$ is the dispersion parameter and β $(-1 \le \beta \le 1)$ is the symmetry parameter. a ($-\infty < a < \infty$) is the location parameter. When $\alpha = 2$ and $\beta = 0$, the α-stable distribution becomes the Gaussian distribution. The theoretical fact that the alpha-stable distribution can be a general noise model is proved by the generalized central limit theorem.

For jointly $S\alpha S$ random variables X and Y with $1 < \alpha \le 2$, the *fractional low-order covariance* (FLOC) of X with Y is defined as

$$c_{XY} = E(XY^{<p-1>}) \qquad 1 \le p < \alpha \tag{1}$$

Where $Y^{<p-1>} = |Y|^{(p-2)} Y^*$. The FLOC is a statistics of alpha stable random variables, which in a certain conditions plays a role for $S\alpha S$ (symmetry α-stable) random variables analogous to the one played by covariance for Gaussian random variables.

3 RBFNN for DOA Estimation

RBFNN is a method for approximating nonlinear mapping since the DOA estimation is of nonlinear nature. The nonlinear mapping from the input space to the output space can be viewed as a hypersurface Γ representing a multidimensional function of the input. During the training phase, the input-output patterns presented to the network are used to perform a fitting for Γ. The generalization phase presents an interpolation of the input data points along the surface Γ. The architecture of the neural network for DOA estimation is shown in Fig 1.

The received signals from the uniform linear array can be expressed as

$$x_m(n) = \sum_{k=1}^{K} a_k e^{j\left(\frac{\omega_k}{c}n+\phi_k\right)} e^{-j(m-1)d\sin\theta_k} + u_m(n) \quad m = 1,\cdots,M \quad n = 1,\cdots,N \tag{2}$$

Where K is the number of source signals, M is the number of elements of a linear array. N is the number of snapshots. a_k is the complex amplitude of the kth source signal, ϕ_k the initial phase, ω_k is the center frequency, and θ_k is the DOA of the kth source signal. $u_m(n)$ is the additive alpha-stable noise. The RBFNN is employed to perform the inverse mapping from the space of sensor output $\{\mathbf{x}(1),\mathbf{x}(2),\cdots,\mathbf{x}(Q)\}$ to the space of DOA $\{\boldsymbol{\theta}(1),\boldsymbol{\theta}(2),\cdots,\boldsymbol{\theta}(Q)\}$. The input vectors are mapped through the hidden layer and each output node computes a weighted sum of the hidden layer outputs, as shown in the following.

$$\theta_k(j) = \sum_{i=1}^{Q} g_i^k h(\| x(j) - x(i) \|^2) \qquad k = 1,\cdots,K \quad j = 1,\cdots,Q \tag{3}$$

Where Q is the number of training samples. $h(\cdot)$ can be selected as Gaussian function, g_i^k is the ith weight of the network.

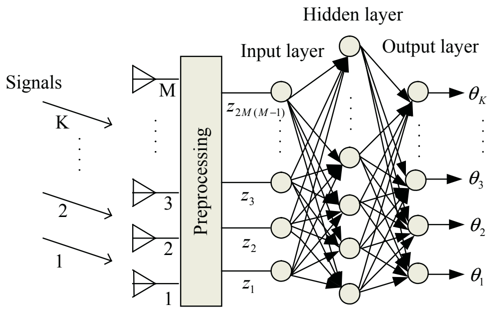


Fig. 1. The architecture of RBFNN for DOA estimation.

3.1 Data Preprocessing

Generating the array output vector according to equation (3), then we transform the vector into an input vector of RBFNN and produce the estimation of DOA. Since in the DOA estimation, the initial phase of signals ϕ_k contains no information about the DOA. It can be eliminated from the training data by forming the FLOC c_{ij}.

$$c_{ij} = \sum_{n=1}^{N} x_i(n)\left(x_j(n)\right)^{<p-1>} \qquad i,j = 1,\cdots,M \tag{4}$$

In Gaussian noise environment, spatial correlation preprocessor can be used. But, this second-order statistics (SOS) preprocessor is not valid here. So this paper uses a more robust preprocessor FLOC (5), we call it as FLOS-based preprocessor, seen in part 4. Since for $i = j$, c_{ij} does not carry any information on the DOA, the rest of the elements c_{ij} can be rearranged into a new input vector $\mathbf{b}$ given as

$$\mathbf{b} = \left[c_{21},\cdots,c_{M1},c_{12},\cdots,c_{M2},\cdots,c_{1M},\cdots,c_{M(M-1)}\right]^{T}_{M(M-1)\times 1} \tag{5}$$

It follows that the number of input nodes is given by $M(M-1)$. Note that we need twice as many input nodes for the neural network since RBFNN does not deal with complex numbers. Hence, the total number of input nodes should be $2M(M-1)$. The dimension of the hidden layer is equal to the number of the Gaussian functions that can be chosen as Q if the perfect recall is desired. Obviously, the number of output node is obviously equal to the number of signals K. In the simulation performed later, the relative signal power is taken as unity though different power levels do not affect the procedure of detecting the DOA. The input vector is then normalized by its norm in the training, testing, i.e. $\mathbf{z} = \mathbf{b}/\|\mathbf{b}\|$.

3.2 Network Training and Testing

Now, the steps for DOA estimation in alpha-stable noise environment can be summarized as follows:

1. Generate array output vectors $\{\mathbf{x}(q), q = 1,\cdots,Q\}$
2. Estimate the FLOC matrix of the array output vector $\{\mathbf{C}(q), q = 1,\cdots,Q\}$

3. Form the vectors $\{\mathbf{b}(q), q=1,\cdots,Q\}$ using (6) and form the normalized vectors $\{\mathbf{z}(q), q=1,\cdots,Q\}$ using (7)
4. Form the training set $\{\mathbf{z}(q), q=1,\cdots,Q\}$ and employ an appropriate training procedure to learn the training set.
5. Form the test set $\{\mathbf{z}_{test}\}$ to evaluate the RBFNN.

The main advantage with an RBFNN over other neural network approaches is that it does not require training the network with all possible combinations of input vectors. In this paper, we generate the source signals uniformly distributed from -90° to $+90^\circ$ to train and test a RBFNN in the simulation.

4 Flom-Music Algorithm

The classical MUSIC degrades because the covariance is sensitive to the impulsive noise, i.e., the covariance is unbounded in alpha-stable noise environment. Boundedness is important for statistical signal processing in that statistical analysis cannot evaluate unbounded quantities to obtain reasonable conclusions. In open documents, the ROC-MUSIC [3] and FLOM-MUIC [4] are notable. Both algorithms are based on the fractional low-order statistics (FLOS) because the FLOS is bounded and valid to impulsiveness distribution.

The vector form of (3) is given as $\mathbf{x}(n) = \mathbf{A}\mathbf{s}(n) + \mathbf{u}(n)$, where $\mathbf{x}(n)$ = $[x_1,(n), \cdots, x_M(n)]^T$ is the output vector of sensor array, $\mathbf{A}$ is the steering matrix, $\mathbf{s}(n)$ = $[s_1,(n), \cdots, s_K(n)]^T$ is the source signal vector and $\mathbf{u}(n)$ = $[u_1,(n), \cdots, u_M(n)]^T$ is a alpha-stable noise vector. According to the definition of FLOC in (2), the FLOC matrix $\mathbf{C}$ can be given as

$$\mathbf{C} = E[\mathbf{X}\mathbf{X}^{<p-1>}] = \mathbf{A}\mathbf{\Lambda}\mathbf{A}^{\mathrm{H}} + \gamma\mathbf{I} \tag{6}$$

The detailed expression of $\mathbf{\Lambda}$ and γ can be found in [4]. From (9), it is found that the DOA information is involved in $\mathbf{C}$. $\mathbf{C}$ is considered with full rank and sub-space techniques can be applied to extract DOA information. Perform SVD on $\mathbf{C}$, and construct the $M\times(M-K)$ matrix $E_n = \lfloor e_{K+1}, e_{K+2,} \cdots, e_M \rfloor$, where $\lfloor e_{K+1}, e_{K+2,} \cdots, e_M \rfloor$ are the left singular associated with the smallest $M-K$ singular values of $\mathbf{C}$. Compute the null spectrum $S(\varphi) = 1/\mathbf{a}^H(\varphi)\mathbf{E}_n\mathbf{E}_n^H\mathbf{a}(\varphi)$. where the $M\times 1$ steering vector $a(\varphi)$ is $\lfloor 1, e^{-j2\pi(d_2/\lambda)\sin\theta}, \cdots, e^{-j2\pi(d_M/\lambda)\sin\theta} \rfloor$. Note that $\mathbf{C}$ should be computed by time average instead of ensemble average. The peak locations of (10) are the estimation of DOAs. For DOA estimation of correlated/coherent signals, spatial smoothing is a possible way to decorrelate [5].

5 Simulation Results

In the simulation performed, a Generalized Signal Noise Ratio (GSNR) is defined as $\mathrm{GSNR} = 10\log\left(E\left\{|s(t)|^2\right\}/\gamma\right)$. In fact, the GSNR is a direct generalization from SNR.

When the alpha-stable noise reduces to the Gaussian noise, the GSNR is the ratio of signal variance to noise variance.

Uncorrelated signal sources: An array of $M = 10$ elements is used. A hidden layer of 200 nodes was chosen. A total of 1000 snapshots for training and testing are available to the simulations. The characteristic exponent of the alpha-stable noise is typically chosen to $\alpha = 1.5$. The width parameter of Gaussian function is uniformly set as 0.05. In Fig 2, the array receives two uncorrelated signals with different angular separations $(\Delta\theta = 10^\circ$ and $15^\circ)$ where the DOAs are assumed to be uniformly distributed from -90° to 90° in both the training and testing phases with GSNR=0 dB. 200 input vectors were used for training. For the test phase, 50 input vectors were used for the networks simulated with $\Delta\theta = 10^\circ$ and 15°. The results show that the network successfully produces actual outputs (+) very close to the desired DOA (solid lines). This also demonstrates that the network improved its performance through generalization and yielded satisfactory results.

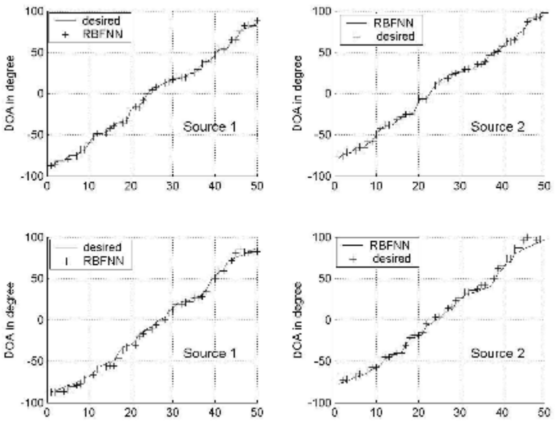


Fig. 2. DOA estimation versus number of samples Q using RBFNN. Source 1 is varied from –90° to 90°, while source 2 is 10° and 15° separated from source 1.

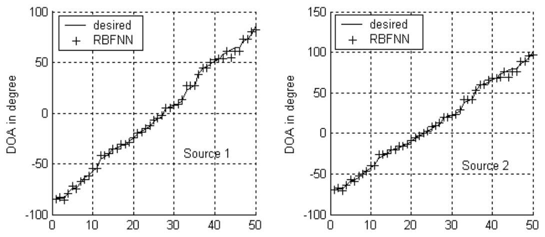


Fig. 3. DOA estimation with two coherent signals with $\Delta\theta = 10^\circ$.

Coherent signal sources: To study the effect of coherent signal sources on the performance of the RBFNN, the testing data was generated assuming the array receives two coherent signals with angular separation of 10°. The training data was performed with data derived from signals with GSNR=0 dB and the testing was performed after spatial smoothing with data contaminated by additive alpha-stable noise with GSNR= 0 dB. DOA obtained from RBFNN for coherent signals are plotted in fig 3.

6 Conclusions

This paper introduces alpha-stable distribution as the impulsive noise model of wireless channel. To avoid huge computation, the problem of DOA estimation is treated as a nonlinear mapping from sensors output to the DOA. A suitable RBFNN is employed to approximate the nonlinear mapping. FLOS-based preprocessor suppresses the negative effect of impulsiveness successfully.

References

1. Nikias, C. L., Shao, M.: Signal Processing with Alpha-Stable Distribution and Application. New York, John Wiley & Sons, Inc (1995) 13-108
2. Blackard, K. L., Rappaport, T. S.: Measurements and Models of the Radio Frequency Impulsive Noise for Indoor Wireless Communications. IEEE Journal on Selected Areas in Communications (1993) 991-1001
3. Tsung-Hsien, L., Mendel, J. M.: A Subspace-Based Direction Finding Algorithm Using Fractional Lower Order Statistics. IEEE Transactions on Signal Processing (2001) 1605-1613
4. El Zooghby, A. H., Christodoulou, C. G., Georgiopoulos, M.: Performance of Radial-basis Function Networks for Direction of Arrival Estimation with Antenna Arrays. IEEE Transactions Antennas and Propagation (1997) 1611-1617
5. Rupi, M., Tsakalides, P., Re, E.D., Nikias, C. L.: Robust Spatial Filtering of Coherent Sources for Wireless Communications. Elsevier Signal Processing (2000) 381-396

Quantum Neural Network for CDMA Multi-user Detection*

Fei Li, Shengmei Zhao, and Baoyu Zheng

Institute of Signal & Information Processing, Nanjing
University of Posts and Telecommunications, Nanjing 210003, China
lifei@njupt.edu.cn

Abstract. Quantum Neural Network (QNN) is a new field which integrates ANN with quantum computing. Taking advantages of the properties of quantum mechanics such as quantum parallelism and entanglement, QNN may give us unprecedented possibilities in creating new system for information processing. The fundamental of QNN is introduced and a model for quantum neuron is described. A novel multi-user detector based on quantum neuron and quantum register is proposed. The simulation results show that the proposed detector has more powerful properties both in bit error rate and near-far resistance than Hopfield neural network based detector.

1 Introduction

In recent years Multi-User Detection (MUD) has received considerable attention as one of the most important signal processing task in wireless communication. Verdu [1] has proven that the optimal solution is equivalent with the optimization of a quadratic form, which yields in MLSE (Maximum Likelihood Sequence Estimation) receiver. However, to find the optimum by exhaustive search is a NP-hard problem as the number of users grows. Many authors proposed sub-optimal linear and nonlinear solutions such as Decorrelating Detector, MMSE (Minimum Mean Square Error) detector, Multistage Detector, Hopfield neural network or Stochastic Hopfield neural network [1], [2], [3], and the references therein. Nonlinear sub-optimal solutions provide quite good performance, however, only asymptotically. Quantum computation based Quantum Multi-user detection algorithms seem to be able to fill this long-felt gap [4].

In the field of ANN, some pioneers introduced quantum computation into analogous discussion such as quantum neural computing, quantum-inspired neural networks and quantum associative memory [5], [6]. They constructed the foundation for further study of quantum computation in ANN. From this scenario, emerge the field of artificial neural networks based on quantum theoretical concepts and techniques. They are called Quantum Neural Network (QNN) [6], [7].

In this paper we propose a novel multi-user detection algorithm based on quantum neural network. The rest of this paper is organized as: In Section 2 we describe the mechanism of quantum neuron. In Section 3 we shortly review the basic notations of multi-user detection. In Section 4 the proposed quantum neural network based multi-user detector model is introduced. In Section 5 the simulation results are discussed. Finally, we give conclusions in Section 6.

* The work is supported by the National Natural Science Foundation of China (No. 60272066).

2 Quantum Neuron

Consider a neuron with N inputs $|x_1>\ldots|x_N>$ as shown in Figure1. $|x_j>$ is a quantum bit (qubit) expressed as

$$\left|x_j\right\rangle = a_j\left|0\right\rangle + b_j\left|1\right\rangle = (a_j, b_j)^T . \tag{1}$$

where a_j and b_j are complex numbers, $|\, a_j\,|^2+|\, b_j\,|^2=1$.

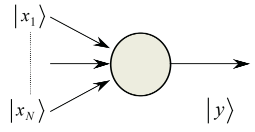


Fig. 1. The Model of Quantum Neuron

A qubit is a unit vector in the two-dimensional Hilbert complex vector space for which a particular basis, denoted by {|0>,|1>}, has been fixed. The orthonormal basis {|0>,|1>} can be expressed as $\{(1, 0)^T, (0, 1)^T\}$. For the purposes of quantum computation, the basis states |0> and |1> are taken to represent the classical bit values 0 and 1 respectively. Unlike the classical bit however, a qubit can be in a superposition of |0> and |1> such as (1). Combining $<x_j|$ and $|x_k>$ as in $<x_j|x_k>$, denotes the inner product of the two vectors, it's a scalar. The notation $|x_j><x_k|$ is the outer product of $|x_j>$ and $|x_k>$, it's an operator.

In Figure 1, the output $|y>$ is also a qubit derived by the rule [7]:

$$\left|y\right\rangle = \hat{F}\sum_{j=1}^{N}\mathbf{W}_j\left|x_j\right\rangle . \tag{2}$$

where $\mathbf{W}_\mathrm{j}$ is 2×2 matrix acting on the qubit $|x_j>$, $\hat{F}$ is an operator. We call the neuron in Figure 1 Quantum Neuron (QN).

3 Multi-user Detection

Let us investigate a one-path uplink wideband CDMA propagation channel, however, an extension to multi-path model can be done, effortlessly. Assume that K users are transmitting $(2M+1)$ symbol long packets on the same channel. The channel distortion for the kth user is modeled by a simple impulse response function $h_k(t) = \alpha_k\delta(t - \tau_k)$, where α_k and τ_k are the path gain and the delay of the kth user, respectively. The received signal is as follows:

$$r(t)=\sum_{i=-M}^{M}\sum_{k=1}^{K}\sqrt{E_k}\alpha_k b_k(i)s_k(t-iT-\tau_k)+n(t) . \tag{3}$$

where $b_k(i)$ is the ith symbol of the kth user, $b_k(i)\in\{+1, -1\}$, $s_k(t)$ and E_k is the continuous signature signal and energy associated to the kth user, T is the time period of one symbol, and $n(t)$ is zero mean white Gaussian noise with one-sided spectral density N_0, respectively. In case of signatures limited to one symbol length and if the system is properly synchronized, then there is no inter-symbol interference.

Consequently, index *i* can then be omitted from (3), yielding:

$$r(t)=\sum_{k=1}^{K}\sqrt{E_k}\alpha_k b_k s_k(t-\tau_k)+n(t)\,. \tag{4}$$

A conventional detector contains a filter bank of *K* filters which are matched to the corresponding signature waveforms and channels and calculates the following decision variable:

$$y_k=\sqrt{E_k}\alpha_k\int_0^T r(t)s_k(t)dt=b_k\rho_{kk}+\sum_{l=1,l\neq k}^{K} b_l\rho_{kl}+n_k\,. \tag{5}$$

where $\rho_{kl}=\sqrt{E_k}\sqrt{E_l}\alpha_k\alpha_l\int_0^T s_k(t)s_l(t-\tau_l)dt$ and $n_k=\sqrt{E_k}\alpha_k\int_0^T n(t)s_k(t)dt$ is still a zero mean white Gaussian noise due to linear transformation.

The optimal Bayesian detection reduces the optimal multi-user detection scheme to the following minimization problem in vector form [1]:

$$\mathbf{b}^{opt}=\arg\min_{\mathbf{b}\in\{-1,+1\}^K}\left[-2\mathbf{b}^T\mathbf{y}+\mathbf{b}^T\mathbf{R}\mathbf{b}\right]. \tag{6}$$

where $\mathbf{y}=[y_1, y_2, \ldots, y_K]^T$, $\mathbf{b}=[b_1, b_2, \ldots, b_K]^T$ and $\mathbf{R}=[\rho_{kl}$, k=1,2,…,*K* *l*=1,2,…,*K*$]$ is a symmetric matrix.

Unfortunately, the search for the global optimum of (6) usually proves to be rather tiresome, which prevents real time detection (its complexity by exhaustive search is $O(2^K)$). Therefore, our objective is to develop new, powerful detection technique, which paves the way toward real time MUD.

4 Quantum Neural Network Based Multi-user Detector

4.1 Representation of Received Sequences in Qregisters

We prepare an n-qubit "quantum register" (Qregister) $|y\rangle=|\varphi_{n-1}\varphi_{n-2}\cdots\varphi_0\rangle=\sum_{i=0}^{K-1}y_i|i\rangle$, which represent the received signal. $|y>$ in vector form is $[y_1, y_2, \ldots, y_K]^T$ in Hilbert Space. $|y>$ is set up from qubits spanned by $|i>$ $i=0\ldots(K\text{-}1)$ computational bases, where $K=2^n$ states can be stored in the Qregister at the same time. It is worth mentioning, that a transformation on a Qregister is executed parallel on all *K* stored states, which is called quantum parallelism. Applying this feature a transformation on *K* states stored in the Qregister $|y>$ can be done in one single step in quantum computers. It is true, however, using classical sequential computers, this operation could take rather long time.

4.2 Quantum Multi-user Detector

The structure of the proposed detector based on quantum neuron is depicted in Figure 2. The input of the QN (quantum neuron) $|y>$ is a Qregister which represent the received signal. The output of the QN $|\underline{b}>$ is also an n-qubit Qregister. **W** is a $K\times K$ connection weight matrix.

The state transition rule is given as follows:

$$|\underline{b}(t+1)\rangle=\hat{F}\left|\underline{b}'(t+1)\right\rangle=\hat{F}\left[|y\rangle-\mathbf{W}|\underline{b}(t)\rangle\right] \tag{7}$$

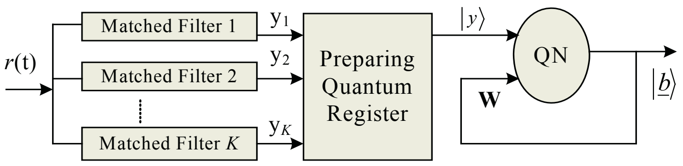


Fig. 2. The structure of the proposed detector

where $\hat{F}$ is an operator. We design an operator as

$$\hat{F}_1 = diag\left[sign(\cdot), sign(\cdot)...sign(\cdot)\right] \tag{8}$$

Then equation (7) can be represented in vector form as

$$\begin{bmatrix} \underline{b}_1(t+1) \\ \underline{b}_2(t+1) \\ \vdots \\ \underline{b}_K(t+1) \end{bmatrix} = \begin{bmatrix} sign(y_1 - \sum_{j=1}^{K} W_{1j}\underline{b}_j(t)) \\ sign(y_2 - \sum_{j=1}^{K} W_{2j}\underline{b}_j(t)) \\ \vdots \\ sign(y_K - \sum_{j=1}^{K} W_{Kj}\underline{b}_j(t)) \end{bmatrix} \tag{9}$$

The state transition rule given in (9) is similar to that of discrete Hopfield neural network[8]and it minimizes the following quadratic form over the state space$\{-1,1\}^K$:

$$J(\underline{\mathbf{b}}(t)) = \underline{\mathbf{b}}^T(t)\mathbf{W}\underline{\mathbf{b}}(t) - 2\underline{\mathbf{b}}^T(t)\mathbf{y} \tag{10}$$

This quadratic form is often referred to as Lyapunov function. One must note the visible similarity between (6) and (10), which makes possible to use the proposed detector for the purpose of MUD with the following replacements: $W_{kl}=\rho_{kl}$, if $k\neq l$ and $W_{ii}=0$. Equation (10) appears in quantum state form as:

$$|\underline{b}\rangle^{opt} = \arg\min_{\mathbf{b}\in\{-1,+1\}^K}\left[-2\langle\underline{b}|y\rangle + \langle\underline{b}|\mathbf{W}|\underline{b}\rangle\right] \tag{11}$$

Observe equation (7) and (9), we can find that the sign function is acting on each element of Qregister in parallel. So we called the operator $\hat{F}_1$ as parallel operator.

Considering the quantum superposition state can collapse into one of the basis states randomly when it is measured, we design a random operator as:

$$\hat{F}_2 = diag\left[f(\cdot), f(\cdot)...f(\cdot)\right],\ f(x) = sign(|x|^2 - u) \tag{12}$$

where u is a random number generated from the range [0,1].

5 Simulation Results

We report simulations of the Hopfield Neural Network based multi-user detector (HNN-MUD), our Quantum Neural Network based multi-user detector using the parallel operator $\hat{F}_1$(QNN-MUD1) and the random operator $\hat{F}_2$(QNN-MUD2). The simulations were performed in a K=8 users environment with 31 length Gold codes. Both the input and output quantum states are the 3-qubit Qregister in Figure 2. On the channel, power control was applied and we assumed perfect knowledge of channel parameters.

The bit-error rate (BER) versus signal-noise ratio (SNR) is depicted in Fig.3. Note that Single User Detector (SUD) resulted absolutely unusable in this environment. Obviously, both QNN-MUD1 and QNN-MUD2 outperform the HNN-MUD. The QNN-MUD2 has lower BER than does the QNN-MUD1, because QNN-MUD2 applying the random operator can get rid of the shortcoming of local optimization of Hopfield net much more. The bit-error rate (BER) versus near-far ratio is depicted in Fig.4. Note that both QNN-MUD1 and QNN-MUD2 outperform the HNN-MUD in the performance of near-far resistance.

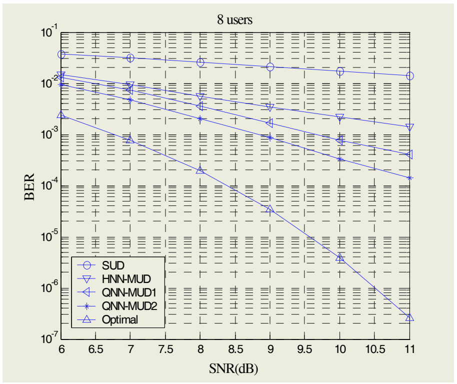


Fig. 3. Bit Error Ratio vs. Signal-Noise Ratio

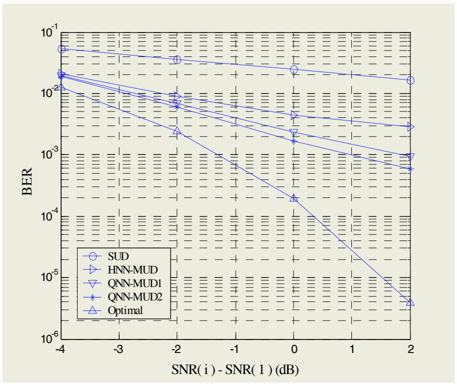


Fig. 4. Bit Error Ratio vs. Near-Far Ratio

6 Conclusions

In this paper we have proposed a novel multi-user detection scheme, which combines the advantages of quantum computing and neural networks. The new method resulted in better performance than traditional Hopfield Neural Network based detector , namely a 1-2 dB gain in performance can be achieved. The structure of the proposed multi-user detector is simpler than that of traditional Hopfield Neural Network and the algorithmic complexity of the new method does not increase exploiting the quantum parallelism.

References

1. Verdu, S.: Multiuser Detection. Cambridge Univ. Press (1998)
2. Varnashi, M., Aazhang, B.: Multistage Detection for Asynchronous Code-division Multiple Access Communication. IEEE Trans. Commun., **38** (1990) 509-519
3. Jeney, G., Levendovszky, J.: Stochastic Hopfield Network for Multi-user Detection. In European Conf. Wireless Technology (2000) 147–150
4. Imre, S., Balázs, F.: Quantum multi-user detection. Proc. 1st. Workshop on Wireless Services & Applications, (2001) 147–154
5. Kak, S.: On Quantum Neural Computing. Information Science, **83** (1995) 143-160
6. Ezhov, A., Venture, D.: Quantum Neural Networks. In: Kasabov, N. (ed.): Future Directions for Intelligent Systems and Information Sciences (2000) 213-234
7. Altaisky, M. V.: Quantum neural network. http://xxx.lanl.gov/quant-ph/0107012 (2001)
8. Hopfield, J.J., Tank, D.W.: Neural Computation of Decision in Optimization Problems. Biological Cybernetics, **52** (1985) 141-152

A New QoS Routing Optimal Algorithm in Mobile Ad Hoc Networks Based on Hopfield Neural Network

Jian Liu[1], Dongfeng Yuan[1,2], Song Ci[3], and Yingji Zhong[1]

[1] School of Information Science and Engineering, Shandong University, Jinan, Shandong 250100, China
{jianliusdu,dfyuan,zhongyingji32}@sdu.edu.cn

[2] State Key Lab. on Mobile Communications, Southeast University, Nanjing, Jiangsu 210096, China

[3] Computer Science, University of Michigan-Flint, Flint, MI 48502, USA
cisong@umflint.edu

Abstract. This paper presents a new model based on Hopfield neural network (HNN) that solves the optimum routing problem for supporting QoS in Ad hoc networks. The model is based on the minimum cost under delay constraints in ad hoc network in order that the neural network could be robust to the changed of the network topology. Under this assumption the general principles involved in the design of the proposed neural network and the method regarding the relationships of different coefficients in the energy function are discussed. The computational power of the proposed neural model is demonstrated through computer simulations. Simulation results indicate that our proposed scheme is very effective and outperforms previous schemes.

1 Introduction

A wireless ad-hoc network[1] is self-organizing, rapidly deployable, and without fixed infrastructure. Due to the lack of fixed infrastructure, the development of efficient protocols to support the various networking operations (e.g. Quality of Service (QoS) support, etc.) presents many issues and challenges. Recently there have been many reports on QoS efforts in ad hoc networks [1], which have focused on designing QoS routing algorithm [1][2]. The QoS routing algorithms are constraint-based routing algorithms and depend on the metrics chosen for the routes. Common route metrics in constraint-based routing can be divided into three classes, which are additive, multiplicative and concave. According to this classification, metrics *delay*, *jitter*, and *cost* are additive, *reliability* is multiplicative, and *bandwidth* is concave [3]. The computing optimal routes subject to constraints of two or more additive and/or multiplicative metrics is NP-complete.

[1] The authors thank the following foundations: National Scientific Foundation of China (No. 60372030); China Education Ministry Foundation for Visiting Scholar (No. [2003] 406); Key Project of Provincial Scientific Foundation of Shandong (No. Z2003G02); China Education Ministry Foundation for Ph. D in Chinese University (No. 20030422031); China Education Ministry Foundation for State key Lab. on Mobile Communications (No. A0205)

Neural networks have often been formulated to solve difficult (e.g. NP-complete) optimization problems. Hopfield and Tank [4] initially demonstrated the computational power of the neural network. The advantage of the HNN is the rapid computational capability of solving the combinatorial optimization problem. In particular, Ali and Kamoun [5] proposed an adaptive algorithm that utilized link cost and topology information. Ahn and Ramakrishna [6] proposed a near-optimal routing algorithm employing a modified HNN. In this paper, a new HNN model is proposed in Ad hoc networks to speed up convergence whilst improving route optimality under multi-constraints, namely the delay and cost.

2 A Hopfield Neural Network Solution to the QoS Routing Problem

In a large scale of Ad hoc network the cluster approaches are usually employed, in which all nodes are grouped into clusters. A cluster is a subset of nodes, which can communicate with a clusterhead and possibly with each other. In Fig. 1, nodes A, B and C are clusterheads for their coverage area respectively. Each of them serves as a regional broadcast node, and as a local coordinator to enhance channel throughput.

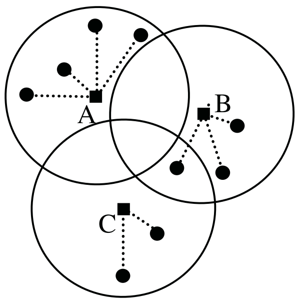


Fig. 1. Example of Clustering Architecture in Ad hoc network

Ad hoc network topology can be described by the undirected graph **G** = (**V**, **E**), where **V** is a set of vertices (**N** nodes) and **E** is a set of its edges (links) [7]. A path inclusion criterion matrix, denoted by $\boldsymbol{\alpha} = [\alpha_{ij}]$, represents the output values of neurons and each element is defined as follows:

$$\alpha_{ij} = \begin{cases} 1 & \text{if link from node } i \text{ to node } j \text{ exists in routing path} \\ 0 & \text{otherwise} \end{cases}$$

There is the delay t_{ij} and cost c_{ij} associated with each link (i, j) and they are specified by the delay matrix $\mathbf{T} = [t_{ij}]$ and the cost matrix $\mathbf{C} = [c_{ij}]$. Source and destination nodes are denoted by **s** and **d**, respectively. The link connection indicator matrix $\boldsymbol{\beta} = [\beta_{ij}]$ is defined as follows:

$$\beta_{ij} = \begin{cases} 1 & \text{if link from node } i \text{ to node } j \text{ does not exist in routing path} \\ 0 & \text{otherwise} \end{cases}$$

It is obvious that all the diagonal elements of α_{ij} must be zero. Then the expression of the delay and cost from source to destination node can be given by

$$\Gamma_{delay} = \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j \neq i}}^{N} t_{ij} \alpha_{ij} \qquad (t_{ij} \geq 0) \tag{1}$$

$$\Psi_{\cos t} = \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j \neq i}}^{N} c_{ij} \alpha_{ij} \qquad (c_{ij} \geq 0) \tag{2}$$

By using the definitions above and assuming the maximum delay in Ad hoc network to be T_0, the problem of QoS routing in Ad hoc network can be represented as constrained combinatorial optimization problem as follows

$$\begin{aligned} &\text{minimize} \qquad \Psi_{\cos t} = \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j \neq i}}^{N} c_{ij} \alpha_{ij} \\ &\text{subject to} \qquad \Gamma_{delay} = \sum_{i=1}^{N} \left(\sum_{\substack{j=1 \\ j \neq i}}^{N} t_{ij} \alpha_{ij} - \sum_{\substack{j=1 \\ j \neq i}}^{N} t_{ij} \alpha_{ji} \right) \leq T_0 \qquad \alpha_{ij} \in \{0,1\} \end{aligned} \tag{3}$$

To formulate this optimization problem in terms of the HNN, the computational network requires $N \times (N-1)$ neurons since it is organized in an $(N \times N)$ matrix with all diagonal elements removed. Defining the Lyapunov (energy) function as equation (4), its minimization drives each neuron of the neural network into its stable state, thereby providing the routing algorithm solution:

In equation (4), the μ_1 term minimizes the total link cost $\Psi_{\cos t}$ of a routing path by taking into account the cost of all existing links; the μ_2 term prevents nonexistent links from being included in the chosen routing path; the μ_3 term oversees constraint satisfaction and maintains the equality of the number of incoming and outgoing links; the μ_4 term drives the neurons towards convergence to a valid route consisting of connected nodes; the μ_5 term was proposed to guarantee that the flow vector is always directed towards the destination; the μ_6 and μ_7 terms are added to eliminate the possibility of loops or partitions in a path. The μ_8 term is presented from the constraint to minimize the delay Γ_{delay}. These terms also nudge the neurons towards speedy and precise convergence to an optimal solution.

$$E=\frac{\mu_1}{2}\sum_{i=1}^{N}\sum_{\substack{j=1\\ j\neq i}}^{N}c_{ij}\cdot\alpha_{ij}+\frac{\mu_2}{2}\sum_{i=1}^{N}\sum_{\substack{j=1\\ j\neq i}}^{N}\beta_{ij}\cdot\alpha_{ij}+\frac{\mu_3}{2}\sum_{i=1}^{N}\left(\sum_{\substack{j=1\\ j\neq i}}^{N}\alpha_{ij}-\sum_{\substack{j=1\\ j\neq i}}^{N}\alpha_{ij}-\chi_i\right)$$
$$+\frac{\mu_4}{2}\sum_{i=1}^{N}\sum_{\substack{j=1\\ j\neq i}}^{N}\alpha_{ij}\cdot(1-\alpha_{ij})+\frac{\mu_5}{2}\sum_{i=1}^{N}\sum_{\substack{j=1\\ j\neq i}}^{N}\alpha_{ij}\cdot\alpha_{ji}+\frac{\mu_6}{2}\left(\sum_{i=1}^{N}\sum_{\substack{j=1\\ j\neq i}}^{N}\left(\sum_{\substack{k=1\\ k\neq i,j}}^{N}\alpha_{ik}-1\right)\cdot\alpha_{ij}^2\right)$$
$$+\frac{\mu_7}{2}\left(\sum_{i=1}^{N}\sum_{\substack{j=1\\ j\neq i}}^{N}\left(\sum_{\substack{k=1\\ k\neq i,j}}^{N}\alpha_{kj}-1\right)\cdot\alpha_{ij}^2\right)+\frac{\mu_8}{2}\left(T_0-\sum_{i=1}^{N}\sum_{\substack{j=1\\ j\neq i}}^{N}t_{ij}\cdot\alpha_{ij}\right)\cdot$$
$$\left(\left(T_0-\sum_{i=1}^{N}\sum_{\substack{j=1\\ j\neq i}}^{N}t_{ij}\cdot\alpha_{ij}\right)-\left|T_0-\sum_{i=1}^{N}\sum_{\substack{j=1\\ j\neq i}}^{N}t_{ij}\cdot\alpha_{ij}\right|\right) \tag{4}$$

where

$$\chi_i=\begin{cases}1 & \text{if } i=s\\ -1 & \text{if } i=d\\ 0 & \text{if } i\neq s,d\end{cases}$$

The motion equation of the HNN is obtained:

$$\frac{dy_{ij}}{dt}=-\frac{y_{ij}}{\tau}+\sum_{k=1}^{N}\sum_{\substack{l=1\\ l\neq k}}^{N}\left(\omega_{ij,kl}^{1}\alpha_{kl}+\omega_{ij,kl}^{2}\alpha_{kl}\alpha_{ij}\right)+h_{ij} \tag{5}$$

where τ is the time constant of the circuit, and the connection weights ω and the biases h are given as:

$$\omega_{ij,kl}^{1}=\mu_3\delta_{jk}+\mu_3\delta_{li}-\mu_3\delta_{ik}-\mu_3\delta_{jl}+\mu_4\delta_{ik}\delta_{jl}-\frac{\mu_5}{2}\delta_{jk}\delta_{il}$$
$$-\mu_6\delta_{ik}\delta_{jl}-\mu_7\delta_{ik}\delta_{jl}-\mu_8\delta_{ik}\delta_{jl} \tag{6}$$
$$\omega_{ij,kl}^{2}=-\mu_6\delta_{ik}-\mu_7\delta_{ljl}\quad h_{ij}=-\frac{\mu_1}{2}c_{ij}-\frac{\mu_2}{2}\beta_{ij}+\mu_3\chi_i-\mu_3\chi_j-\frac{\mu_4}{2}$$

where δ_{ij} is the Kronecker delta function. The HNN finds the optimal Qos routing between two nodes by properly adjusting the input biases in (6).

3 Performance Evaluation

The simulation program is written in C++. The simulation is used to compare the performance of the proposed algorithm with those of Ali and Kamoun [5], Park and

Choi [8] and Ahn [6] for a 14-node network. Fig. 2 compares the route costs achieved by the algorithms, in which Dijkstra's algorithm provides a reference of optimal result. Clearly, the proposed algorithm exhibits fastest convergence. In Fig.2 we can observe total cost in [6] drops below Dijkstra's optimal cost in course of iterations, while the proposed algorithm can approximate steadily to Dijkstra's optimal algorithm.

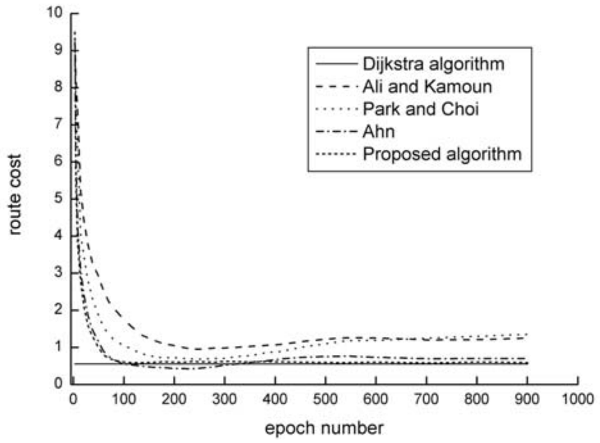


Fig. 2. Convergence properties of various algorithms

A total of 10000 random network topologies with varying sizes and randomly assigned link costs were investigated. Table 1 compares the algorithms. The proposed algorithm converges to a stable state in about 380 iterations, about 25% improvement over the algorithms in [5][8] and about 5% improvement in [6].

Table 1. Performance comparison of four algorithms

Algorithm / Performance	Ali and Kamoun	Park	Ahn	Proposed
Average number of iterations	507.32	505.84	398.86	381.53
Standard deviation of iterations	255.86	282.42	234.62	230.25
Route optimality (%)	84.95	85.53	93.69	95.21

Furthermore, it attains about 95% of route optimality performance, about 10%, 11% and 2% improvement over those algorithms. In Table 1, the route optimality refers to the percentage of the correct optimal routes found by the proposed approach in the random network topologies tested in the experiment.

4 Conclusions

In this paper we have proposed a HNN based QoS routing algorithm in mobile Ad hoc networks under minimum cost constraint. This work shows the capability of HNN as a computational tool for solving a constrained optimization problem. It is suitable for multi-hop radio networks. It utilizes different kinds of information (available with peripheral neurons) and exploits the highly correlated knowledge (available at the local neuron).

Simulation results show that the proposed QoS routing algorithm is insensitive to variations in network topologies. Convergence to a stable state was seen to occur in about 380 iterations that is an improvement by about 25% over both Ali & Kamoun and Park & Choi algorithms, and 5% over Ahn. It also achieves up to 95% route optimality performance, about 10%, 11% and 2% improvement.

References

1. Xiao, Y., Li, H.: Local Data Control and Admission Control for Qos Support in Wireless Ad Hoc Networks. IEEE Transactions on Vehicular Technology, **53** (2004) 1558 - 1572
2. Sheng, M., Li, J., Shi, Y.: Routing Protocol with Qos Guarantees for Ad-Hoc Network. Electronics Letters, **39** (2003) 143 – 145
3. Xiao, X., Ni, L.M.: Internet Qos: A Big Picture. IEEE Network, **13** (1999) 8 - 18
4. Hopfield, J.J., Tank, D.W.: Neural Computations of Decision in Optimization Problems. Biological Cybernetics, **52** (1986) 141-152
5. Ali, M.K.M., Kamoun, F.: Neural Networks for Shortest Path Computation and Routing in Computer Networks. IEEE Transactions on Neural Networks, **4** (1993) 941 - 954
6. Ahn, C.W., Ramakrishna, R.S., Kang, C.G., Choi, I.C.: Shortest Path Routing Algorithm Using Hopfield Neural Network. Electronics Letters, **37** (2001) 1176 - 1178
7. Kocak, T., Seeber, J., Terzioglu, H.: Design and Implementation of A Random Neural Network Routing Engine. IEEE Transactions on Neural Networks, **14** (2003) 1128 - 1143
8. Park, D.C., Choi, S.-E.: A Neural Network Based Multi-Destination Routing Algorithm for Communication Network. IEEE International Joint Conference on Neural Networks, **2** (1998) 1673 - 1678

Content Filtering of Decentralized P2P Search System Based on Heterogeneous Neural Networks Ensemble

Xianghua Fu and Boqin Feng

Department of Computer Science and Technology,
Xi'an Jiaotong University, Xi'an, Shaanxi 710049, China
csdnfxh@yahoo.com.cn

Abstract. A Peer-to-Peer (P2P) based decentralized personalized information access system called PeerBridge for edge nodes of the Internet network is proposed to provide user-centered, content-sensitive, and high quality information search and discovery service from Web and P2P network timely. The general system architecture, user modeling and content filtering mechanism of PeerBridge are discussed in detail. Moreover in order to only find information which users are interested in, a new heterogeneous neural network ensemble (HNNE) classifier is presented for filtering irrelevant information, which combines several component neural networks to accomplish the same filtering task, and improves the generalization performance of a classification system. Performance evaluation in the experiments showed that PeerBridge is effective to search relevant information for individual users, and the filtering effect of the HNNE classifier is better than that of support vector machine, Naïve Bayes, and individual neural network.

1 Introduction

Researchers have developed many different techniques to address the challenging problem of locating relevant web information efficiently. The most conventional example is Centralized Search Engine (CES) such as Google (http://www.google.com). But there are some problems of The CES. With the emergence of successful application like Gnutella (http://www.gnutella.com/), Kazaa (http://www.kazza.com), Freenet (http://freenetproject.org/), P2P technology has received significant visibility over the past few years. There are a few projects such as Apoidea [1], Coopeer [2], ODISSEA [3] attempt to build a P2P based Web search or crawling system.

In this paper, we present a P2P based decentralized Web search system called PeerBridge, which is developed based on our focused crawling system called WebBridge [4]. In PeerBridge, each node only search and store a part of the Web model that the user is interested in, and the nodes interact in a peer-to-peer fashion in order to create a real distributed search engine. To avoid get irrelevant information, A heterogeneous neural network ensemble (HNNE) [5] classifier is used as a content filter to model the peer's preference and filter irrelevant information, enhance the quality of the retrieval search results.

2 Design Overview of Decentralized P2P Search System

PeerBridge have five main components: a content filter which makes relevance judgments on pages crawled from Web and query results searched from other nodes, a distiller which determines a measure of centrality of crawled pages to determine visit priorities, a crawler with dynamically reconfigurable priority controls which is governed by the content filter and distiller, a P2P infrastructure which supports to construct a P2P overlay network with other nodes to share and search information each others, and an user interface with which user can edit training samples, select category taxonomy to training classifier, and query information from the personalized data resource base and other nodes. DHT based distributed lookup and information-exchange protocols [6] are used to exchange vital information between the peers. A block diagram of the general architecture of PeerBridge is shown in Fig. 1.

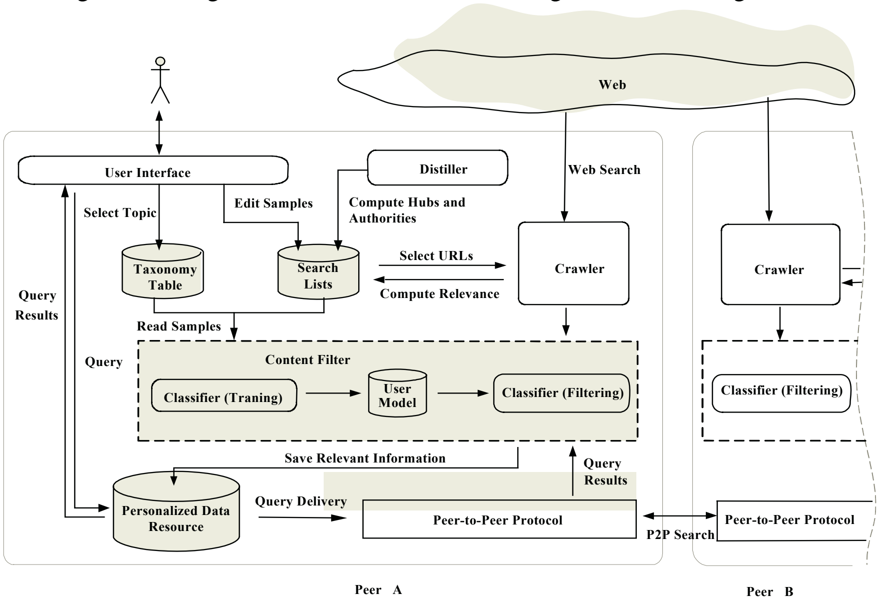


Fig. 1. The general architecture of PeerBridge

3 Adaptive Content Filtering Model

An information filtering system can use intelligent content analysis to automatically classify documents. Such methods include k-nearest neighbor classification, linear least square fit, linear discriminant analysis, and naïve Bayes probabilistic classification [7]. We use artificial neural networks because they are robust enough to fit a wide range of distributions accurately and can model any high-degree exponential models. Neural networks are chosen also for computational reasons since, once trained, they operate very fast. Moreover, such a learning and adaptation process can give semantic meaning to context-dependent words.

3.1 User Model

To filter information for specific users according to their preference and interests, user model is created as an image of what users need. We define a user model as:

$$\mathbf{UM} := (\mathbf{UMID},\mathbf{FD},\mathbf{FT},\mathbf{UI},\mathbf{UIV}) \tag{1}$$

where **UMID** is an user model identifier, **FD** := $\{d_1, d_2, \ldots ,d_N\}$ is a set of sample documents, **FT** := $\{t_1, t_2, \ldots, t_M\}$ is a lexicon comprise all feature terms of **FD**, **UI :=** $\{u_1,u_2,\ldots,u_T\}$ is a set of interests specified by users, and **UIV** := $\{\mathbf{UIV}_1, \mathbf{UIV}_2,\ldots, \mathbf{UIV}_T\}$ is a set of interest vectors of a special user, of which every element responds to a interest u_k $(1 \le k \le T)$, and is defined as $\mathbf{UIV}_k := \langle(t_1,w_{1k}), (t_2,w_{2k}), \ldots , (t_M,w_{Mk})\rangle$, where w_{ik} is the frequency of term t_i $(1 \le i \le M)$ in $\mathbf{UIV}_k$.

According vector space model (VSM), FD constitutes a term by document matrix $X := (d_1,d_2,\ldots,d_N)$, where a column $d_j:=\langle(t_1,x_{1j}),(t_2,x_{2j}),\ldots, (t_M,x_{Mj})\rangle$ is a document vector of the document d_j and every element x_{ij} is the frequency of the term t_i in document d_j. TDFIF frequency is used, which is defined as:

$$x_{ij} = tf_{ij} \cdot \log(N/df_i) \tag{2}$$

where tf_{ij} is the number of the term t_i that occurs in the document d_j, and df_i is the number of documents where the word t_i occurs. The similarity between document vectors is defined as:

$$\mathrm{Sim}(d_i,d_j)= \mathbf{d}_i^T\mathbf{d}_j = \frac{\sum_{k=1}^{N} x_{ki}x_{kj}}{\sqrt{\sum_{k=1}^{N} x_{ki}^2 \sum_{k=1}^{N} x_{kj}^2}}\,. \tag{3}$$

Equation (3) also can be used to compute the similarity between document vector and interest vector.

3.2 Neural Networks-Based Content Filtering

The Neural networks contain an input layer, with as many elements as there are feature terms needed to describe the documents to be classified as well as a middle layer, which organizes the training document set so that an individual processing element represents each input vector. Finally, they have an output layer also called a summation layer, which has as many processing elements there are interests of user to be recognized. Each element in this layer is combined via processing elements within the middle layer, which relate to the same class and prepare that category for output.

In our content filter, the numerical input obtained from each document is a vector containing the frequency of appearance of terms. Owing to the possible appearance of thousands of terms, the dimension of the vectors can be reduced by singular value decomposition (SVD), Principal Component Analysis, Information Entropy Loss, and word frequency threshold [7], etc.

3.3 Heterogeneous Neural Networks Ensemble Classifier

Neural Network ensemble (NNE) is a learning paradigm where many neural networks are jointly used to solve a problem [8]. It originates from Hansen and Salamon's work [9], which shows that the generalization performance of a neural network system can be significantly improved through combining several individual networks on the same task. There has been much work in training NN ensembles. However, all these methods are used to change weights in an ensemble. The structure of the ensemble, e.g., the number of NNs in the ensemble, and the structure of individual NNs, e.g., the number of hidden nodes, are all designed manually and fixed during the training process. In literature [6], we propose a new method to construct heterogeneous neural network ensemble (HNNE) with negative correlation. It combines ensemble's architecture design with cooperative training of individual NNs in an ensemble. It determines automatically not only the number of NNs in an ensemble, but also the number of hidden nodes in individual NNs. It uses incremental training based on negative correlation learning in training individual NNs. The main advantage of negative learning is that it encourages different individual NNs to learn different aspects of the training data so that the ensemble can learn the whole training data better. It does not require any manual division of the training data to produce different training sets for different individual NNs in an ensemble.

4 Performance Evaluation

Based on PeerBridge we have evaluated the filtering performance of the Chinese Web pages content filter with variant number of component neural network in Web search task. In our experiments six different heterogeneous neural network ensembles are tested, the numbers of component neural network of which are respectively 1,5,10,15,20,25, and are notated as NNE1, NNE5, NNE10, NNE15, NNE20 and NNE25. With above different content filters trained by the same interest documents, PeerBridge search relevant web documents from Yahoo China (http://cn.yahoo.com). The evaluation results are shown in Fig. 2.

In addition we test the heterogeneous neural network ensemble classifier (NNE20 is used) to compare with individual neural network (NNE1 is used), Support Vector Machine (SVM) and Naïve Bayes [10] classifier. The most frequently used Reuters-21578 collection (http://www.research.att.com/~lewis/reuters21578.html) is used in this experiment. The measurement $F1$ is used to evaluate the performance. The experiment results are shown in Fig. 3.

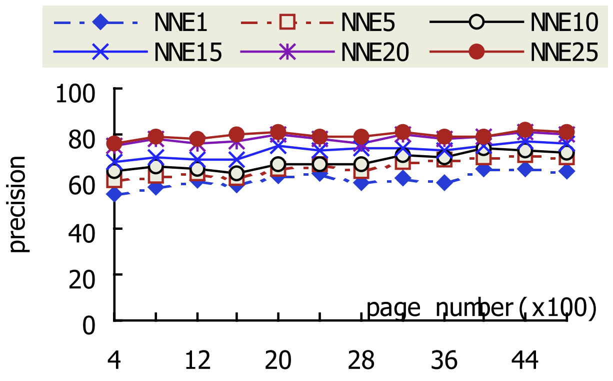


Fig. 2. Precision of content filter with different number of component neural networks

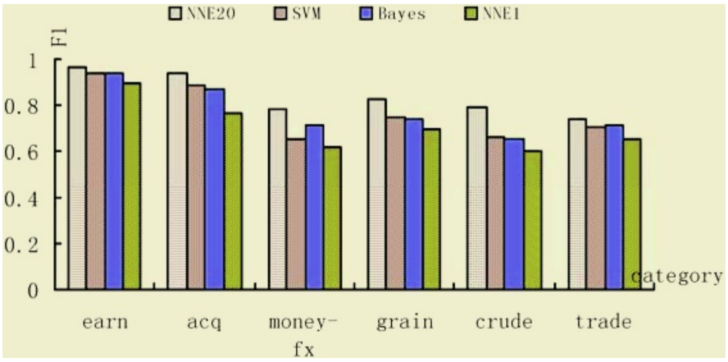


Fig. 3. Comparison with NNE20, SVM, Bayes, NNE1 in Reuters-21578 collection

(1) Fig.2 manifested combining many component neural networks improved the content filtering precision of the Web search system. It is also obviously that increasing the number of the component neural network can improve the precision largely at the beginning, but when the number is sufficiently large, the improvement became small.
(2) Fig.3 showed that the heterogeneous neural network ensemble based classification algorithm was better than other classification algorithm. Once trained, neural network ensemble operates very fast. Moreover, the assumptions on the problem's distribution model of neural network classifier are much less than that of Naïve Bayes classifier, so it is has less independence on the problem, and they are robust enough to fit a wide range of distributions accurately and can model any high-degree exponential models.

5 Conclusions

Facing to the information overload on the Web and the prevalence of the Peer-to-Peer based information sharing, we provide a new pull-based, content-sensitive, interest-related and adaptive information discovery tool, which integrate all of the edge nodes of the Internet network to search useful information from Web and other peers for individual user. To guarantee each node only to search personalized relevant information, HNNE classifier is presented to filter irrelevant content. We compare it with other classifiers such as SVM, Bayes, and individual artificial neural network. The experiment result showed it is efficient and feasible. Although we have developed the system as a prototype, there are still many issues in PeerBridge such as efficiently information search, fault tolerance, and access control etc al in peer-to-peer network left unresolved, which we'll take into account in our future work.

References

1. Singh, A., Srivatsa, M., Liu, L., Miller, T.: Apoidea: a Decentralized Peer-to-Peer Architecture for Crawling the World Wide Web. In SIGIR 2003 Workshop on Distributed Information Retrieval (2003)

2. Zhou, J., Li, K., Tang, L.: Towards a Fully Distributed P2P Web Search Engine. In 10th IEEE International Workshop on Future Trends of Distributed Computing System (2004)
3. Suel, T., Mathur, C., Wu, J.-W., Zhang, J.: ODISSEA: A Peer-to-Peer Architecture for Scalable Web Search and Information Retrieval. In 6th International Workshop on the Web and Databases (2003)
4. Fu, X.H., Feng, B.Q., Ma, Z.F., Ming, H.: Focused Crawling Method with Online-Incremental Adaptive Learning. Journal of Xi'an Jiaotong University, **38** (2004) 599-602
5. Fu, X.H., Feng, B.Q., Ma, Z.F., Ming, H.: Method of Incremental Construction of Heterogeneous Neural Network Ensemble with Negative Correlation. Journal of Xi'an Jiaotong University, **38** (2004) 796-799
6. Stoica, I., Morris, R., Karger, D., Kaashoek, M.F., Balakrishnan, H.: Chord: A Scalable Peer-To-Peer Lookup Service for Internet Application. In SIGCOMM Annual Conference on Data Communication (2001)
7. Sebastiani, F.: Machine Learning in Automated Text Categorization. ACM Computing Surveys, **34** (2002) 1-47
8. Zhou, Z.H., Wu, J.-X., Tang, W.: Ensembling Neural Networks: Many Could Be Better Than All. Artificial Intelligence, **137** (2002) 239-263
9. Hansen, L.K., Salamon, P.: Neural Network Ensembles. IEEE Transaction on Pattern Analysis and Machine Intelligence, **12** (1990) 993-1001
10. Joachims, T.: Text Categorization with Support Vector Machines: Learning with Many Relevant Features. In 10th European Conference on Machine Learning (1998)

Collaborative Filtering Based on Neural Networks Using Similarity

Eunju Kim, Myungwon Kim, and Joungwoo Ryu

School of Computing, Soongsil University, 1-1, Sangdo 5-Dong, Dongjak-Gu, Seoul, Korea
mkim@comp.ssu.ac.kr, blue7786@naver.com, ryu0914@orgio.net

Abstract. Recommendation is to offer information which fits user's interests and tastes to provide better services and to reduce information overload. It recently draws attention upon Internet users and information providers. Collaborative filtering is one of the widely used methods for recommendation. It recommends an item to a user based on the reference users' preferences for the target item or the target user's preferences for the reference items. In this paper, we propose a neural network based collaborative filtering method. Our method builds a model by learning correlation between users or items using a multi-layer perceptron. We also investigate integration of diverse information to solve the sparsity problem and selecting the reference users or items based on similarity to improve performance. We finally demonstrate that our method outperforms the existing methods through experiments using the EachMovie data.

1 Introduction

In this paper, we propose collaborative filtering based on neural network. In general, it has several advantages over the conventional models. Some important features of a neural network include: (1) it can learn a complex relationship between input and output values; (2) it can easily integrate diverse information; (3) it can handle various data types; (4) it can handle incomplete information efficiently. These distinguishing features can well fit to problems such as collaborative filtering. In our method, a multi-layer perceptron is adopted as the basic neural network architecture. A multi-layer perceptron is trained to learn a correlation among preferences of the target user and the reference users. The resulting model is called a user model. The same principle can be applied to build an item model.

Our neural network model has several advantages over the existing methods. First of all, it is expected that its performance is improved because it can learn a complex relationship of preferences among users or items. In addition, the hidden nodes of the model represent latent concepts for recommendation and it causes performance improvement. Next, it is easy to handle diverse data types including continuous numeric data, binary or logical data, and categorical data. Finally, it is easy to integrate diverse kinds of information such as contents and demographic information for efficient filtering. We investigate integration of contents information into the user model to examine the possibility of solving the sparsity problem. In training a neural network it is important to use good features as input data. For this we also investigate selection of the reference users or items based on the similarity between the target user and the reference users or the target item and the reference items.

2 Collaborative Filtering Based on Neural Network Using Similarity

Among many different types of neural networks we adopt as the basic neural network model the multi-layer perceptron (MLP) [3], which is commonly used in various applications. In this section we describe our method of building collaborative recommendation models based on the MLP. We also describe integration of additional information and selection of the reference data to improve the performance of our model.

There are two neural network models for collaborative filtering (CFNN): a user model called U-CFNN and an item model called I-CFNN. In the U-CFNN model the input nodes correspond to the reference users' preferences and the output node corresponds to the target user's preference for the target item. In the I-CFNN model the input nodes correspond to the target user's preferences for the reference items and the output node corresponds to his preference for the target item. An I-CFNN model can be built in a similar way to that described in the above and it is basically of the same structure as that of the U-CFNN model. It differs from the U-CFNN model in that it represents preference association among items. Given an item for recommendation its rating is estimated based on ratings for other items rated by the target user.

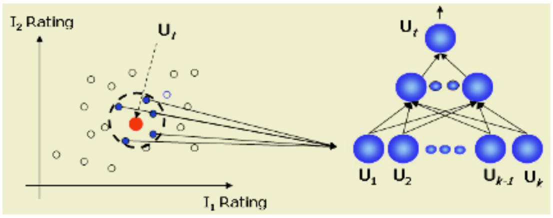


Fig. 1. U-CFNNS

In this paper we investigate collaborative filtering based on neural network using similarity (CFNNS). The CFNNS uses similarity in selecting the reference users or items. In the CFNN model there is no restriction on selecting the reference users or items. However, for the U-CFNN model if we select those users who are similar to the target user in preference, the model can learn stronger correlation among users than random selection of the reference users and consequently, the performance can be improved. Fig. 1 illustrates a U-CFNN model with the selected reference users as input users. Similarly, we can build an I-CFNN model with the selected reference items as input. In this paper we use Pearson's correlation coefficient as similarity measure.

2.1 Information Integration

In this paper we investigate integration of additional information into CFNN to solve the sparsity problem. We consider integrating content information of items into the U-CFNN model. For example, we can integrate content information such as the genre of movies into the U-CFNN model. Also we can integrate into the I-CFNN model user's demographic information such as age, gender, job, and hobbies.

An MLP for integrating additional information can be built simply by adding new input nodes corresponding to the additional information and connecting them to hidden nodes. Our method takes advantage that it is very easy to integrate different kinds of information using neural networks. However, we describe later an experiment with integrating the genre information into the U-CFNN model. We do not consider integrating the demographic information into the I-CFNN model because of lacking useful demographic information in the data.

3 Related Works

The *k*-NN method, which was used in GroupLens for the first time [4], is a memory-based collaborative filtering. In the *k*-NN method, a subset of users are chosen based on their similarities to the target user in preference and a weighted combination of their ratings is used to produce a prediction for the target user. The *k*-NN method is simple and easy to use, however, its performance is generally low. The reason for this is that the *k*-NN algorithm assumes that the attributes are independent. Consequently, it may not be efficient if independence among attributes is not guaranteed. It also has the scalability problem that computation time increases as the size of data increases [5].

Association rules represent the relations between properties of data in the form of 'IF *A* Then *B*.' In association rules we can consider that the correlations (weights) between users or items are represented in terms of support and confidence of rules. However, more detail correlation may not be represented by support or confidence. In addition the performance is low because they rely on a simple statistics of co-occurrences of data patterns without considering intrinsic semantic structure in the data. It is also difficult to represent complex (not logical) relationships among data only in rules of attribute-value pairs.

4 Experiments

We experimented with our method over the domain of movie recommendation. We used the EachMovie dataset [7]. It consists of 72,916 users, 1,628 movies, and 2,811,983 numeric ratings. Movies are rated by six different levels between 0.0 and 1.0 for the lowest preference (or "dislike"), the highest preference (or "like"), respectively. In our research we chose the first 1000 users who rated more than 100 movies.

For our method, the ratings were transformed for efficient training as shown in Table 2. We also represented the missing ratings by 0.0. In training the MLP we only considered the target ratings greater than 0.7 or less than 0.3 and other ratings were ignored. We also transformed the ratings greater than 0.7 into 1.0 and the ratings less than 0.3 into 0.0 as the target output for the MLP.

Table 1. Rating Encoding

User Rating	0.0	0.2	0.4	0.6	0.8	1.0	missing
CFNN Encoding	-1.0	-0.6	-0.2	0.2	0.6	1.0	0.0

For its generalization power we also fixed the number of hidden nodes to be small, such as five. In our experiments we used the learning rate 0.05 and the momentum constant 0.0.

We use four different measures for performance evaluation: accuracy, precision, recall, and F-measure as defined in [8]. For performance evaluation using those measures we quantized the output ratings of our model greater than or equal to 0.5 to “like” and less than 0.5 to “dislike”, respectively.

In this section, we analyzed the performance of our method for recommendation and compared it with the existing methods in various aspects. First we evaluated the performance for integrating the genre information into the U-CFNNS model. Finally, we compared CFNN with the *k*-NN and the association rule methods.

4.1 Information Integration

It is often the case that integrating diverse relevant information or combining multiple methods yields better performance. This is also true with recommendation. [1] and [6] demonstrated performance improvement by integrating additional information and combining different filtering methods. Especially, such integration can solve the sparsity problem in collaborative filtering. In this paper we describe integration of diverse information for performance improvement and solving the sparsity problem using a neural network. Demographic information in the EachMovie data includes gender, age, residential area, and job. However, we decided to ignore it. It is because such demographic information is of little use: residential area and job are little correlated with movie preference while gender and age appear to be significantly biased in the data.

Instead, we integrated the genre of movies into the U-CFNN model as well correlated to user’s preference. The EachMovie dataset has 10 different genres of movies and each movie can have more than one genre specified. Although each genre is represented in binary, when we trained the model we transformed it by multiplying the target user’s rating for the movie and used the result as input to the neural network model. We experimented with the 10 users under the same conditions as described previously.

Table 2 compares the performances of U-CFNNS with and without the genre information. In the experiment we examined the effect of the genre information by varying the number of the reference users. As we can see in the table when the number of the reference users is smaller, the performance improvement is larger. Integration of diverse relevant information will help recommending items correctly to a new user who only has a limited number of reference users. This result demonstrates that integration of relevant information can solve the sparsity problem in collaborative filtering and our neural network model proves to be easy to integrate such information.

Table 2. Performance Comparison of U-CFNNS and U-CFNNS with Genre (%)

	No. of Reference Users (U-CFNNS)					
	10		50		100	
	User	User/ Genre	User	User/ Genre	User	User/ Genre
Accuracy	77.3	82.8	80.5	84.2	83.6	84.1
Precision	75.2	81.4	80.3	83.4	82.6	84.0
Recall	85.9	87.8	82.8	87.0	85.6	85.2
F-measure	79.6	83.4	81.4	84.4	84.1	83.7

4.2 Performance Comparison of CFNN and the Existing Methods

In this section, we compare our collaborative filtering method with the existing methods. For comparison we used the first 1000 users in the EachMovie dataset, who rated more than 100 movies as the training data, and the first 100 users whose user IDs are greater than 70,000 and who rated more than 100 movies as the test data. Using the data we built 30 user and item models. Especially we selected 30 target users randomly among those who have user ID over 70,000. Two thirds of 30 models are trained using data of unbiased preferences and the rest are trained using data of a little biased preferences.

Tables 3 compare in performance our method with the existing methods such as *k*-NN and association rule methods. As shown in the tables the CFNN and CFNNS models show a significant improvement of performance compared with the existing methods.

Table 3. Perfomance Comparison of the User, Movie model and the Existing Methods (%)

	k-NN		Assoc. Rule		CFNNS		
	User	Movie	User	Movie	User	Movie	User/Genre
Accuracy	67.8	64.7	72.0	61.1	87.2	88.1	88.1
Precision	60.3	67.8	75.1	75.4	86.6	88.5	88.5
Recall	55.7	59.8	58.4	22.6	82.8	88.3	88.3
F-measure	57.9	62.4	65.7	34.	83.3	85.7	85.7

5 Conclusions

In this paper, we propose a collaborative filtering method based on neural network called CFNN. We also propose some methods to improve the performance including integration of additional information and selection of the reference users or items based on similarity. Our model utilizes the advantages of a neural network over other methods. It is powerful to learn a complex relationship among data and it is easy to integrate diverse information. The experiment results prove that our method show a significant improvement of performance compared with the existing methods. One of the weaknesses of our method is that the neural network model is not comprehensible, however, in recommendation the comprehensibility of a model should not be important.

Acknowledgements

This work was supported by grant no. 00013078 from the development of Intelligent and Interactive module under the ministry of Commerce, Industry and energy.

References

1. Pazzani, M.J.: A Framework for Collaborative, Content-Based and Demographic Filtering. Artificial Intelligence Review, **13** (1999) 393-408
2. Cheung, K.W., Kwok, J.T., Law, M.H., Tsui, K.C.: Mining Customer Product Ratings for Personalized Marketing. Decision Support Systems, **35** (2003) 231-243
3. Haykin, S.: Neural Networks: A Comprehensive Foundation. 2nd edition. Prentice-Hall. Inc (1999)
4. Konstan, J., Miller, B., Maltz, D., Herlocker, J., Gordon, L. And Riedl, J.: GroupLens: Applying Collaborative Filtering to Usenet News. Communications of the ACM, **40**. (1997) 77-87
5. Sarwar, B.M., Karypis, G., Konstan, J.A., and Ried, J.: Analysis of Recommendation Algorithms for E-Commerce. Proceedings of the ACM EC'00 Conference. Minneapolis. MN (2000) 158-167
6. Press, W. H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical Recipes in C++. 2nd edition. Cambridge University Press (2002)
7. McJones, P.: Eachmovie Collaborative Filtering Data Set. DEC Systems Research Center http://www.rearchdigital.com/SRC/eachmovie. (1997)
8. Kim, M.W., Kim, E.J., Ryu, J.W.: A Collaborative Recommendation Based on Neural Networks, Springer, LNCS **2973** (2004) 425-430

Using Double-Layer One-Class Classification for Anti-jamming Information Filtering

Qiang Sun[1], Jianhua Li[1], Xinran Liang[2], and Shenghong Li[1]

[1] Department of Electronic Engineering, Shanghai Jiao Tong University
200030, China
{sunqiang,lijh888,shli}@sjtu.edu.cn
[2] UCSD ECE Department, 9500 Gilman Drive, MC 0407
La Jolla, CA 92093-0407, USA
xliang@ucsd.edu

Abstract. One-class classification, a derivative of newly developed Support Vector Machine (SVM), obtains a spherically shaped boundary around a dataset, and the boundary can be made flexible by using kernel methods. In this paper, a new method is presented to improve the speed and accuracy of one-class classification. This method can be applied to anti-jamming information filtering with the aim of making it more practical. The experimental results show that the algorithm has better performance in general.

1 Introduction

With the rapid growth of online information available, keep people away from the jamming information, such as unsolicited junk e-mail [1], [2], link spam [3], becomes an important research area in network information technology. Information filtering [4] has become one of the key techniques for processing and organizing massive text data. The problems we have encountered in the conventional information filtering problems, such as the definition of the information filter, and the generalization of the method, also appear in anti-jamming information filtering. Some problems become even more prominent. Because in conventional information filtering, we are familiar with what we want to filter and we can find some other congenial to us easily. But in anti-jamming information filtering, the thing has moved towards reverse side. As the techniques for filtering evolve, so do the tactics that help jamming information producer to avoid being filtered as illegitimate ones. Therefore, simple analysis on the content as an unstructured bag-of-word text is not enough.

In this paper, we present a content based filtering model based on One-Class Classification (OCC), which mines the proper text representation of jamming information and constructs the information filter model. Meanwhile, our proposed model implements the concept-based evaluation whether the information item is of interest or not.

2 Review of One-Class Classification

One-class classification problem can be described as that we develop an algorithm which returns a function that takes the value +1 in a “simple” subset of input space, and -1 elsewhere. [5] proposed a method for solving one-class classification problem: One searches for the sphere that can include all the training data such that the sphere has minimum volume.

In order to describe related previous work, assuming $x_1 \cdots x_n$ are i.i.d. random variables in a set X with unknown distribution P. Suppose the data points are mapped from input space to the high dimensional feature space through a non-linear transformation function Φ. The goal of learning is to look for the smallest sphere that encloses the image of the data.

[6] solved the problem by the following way: Consider the smallest enclosing sphere which characterized by center **a** and radius R >0, the optimization problem is to minimize the error function:

$$\min F(R,a) = R^2 \tag{1}$$

with the constraints:

$$\left\|\Phi(x_i) - a\right\|^2 \le R^2 \ \forall i \tag{2}$$

$\|\bullet\|$ is the Euclidean norm, to allow the possibility of outliers in the training set. Soft margin constraints are incorporated by adding slack variables ξ_i, so transform the original optimization problem:

$$\min\ F(R,a)=R^2 + C\sum_i \xi_i \tag{3}$$

with

$$\left\|\Phi(x_i) - a\right\|^2 \le R^2 + \xi_i \quad \xi_i \ge 0 \quad \forall i \tag{4}$$

By the introduction of Lagrangian Multiplier, the constrained problem was:

$$\begin{aligned} L(R,a,\alpha_i,\gamma_i,\xi_i) &= R^2 + C\sum_i \xi_i \\ &-\sum_i \alpha_i \{R^2 + \xi_i - (\left\|\Phi(x_i)\right\|^2 - 2a\cdot\Phi(x_i) + \left\|a\right\|^2)\} - \sum_i \gamma_i \xi_i \end{aligned} \tag{5}$$

α_i *and* $\gamma_i \ge 0$ are Lagrangian Multipliers, C is a constant that controls the trade-off between the volume and the errors. L should be minimized with respect to R,a, ξ_i and maximized with respect to α_i *and* γ_i. Set the partial derivatives to zero, then one can get:

$$L = \sum_i \alpha_i (K(x_i,x_i)) - \sum_{i\,j} \alpha_i \alpha_j (K(x_i,x_j)) \tag{6}$$

when

$$0 \le \alpha_i \le C \tag{7}$$

The non- linear transformation function Φ is a feature map. Φ can be computed by evaluating some simple kernel. Define $K(x,y) \equiv \Phi(x)\cdot\Phi(y)$.

A test object z is accepted when this distance is smaller or equal than the radius:

$$\|\phi(z-a)\|^2 = K(z,z) - 2\sum_i \alpha_i K(z,x_i) + \sum_{ij} \alpha_i \alpha_j K(x_i,x_j) \leq R^2 \tag{8}$$

By definition, R^2 is the distance from the center of the sphere **a** to the boundary.

3 Double-Layer One-Class Classification Model

The model of anti-jamming information filtering is shown in Fig1. On the basis of present method of keyword matching shown in the left of Fig1, a new module named OCC module (Right part of Fig. 1) is added. This module is used to distinguish true positive texts from negative texts (jamming information). High recall and high speed of filtering system can be preserved in the keyword matching module, and high precision can be ensured by the OCC module. The double-layer system needs different information filters for each layer, but both information filters is acquired by unbalanced train set without enough negative sample.

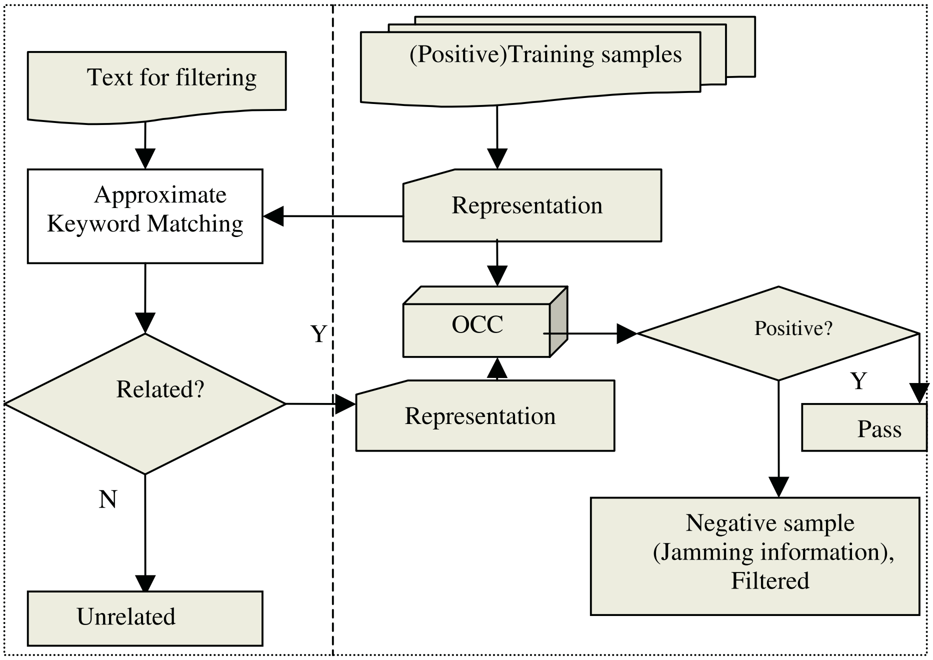

Text for filtering
(Positive)Training samples
Approximate Keyword Matching
Representation
OCC
Positive?
Y
Related?
Representation
Y
Pass
N
Negative sample (Jamming information), Filtered
Unrelated

Fig. 1. Double-layer one-class classification model

Two key points of the model are the approximate keyword matching and the text representation. Analysis of the corpus used to train proves our assumptions: positive text and negative text has the similar keyword representation but different keyword combination, at the same time, the jamming information will change its presentation form to disturb the filtering.

In our system, texts for filtering are processed by keyword matching module firstly. If more than N (N was decided by training result) key-words are found coexisting in the text, the text is considered as a related one. Then, any text filtered by keyword matching module is fed into the OCC module. These related texts will be classified by OCC module. If a related text is identified as a negative one by the classifier of OCC module, the text will be finally filtered out.

4 Performance Evaluation

4.1 Experiment Design

In order to evaluate our double-layer OCC model, we conduct the simulated experiments which are based on the training samples. The training samples are collected from two parts: first part is collected from four weeks' scroll news of sohu's website (http://news.sohu.com/news_scrollnews.shtml); the second part is collected from some junk email related to business and finance. The part of business and financial scroll news of sohu is about 4700 that we regard as positive samples. All other parts of scroll news about 34800 belong to unrelated samples. The junk email, about 120, is the negative part.

70 percent of positive samples make up the training set; left part and negative samples belong to testing set. Cross validation technique is applied to prevent overtraining. We define the following measures to evaluate double-layer OCC performance.

Recall of Related, choosing different values of keywords number, then by the ratio: classified as a related object from related class/ object from related class.

Precision of Related, choosing different values of keywords number, then by the ratio: classified as a related object from related class/ classified as a related object.

Recall of Negative and Precision of Negative are similar to the above items.

4.2 Experiment Result and Analysis

Firstly, we make simulated related decision experiment based on the testing-dataset of related and unrated samples. We transform the Chinese sentences into the Pinyin representation, and transform the Keywords into the Pinyin representation, adjust the approximate keyword matching threshold (the tolerant error number). The experimental results are shown in Fig. 2. As can be seen, when we increase the threshold, recall does the decrease as the increase in precision. In this layer, we need hold a high recall, so we choose threshold to 7 for the next layer.

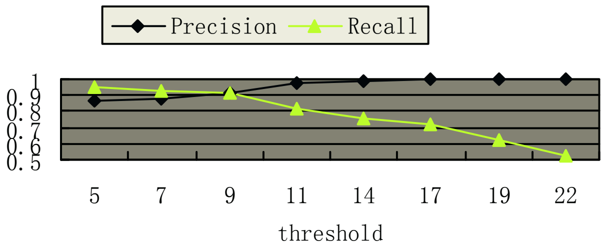


Fig. 2. Related decision experiment

Secondly, we make simulated positive or negative experiment. There are four common kernels mentioned in OCC as in SVM: linear, polynomial, radial basis function (RBF), and sigmoid. RBF is our choice for the recommendation from [7]. $K(x_i, x_j) = \exp(\frac{-\|xi - xj\|^2}{s^2})$. When choose different s, the experimental results are shown in Fig. 3. With the increasing of S, the number of support vectors decrease. As more and more negative samples was classified as positive, recall decreases. And after some level, precision also decreases. We should choose suitable s on the basis of specific application. For example, junk e-mail filter need high precision of negative samples. News filter, on the other hand, prefer a high recall of negative samples.

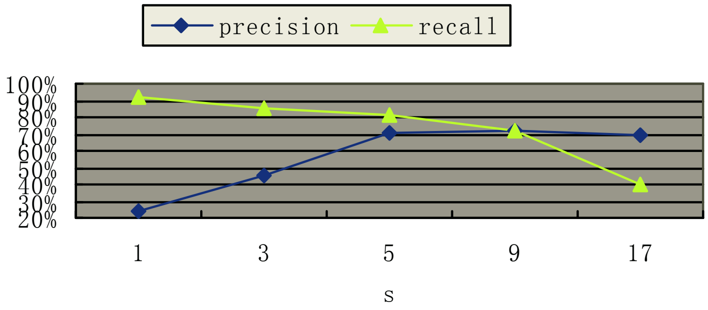


Fig. 3. Positive and negative experiment

5 Conclusion

Information filtering provides useful information and improves web service, which is implemented successfully in many related applications. In this paper, we present a double-layer one-class classification model. This model mines feature representation between positive and negative samples. It constructs the double-layer representation. Based on double-layer representation, one-class classification model makes filter decisions. This double-layer model gains high recall bye first layer and high precision by second layer. The simulated experiments show that our proposed model has better filter performance.

After simple adjustment, this model can be easily extended to some relative domains such as junk e-mail filtering, information recommender system, link spam detection, etc.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grant No.60402019; the National High-Tech Research and Development Plan of China under Grant No.2003AA142160; the National High-Tech Research and Development Plan of China under Grant No.2002AA145090

References

1. Ozgur, L., Gungor, T., Gurgen, F.: Adaptive Anti-Spam Filtering for Agglutinative Languages: A Special Case for Turkish. Pattern Recognition Letters, **25** (2004) 1819-31
2. Kadoya, Y., Fuketa, M., Atlam, E., Morita, K., Kashiji, S., Aoe, J.: An Efficient E-Mail Filtering Using Time Priority Measurement. Information Sciences, **166** (2004) 213-29
3. Gyongyi, Z., Garcia-Molina, H., Pedersen, J.: Combating Web Spam With Trustrank. International Conference on Very Large Data Bases, (2004)
4. Macskassy, Sofus Attila.: New Techniques in Intelligent Information Filtering. The State University of New Jersey (2003)
5. Tax, D.: One-Class Classification. (2001)
6. Tax, D. and Robert Duin, R.P.W.: Support Vector Data Description. Machine Learning, **54** (2004) 45-66
7. Chang, C. and Lin, C.: Libsvm.: A Library for Support Vector Machines. (2001)

Remote OS Fingerprinting Using BP Neural Network

Wenwei Li[1], Dafang Zhang[1], and Jinmin Yang[2]

[1] College of Computer and Communication, Hunan University, Changsha 410082, China
{liww,dfzhang}@hnu.cn
[2] School of Software, Hunan University, Changsha 410082, China
rj_yjm@hnu.cn

Abstract. Remote OS fingerprinting is valuable in areas such as network security, Internet modeling, and end-to-end application design, etc. While current rule-based tools fail to detect the OS of remote host with high accuracy, for users may modify their TCP/IP parameters or employ stack "scrubbers". In this paper, a BP neural network based classifier is proposed for accurately fingerprinting the OS of remote host. To avoid the shortages of traditional BP algorithm, the classifier is also enforced with Levenberg-Marquardt algorithm. Experimental results on packet traces collected at an access link of a website show that, rule-based tools can't identify as many as 10.6% of the hosts. While the BP neural network based classifier is far more accurate, it can successfully identify about 97.8% hosts in the experiment.

1 Introduction

In recent years, the need for automated Internet vulnerability assessment software has been understood and has resulted in the very fast growth of widely available solutions. Remote Operating Systems detection, a.k.a. OS Fingerprinting, is an essential part of the assessment process. OS fingerprinting is the process of determining the identity of the Operating System of a remote host on the Internet. This may be accomplished passively by sniffing network packets traveling between hosts, or actively by sending carefully crafted packets to the target machine and analyzing the response, it leverages the fact that different operating systems implement differing TCP/IP stacks, each of which has a unique signature. Even between versions or patches of an operating system there exist subtle differences as developers include new features and optimize performance [13].

Robust and practicable OS Fingerprinting must meet some requirements. At first, it must be accurate, i.e. it does not fingerprint the OS falsely; secondly, it must be quickly for allowing large network scans; furthermore, it also need that the signature database can be extended easily. To meet these requirements, the design of the classifier in the OS fingerprinting tools plays an important role.

Several TCP/IP fingerprinting tools exist employing both active and passive techniques. The most widely used active tool is the nmap [10] program. Nmap fingerprint OS using a database of over 450 signatures. P0f [2] is a typical rule-based passive tool containing approximately 200 signatures. The signatures contain many fields in TCP/IP stacks, such as: IP time to live (TTL), the IP don't fragment (DF) bit, etc. Each signature maps distinctive fields of the TCP/IP packet header to different operating systems. For each observed packet to a certain remote host, these tools search the signature space for a match. They are rule-based. However, rule-based tools mainly rely on an exact match from an exhaustive list of TCP/IP settings, which make it less accurate and inefficient. For example, previous rule-based approaches fail to identify as many as 10.6% of the hosts in traces we collect from the access link of a website, likely due to users modifying their TCP parameters or employing stack "scrubbers" [3].

The neural network is an efficient and widely used approach for solving classification problems [1],[4]. It also has been used in many areas such as: decision support [5],[7],[8], automatic control [11],[12],[14], and image process [15], etc. In this paper, we designed an OS fingerprinting classifier based on BP neural network. While, the traditional BP algorithm has the disadvantages that, it converges slowly, and it falls into the local minimum point easily. So we enforced our classifier with Levenberg-Marquardt (LM) algorithm. The experimental results show that, the BP neural network based classifier is far more accurate and quicker than previous rule-based approaches; it can provide a continuous degree of identification confidence without deep-packet inspection.

While OS fingerprinting is often regarded as a security attack, we argue that our work is motivated by a number of positive and practical applications. Remotely determining operating systems is valuable in intrusion detection systems [6], serving operating system specific content, providing security against unauthorized access, compiling network inventories, building representative Internet models and measuring NAT (Network Address Translation) deployment which has important implications [9] to the Internet address registries, developers of end-to-end applications and designers of next generation protocols.

The remainder of the paper is organized as follows: Section 2 describes the design of the BP neural network based classifier. In Section 3 we give our measurement results. We conclude the paper in section 4.

2 BP Neural Network Based Classifier

2.1 Structure of the Classifier

The classifier is composed of data acquiring module, pre-process module, BP neural network module and signature database, the structure of it is shown in Fig.1. The data-acquiring module obtains original network packets using active or passive method. The pre-process module picks up signature fields from the packet header and normalizes these signature fields to avoid large signature field value difference having negative effect on network training and classifying. The signature database has 213

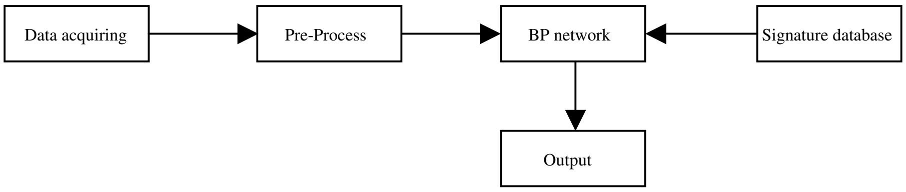


Fig. 1. Structure of the BP network based classifier

signatures, which are the packet signatures used in p0f. The BP neural network module is the pivotal module of the classifier, after training it using the signature database, we can use it to classify and fingerprint the OS of remote hosts.

2.2 Implementation of the BP Network Module

The BP network used in the classifier has three-layer neural network structure, i.e. input layer, hidden layer and output layer. BP algorithm is a supervised learning method. It uses mean squared error and gradient descent method to fix the network weight with the goal of obtaining minimum mean squared error between the real outputs of the network and prescriptive outputs. Assuming there are *p* learning samples inputted, denoting the corresponding supervising signal as t_{pi}, and real output as y_{pi}, then the error of t_{pi} and y_{pi} is

$$E = \frac{1}{2} \sum_{p} \sum_{i} (t_{pi} - y_{pi})^2 \tag{1}$$

Denoting the weight of any two neurons as W_{sq}, then we have

$$\Delta W_{sq} = -\sum_{i=1}^{p} \eta \frac{\partial E}{\partial W_{sq}} \tag{2}$$

Where η is the incremental amplitude of learning rate.

If noticed that $\Delta E<0$, then the error can decay till $\Delta E=0$. The function values of the gradient descent method used in traditional BP algorithm descend fast at a few initial iterating steps, however when it is more close to the minimum, it descend more slowly, therefore, the learning rate is slow. To overcome this shortage of traditional BP algorithm, we adopt Levenberg-Marquardt (LM) algorithm to improve it, for shorting the learning time.

The LM algorithm is designed for made training convergent fast, and it avoid calculating the Hessian matrix. When the evaluation function has the form of square sum, the Hessian matrix is approximated as

$$H = J^T J \tag{3}$$

Then the gradient is

$$g = J^T e \tag{4}$$

Where *J* is the Jacobian matrix, *e* is the network error vector.

The Jacobian matrix can be calculated using standard BP algorithm, and it is simpler to calculate the Jacobian matrix than the Hessian matrix. The LM algorithm uses an approximate matrix of Hessian matrix, then the weight adjusting rate is

$$\Delta W = [J^T J + \mu I]^{-1} J^T e \tag{5}$$

Where *I* is the identity matrix.

The whole LM algorithm used in the BP network classifier is given as follows, here we denote the evaluation function as $E(a) = \sum_{i=1}^{n} f_i^2(a)$:

```
The LM algorithm used in the BP network classifier
1. $\mu \leftarrow 10^{-3}$, $a(0) = a_0$.
2. Calculate $E(a)$.
3. Calculate Jacobian matrix $J = [\frac{\partial f}{\partial a_i}]$.
4. $a(k+1) = a(k) - [J^T J + \mu I]^{-1} J^T f(a(k))$.
5. Calculate $E(a(k+1))$.
6. If $E(a(k+1))>E(a(k))$, then $\mu \leftarrow \mu \times 10$, goto (4).
7. If $E(a(k+1))<$ target error, then success, end the algorithm.
8. $\mu \leftarrow \mu \times 0.1$.
9. $a(k)=a(k+1)$, goto (2) .
```

3 Experiments Results

This section gives the results of our classifier on Internet traces. We evaluate the BP network based classifier and rule-based tools (p0f) with a one-day long packet trace into the website of Hunan university, for the website record the OS of the remote host which access it in its web logs.

There are totally 9563 distinct hosts in the web log. Figure 2 shows the OS distribution among these hosts in web log (denoted WL) and which fingerprinted using the BP network based classifier (denoted BPN) and p0f (denoted Rule-Based) from a one-day long packet trace. The OS distribution fingerprinted using BP network based classifier is more similar with which among hosts in web log, when comparing with rule-based tools. Table 1 displays the correct classification rate (CCR), false accept rate (FAR), and false reject rate (FRR) of our classifier and rule-based tools. The BP network based classifier has higher CCR than rule-based tools, with a difference of

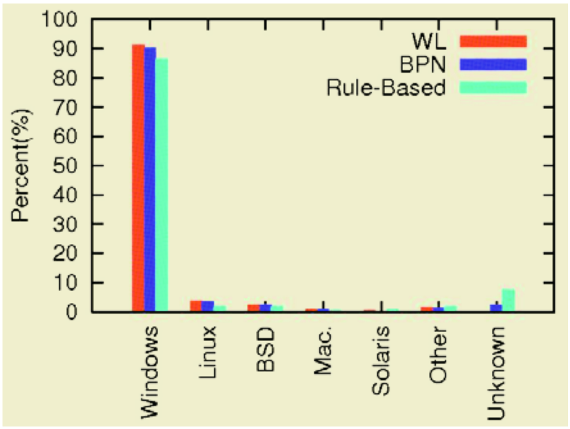


Fig. 2. The OS Distribution among hosts in web log (denoted WL) and which fingerprinted using the BP network based classifier (denoted BPN) and p0f(denoted Rule-Based).

8.4%. The FAR and FRR of our classifier are also less than rule-based tools, for FAR, the difference is 2.0%, and for FRR, 6.4%. These results indicate that the proposed BP network based classifier is more accurate than the rule-based tools in fingerprinting remote hosts' OS.

Table 1. The CCR,FAR, and FRR of BPN classifier and rule-based tools

	CCR(%)	FAR(%)	FRR(%)
BPN	97.8	0.9	1.3
Rule-Based	89.4	2.9	7.7

In addition, we also use our classifier to analyze approximately 618M packets from a three hour-long trace collected at the inbound access point of our campus network. This trace contains 78694 hosts. The results are similar with figure 1, for space limitation, we do not present them anymore. In a word, although the BP neural network based classifier still has about 0.9% false accept rate and 1.3% false reject rate when used to do OS fingerprinting, while compare with rule-based tools, it is far more accurate.

4 Conclusions

In this paper, we have proposed a BP neural network based classifier used in remote OS fingerprinting, for previous rule-based tools are inaccurate. To overcome the shortage of traditional BP algorithm, the LM algorithm was adopted to shorten the BP network training time. Experimental results demonstrate the accuracy of the classifier. Comparing with former rule-based tools, the correct classification rate of our classifier is 8.4% higher, which reaches 97.8%. However there still have few hosts that our classifier misclassified or couldn't classify at all. As a future work, we will collect more OS signatures, and train our classifier using these signatures, to improve the correct classification rate of our classifier.

Acknowledgement

This work was supported by National Natural Science Foundation of China under the contract 60273070.

References

1. Cao, Y., Liao, X., Li, Y.: An E-mail Filtering Approach Using Neural Network. In: Lecture Notes in Computer Science, Vol.3174. Springer-Verlag, Berlin Heidelberg New York (2004) 688-694
2. Zalewski, M.: Passive OS Fingerprinting Tool (2003) http://lcamtuf.coredump.cx/p0f.shtml.
3. Smart, M., Malan, G.R., Jahanian, F.: Defeating TCP/IP Stack Fingerprinting. In: Proc. of the 9th USENIX Security Symposium (2000)
4. Yang, S., Yi, Z.: Self-Organizing Feature Map Based Data Mining. In: Lecture Notes in Computer Science, Vol.3174. Springer-Verlag, Berlin Heidelberg New York (2004) 193-198

5. Yang, Y., Cao, J., Zhu, D.: A Study of Portfolio Investment Decision Method Based on Neural Network. In: Lecture Notes in Computer Science, Vol.3174. Springer-Verlag, Berlin Heidelberg New York (2004) 976-981
6. Taleck, G.: Ambiguity Resolution via Passive OS Fingerprinting. In: Proc. 6th International Symposium on Recent Advances in Intrusion Detection (2003)
7. Guo, G., Kuh, A.: An Optimal Neural-Network Model for Learning Posterior Probability Functions from Observations. In: Lecture Notes in Computer Science, Vol.3173. Springer-Verlag, Berlin Heidelberg New York (2004) 370-376
8. Zeng, Z., Huang, D., Wang, Z.: Stability Analysis of Discrete-Time Cellular Neural Networks. In: Lecture Notes in Computer Science, Vol.3173. Springer-Verlag, Berlin Heidelberg New York (2004) 114-119
9. Senie, D.: Network Address Translator (NAT)-friendly Application Design Guidelines. RFC 3235, Internet Engineering Task Force (2002)
10. Fyodor: Active remote OS fingerprinting (1998) http://www.insecure.org/nmap.
11. Ye, M., Yi, Z.: On the Discrete Time Dynamics of the MCA Neural Networks. In: Lecture Notes in Computer Science, Vol.3173. Springer-Verlag, Berlin Heidelberg New York (2004) 815-821
12. Sun, F., Zhang, H., Wu, H.: Neuro-Fuzzy Hybrid Position/Force Control for a Space Robot with Flexible Dual-Arms. In: Lecture Notes in Computer Science, Vol.3174. Springer-Verlag, Berlin Heidelberg New York (2004) 13-18
13. Paxson, V.: Automated packet trace analysis of TCP implementations. In: SIGCOMM (1997) 167-179
14. Sun, C., Li, X., Feng, C.B.: On Robust Periodicity of Delayed Dynamical Systems with Time-varying Parameters. In: Lecture Notes in Computer Science, Vol.3173. Springer-Verlag, Berlin Heidelberg New York (2004) 32-37
15. Lu, W., Lu, H., Shen, R.: Color Image Watermarking Based on Neural Networks. In: Lecture Notes in Computer Science, Vol.3174. Springer-Verlag, Berlin Heidelberg New York (2004) 651-656

Emotional Learning Based Intelligent Traffic Control of ATM Networks

Mahdi Jalili-Kharaajoo[1], Mohammadreza Sadri[2], and Farzad Habibipour Roudsari[2]

[1] Young Researchers Club, Islamic Azad University, Tehran, Iran
mahdijalili@ece.ut.ac.ir
[2] Iran Telecommunication Research Center, Tehran, Iran

Abstract. In this paper, an intelligent controller is applied to traffic control of ATM networks. First, the dynamics of the network is modeled by a Locally Linear Neurofuzzy Models. Then, an intelligent controller based on brain emotional learning algorithm is applied to the identified model. Simulation results show that the proposed fuzzy traffic controller can outperform the traditional Usage Parameter Control mechanisms.

1 Introduction

Asynchronous Transfer Mode (ATM) is a new technology to support wide variety of services including Constant Bit Rate (CBR), Variable Bit Rate (VBR), Unspecified Bit Rate (UBR) and Available Bit Rate (ABR). The traffic control includes traffic parameters and the requested QoS. Traffic parameters such as peak cell rate, maximum burst size and minimum cell rate are used to describe the inherent characteristics of a traffic source. At the network side, Connection Admission Control (CAC) [1],[2] is responsible to describe the acceptance or rejection of the new requested connection.

A cognitively based version of reinforcement learning has been developed in which a critic constantly assesses the consequences of actuating the plant with the selected control action in any given state in terms of the overall objectives or performance measures and produces an analog reinforcement cue which in turn directs the learning in the controller block. This cognitive version of the reinforcement signal has been denoted as an emotional cue, for it is indeed the function of emotions like stress, concern, fear, satisfaction, happiness, etc. to assess the environmental conditions with respect to goals and utilities and to provide cues regulating action selection mechanisms. Whether called emotional control or merely an analog version of reinforcement learning with critic (evaluative control), the method is increasingly being utilized by control engineers, robotic designers and decision support systems developers and yielding excellent results.

In this paper, an intelligent controller will be applied to the traffic control problem in an ATM network. To this end, the dynamic behavior of the networks is identified using Locally Linear Model Tree (LoLiMoT) algorithm for training of neurofuzzy network [3],[4] and then Brain Emotional Learning Based Intelligent Controller (BELBIC) is applied to the plant. We have used BELBIC [5], the recently developed neuromorphic controller based on emotional learning model to produce the control action. Simulation result of the controller with show the effectiveness of the proposed control action.

2 Local Linear Neuro-fuzzy Modeling

The network structure of a local linear neuro fuzzy model can be followed in [4]. Each neuron realizes a local linear model (LLM) and an associated validity function that determines the region of validity of the LLM. The network output is calculated as a weighted sum of the outputs of the local linear models, where the validity function is interpreted as the operating point dependent weighting factors. The validity functions are typically chosen as normalized Gaussians. LOLIMOT is an incremental tree-construction algorithm that partitions the input space by axis-orthogonal splits [4]. In each iteration, a new rule or local linear model is added to the model. In each iteration of the algorithm the validity functions that correspond to the actual partitioning of the input space are computed, and the corresponding rule consequence are optimized by a local weighted least squares technique

3 Structure of BELBIC

Motivated by the success in functional modeling of emotions in control engineering applications [6], [7], a network model have been adopted which is developed by Moren and Balkenius in [6], [7], as a computational model that mimics amygdala, orbitofrontal cortex, thalamus, sensory input cortex and generally, those parts of the brain thought responsible for processing emotions. There are two approaches to intelligent and cognitive control. The model is illustrated in Fig. 1. BELBIC is essentially an action generation mechanism based on sensory inputs and emotional cues. In general, these can be vector valued, although in the benchmark discussed in this paper for the sake of illustration, one sensory input and one emotional signal (stress) have been considered. The emotional learning occurs mainly in amygdala. The learning rule of amygdala is given by

$$\Delta G_a = k_1 . \max\left(0, EC - A\right) \tag{1}$$

where G_a is the gain in amygdala connection, k_1 is the learning step in amygdala and EC and A are the values of emotional cue function and amygdala output at each time. The term max in the formula (1) is for making the learning changes monotonic, implying that the amygdala gain can never be decreased as it is modeled to occur in biological process in amygdala. This rule is for modeling the incapability of unlearning the emotion signal (and consequently, emotional action), previously learned in the amygdala [5],[6]. Similarly, the learning rule in orbitofrontal cortex is shown by

$$\Delta G_o = k_2 .(MO - EC) \tag{2}$$

which is completely based on the original biological process. In the above formula, G_o is the gain in orbitofrontal connection, k_2 is the learning step in orbitofrontal cortex and MO is the output of the whole model, where it can be calculated as

$$MO = A - O \tag{3}$$

in which, O represents the output of orbitofrontal cortex. In fact, by receiving the sensory input S, the model calculates the internal signals of amygdala and orbitofrontal cortex by the relations in (4) and (5) and eventually yields the output.

$$A = G_a.SI \tag{4}$$

$$O = G_o.SI \tag{5}$$

Since amygdala does not have the capability to unlearn any emotional response that it ever learned, inhibition of any inappropriate response is the duty of orbitofrontal cortex.

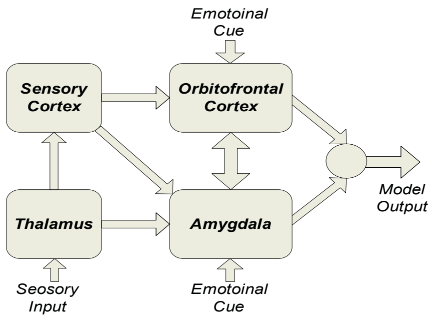


Fig. 1. The abstract structure of the computational model mimicking some parts of mammalian brain

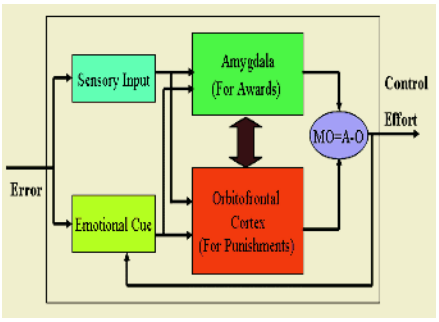


Fig. 2. Control system configuration using BELBIC

Controllers based on emotional learning have shown very good robustness and uncertainty handling properties [5], while being simple and easily implementable. To utilize our version of the Moren-Balkenius model as a controller, it should be noted that it essentially converts two sets of inputs (sensory input and emotional cue) into the decision signal as its output. A closed loop configuration using this block (termed BELBIC) in the feed forward loop of the total system in an appropriate manner have been implemented so that the input signals have the proper interpretations. The block implicitly implemented the critic, the learning algorithm and the action selection mechanism used in functional implementations of emotionally based (or generally reinforcement learning based) controllers, all at the same time [6,7]. The structure of the control circuit we implemented in this study is illustrated in Fig. 2. The implemented functions in emotional cue and sensory input blocks are given in

$$EC = MO.(-W_1.\dot{e}.e + W_2.e) \tag{6}$$

$$SI = W_3.e + W_4.\dot{e} + W_5.\int edt \tag{7}$$

where *EC*, *MO* , *SI* and *e* are emotional cue, controller output, sensory input and output error and the W_1 through W_5 are the gains must tuned for designing a satisfactory controller. In the choice of these two signals some principles are taken into consideration as following steps.

1. Sensory input is a kind of control signal which in BELBIC is reinforced or punished based on Emotional cue so it should be chosen as a function of error just like a PID controller. This choice has some advantages such as existence of a systematic way to tune the gains. In this way one can set the Emotional cue equal to

zero at first and then tune the gains of Sensory input as a simple PID controller and then proceed to tune the gains of the other parts of BELBIC in the direction of improving the performance of primary sensory input signal. This method can solve the main problem of BELBIC which was the tuning of the gains. In addition to this point the controller now has more reliability because of being based on a classic controller (PID). Also PID controller has some other advantages such as robustness to some extend which is very desirable especially in this work with possible uncertainties in estimated model including less than enough neurons. Besides, using this signal selection it does not need to concern on effect of noises on identification. So an identification using less numbers of neurons can easily filter the noises while could be used in tuning of controller and it will accelerate the online control process certainly.

2. When the Emotional cue is a positive number, the gain of amygdala connection will be increased and when the Emotional cue is a negative number, the gain of orbitofrontal connection will be increased and the bigger Emotional cue causes the bigger reinforcement or punishment so the Emotional cue should increase when the absolute value of error decreases. In order to avoid the offset error the Emotional cue should include error in addition to its derivative but with a very smaller coefficient. Finally the Emotional cue is compared with the control signal (MO) therefore it should have the same dimension with (MO).

4 Simulation Results

In this section, the performance of the proposed controller is evaluated and compared with that of traditional UPC mechanisms including LB, JW and EWMA. Two voice traffic sources, with traffic characteristics shown in the Table, are used to derive the system. The number of cells per burst has a geometric distribution with a mean of *E[x]* and the silence phase has an exponential distribution with a mean of *E[s]*. The traffic parameters are based on ADPCM with a 64 bytes information fields [8]. In both the FC1 and the FC2, the mamdani's interface method and center of gravity difuzzification technique are used [9]. An ATM multiplexer with N independent ON/OFF input voice sources is simulated. The multiplexer is modeled as a finite queue served by a single server with First-In First-Out (FIFO) service discipline. The output link of multiplexer is a T1 (1.544 Mbps) pipe. As mentioned above, the actual mean cell rate of traffic source is estimated by the intelligent controller named BELBIC. All UPC mechanisms are dimensioned to achieve a detection probability less than 10^{-6} at the nominal cell arrival rate. The time window T_o is set to 10Δ. The flexibility factor of EWMA mechanism, γ, is selected as $\gamma = 0.91$. For the traffic source 1, the values of parameters are selected as follows

$$C_{LB} = 1.2, C_{JW} = 1.77, C_{EWMA} = 1.2, N = 1100, Z1 = 0.12(0.06), Z2 = 0.5(0.5).$$

where *N* and *C* represent the counter size and the over-dimensioning factor of UPC mechanism, respectively.

For the traffic source 2, the values of parameters are selected as

$$C_{LB} = 1.45, C_{JW} = 2.5, C_{EWMA} = 1.3, N = 670, Z1 = 0.112(0.07), Z2 = 0.6(0.5).$$

The selectively curves of the proposed controller and the other traditional mechanisms for two traffic sources 1 and 2 are shown in Figs. 3 and 4, respectively. As it can be seen, the proposed controller has a detection probability which is very close to the ideal controller specially BELBIC2. For example, when traffic source1 generates 10% excessive load, the violation probability of BELBIC2 is improved about 3.2,2 and 1.5 order of magnitude in comparison with JB, LB and EWMA, respectively.

Table 1. Traffic characteristics of two voice sources

Traffic characteristics			
Source	b_o(bps)	m_o(bps)	E(X)
1	32000	11200	22
2	64000	22000	58

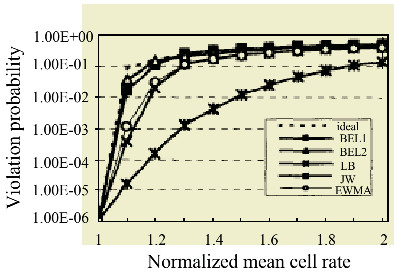


Fig. 3. Selectivity curve for traffic source 1

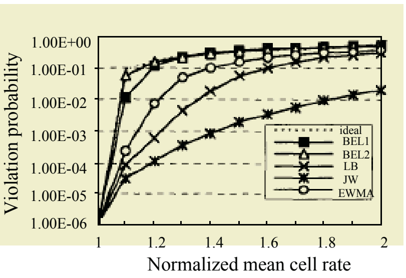


Fig. 4. Selectivity curve for traffic source 2

In Fig. 5, the dynamic behavior of BELBIC is compared with that of LB. In this case, the traffic source 2 generates 50% overload by increasing the mean number of cells during the burst phase. In order to improve the dynamic behavior of LB, the bucket size is set to 300. as shown in Fig. 5, the proposed BELBIC2 starts detecting violation after only 500 and its violation probability grows very fast so that after emitting 1500 cells the violation probability grows slowly.

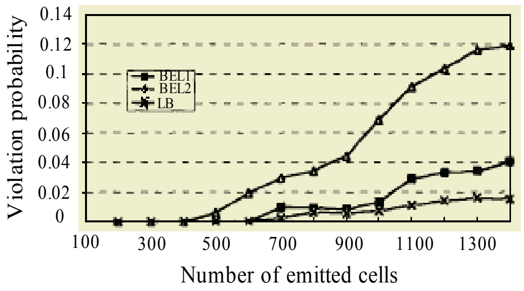


Fig. 5. Dynamic behavior of BELBIC and LB

5 Conclusions

In the paper, a high performance intelligent traffic controller for ATM networks was proposed. The dynamic behavior of the network was identified using a Neurofuzzy network with locally linear training algorithm and then the controller was adjusted in such a manner that its loss load is equal to excessive load generated by the traffic source. Simulation results showed that the proposed emotional based controller demonstrates much better selectivity and effectiveness than the other conventional UPC mechanisms.

References

1. Jaganattan, S., Talluri, J.: Adaptive Predictive Congestion Control of High Speed ATM Networks, IEEE Trans. Brodcasting, **48** (2002) 129-130
2. Tarbonriech, S., Abdallah, C.T., Ariola, M.: Bounded Control of Multiple-Delay Systems with Application to ATM Networks. In Proc 40th IEEE CDC (2001) 2315-2320
3. Nelles, O.: Orthonormal Basis Functions for Nonlinear System Identification with Local Linear Model Trees (LoLiMoT). in Proc. IFAC Symposium on System Identification, Kitakyushu, Fukuoka, Japan (1997)
4. Nelles, O.: Local Linear Model Tree for On-Line Identification of Time Variant Nonlinear Dynamic Systems. in Proc. International Conference on Artificial Neural Networks (ICANN), Bochum, Germany (1996) 115-120.
5. Shahmirzadi, L.D., Sheikholeslami, N.: Introducing BELBIC: Brain Emotional Learning Based Intelligent Controller. International Journal of Intelligent Automation and Soft Computing, 10 (2004)
6. Moren, J., Balkenius, C.: A Computational Model of Emotional Learning in The Amygdala: From animals to animals," in Proc. 6th International conference on the simulation of adaptive behavior, Cambridge, Mass., The MIT Press (2000)
7. Moren, B.J.: A Computational Model of Emotional Conditioning in the Brain. in Proc. Workshop on Grounding Emotions in Adaptive Systems, Zurich (1998)
8. Bensaou, S.T.C. Chu, D.H.K.: Tsang, Estimation of the Cell Loss Ratio in ATM Networks with a Fuzzy System and Application to Measured-Based Call Admission Control. IEEE/ACM Trans. Networking, **5** (1997) 572-584
9. Develekos, D.G.: A fuzzy Logic Approach to Congestion Control in ATM Networks. in Proc. IEEE Int. Conf. Communication, WA, USA (1995) 1969-1973

Multi-agent Congestion Control for High-Speed Networks Using Reinforcement Co-learning

Kaoshing Hwang[1], Mingchang Hsiao[2], Chengshong Wu[1], and Shunwen Tan[2]

[1] National Chung Cheng University, Ming-Hsiung, Chia-Yi 621, Taiwan, China
{hwang,ieecsw}@ccu.edu.tw

[2] WuFeng Institute of Technology, Ming-Hsiung, Chia-Yi 621, Taiwan, China
{mcsau,swtan}@mail.wfc.edu.tw

Abstract. This paper proposes an adaptive reinforcement co-learning method for solving congestion control problems on high-speed networks. Conventional congestion control scheme regulates source rate by monitoring queue length restricted to a predefined threshold. However, the difficulty of obtaining complete statistics on input traffic to a network. As a result, it is not easy to accurately determine the effective thresholds for high-speed networks. We proposed a simple and robust Co-learning Multi-agent Congestion Controller (CMCC), which consists of two subsystems: a long-term policy evaluator and a short-term rate selector incorporated with a co-learning reinforcement signal to solve the problem. The well-trained controllers can adaptively take correct actions to regulate source flow under time-varying environments. Simulation results showed the proposed approach can promote the system utilization and decrease packet losses simultaneously.

1 Introduction

In most networks, there are circumstances in which the externally offered load is large than can be handled. Then, if no measures are taken to restrict the entrance of traffic into the network, the link will be congested. Congestion results from mismatch between network resources available for these connections and traffic capacities admitted for transmission. As a result, as the offered load increases, the actual network throughput decrease and packet loss increase enormously. Therefore, a high-speed network must have an appropriate flow control scheme not only to guarantee Quality of Service (QoS) for existing links but also to achieve high system utilization.

In general, packet losses in high speed network are inevitable owing to the uncertainties and the highly time-varying of different traffic patterns in no control scheme (e.g. best effort service model) [1] or in conventional end-to-end rate control scheme which uses Additive Increase, Multiplicative Decrease (AIMD) [2]. Recently, fuzzy logic and neural networks has been employed in congestion control. Lee et al. [3] proposed a neural-fuzzy mechanism to control source rate. By monitoring the current queue state and the rate of change of the queue length, the future queue behavior is predicted. Cheng et al. [4] proposes a neural fuzzy approach to connection admission control for multimedia high-speed networks.

In this paper, we propose an adaptive congestion control method, called Co-learning Multi-agent Congestion Controller (CMCC), to solve above problem. Each CMCC in the network does not learn independently but cooperatively with positively correlated global rewards, which are co-learning feedback signals. Also, each CMCC has a separate memory structure to explicitly implement its own objectives to achieve co-learning Nash equilibria [5].

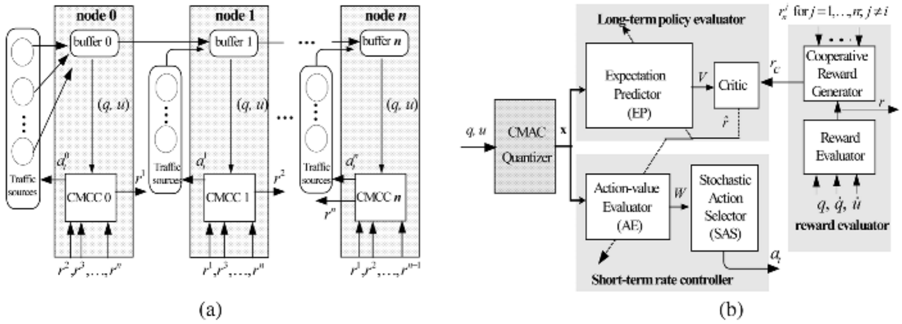

node 0
buffer 0
(q, u)
Traffic sources
CMCC 0
node 1
buffer 1
CMCC 1
node n
buffer n
CMCC n
Long-term policy evaluator
Expectation Predictor (EP)
Critic
CMAC Quantizer
Action-value Evaluator (AE)
Stochastic Action Selector (SAS)
Short-term rate controller
Cooperative Reward Generator
Reward Evaluator
reward evaluator
(a)
(b)

Fig. 1. (a) The proposed CMCC model in n-node case, (b) the schematic structure of CMCC for node i; bold lines indicate the path of signals and thinner lines represent the path of a single scalar.

2 Architecture of CMCC

Figure 1(a) shows the configuration of n nodes, with controllable sources, controlled by CMCC cooperatively. Each CMCC consists of two subsystems: Expectation Predictor (EP) is the long-term policy selector and the short-term rate controller is composed of Action-value Evaluator (AE) and Stochastic Actions Selector (SAS) as shown in Fig. 1(b). The CMCC persistently receives the state inputs decoded by a Cerebellar Model Articulation Control (CMAC) method [6] and a scalar signal called the cooperative reinforcement signal from the cooperative reward generator (CRG). The EP network, by means of the cooperative reinforcement signal (r_c), predicts a heuristic reinforcement signal ($\hat{r}$) to the AE network, which receives not only the heuristic reinforcement signal but also the cooperative reinforcement signal to evaluate the control signal action selected by the SAS element. The specifications of CMCC network variables are given in Table 1.

Topologically, two neural networks jointly implement the structure of the EP and the AE networks. The EP and the AE networks produce an optimal control signal a in response to the status of high-speed networks.

3 Theoretical Framework

In high-speed networks, CMCC in each node acts as a congestion control agent with congestion control ability. Each CMCC has its own state variables (**S**), which is composed of buffer length (q) and sending rate (u) irrespective of other CMCCs in the network. Reinforcement signal r for a specified state is denoted as r = 1 for award, and r = 0 for penalty. The reward is given, when satisfying one of these three rules:

Table 1. The specifications of the variables of CMCC for node i.

q	queue lengths in a node
u	explicit sending rate of sources
s_t	state vector delineated by variables q and u at time t
$\mathbf{x}$	the hybercube selection vector
V	output of EP network
V	expected evaluation value of state vector s_t
W	output of AE network
a_t^i	feedback control signal generated by SAS element
r_n^j	reinforcement scalar signal provided by neighboring node j
r^i	reinforcement signal generated by node i
r_c^i	Cooperative reward $r_c^i = \mu^i r^i(s_t) + \sum_{j \neq i} \mu^j r_n^j(s_{t-1})$
$\hat{r}$	heuristic reinforcement signal $= r_c^i + \gamma V^i(s') - V^i(s)$

Rule 1. $q_L < q_{k+1} < q_H$ and ($\dot{q}_{k+1} > 0$ or $\dot{R}_{k+1} > 0$)

Rule 2. $q_{k+1} < q_L$ and ($\dot{R}_{k+1} > 0$ or $R_{k+1} = R_0$)

Rule 3. $q_{k+1} > q_H$ and ($\dot{R}_{k+1} < 0$ or $R_{k+1} = 0.25R_0$) It is assumed that a CMCC agent in nodes/switches does not know reward functions of the other CMCC agents but can receive their immediate rewards to form cooperative rewards. However, all agents have incomplete information of other agents' reward functions and the state transition probabilities. By way of a multiagent policy-search learning, the proposed CMCC could converge to a Nash equilibrium under certain restrictions to the high-speed systems. The CMCC extends a traditional reinforcement learning method based on Markov decision processes (MDPs) to stochastic games [5], which essentially are the n-agent MDP.

The proposed on-line learning algorithm improves the performance CMCC network over time in two ways. First, each CMCC aims at learning an optimal control policy from its own mistakes through the cooperative reinforcement signal, and then reinforces its action to improve long-term performance. Second, node states associated with the positive reinforcement are memorized through a network learning process so that similar states will be more positively associated with a control action leading to a positive reinforcement. Each policy is guaranteed to be a strict improvement over the previous ones (unless it is already optimal). The AE element maintains a separate estimated value for each state, in order to balance exploration and exploitation in the process of RL. The SAS element determines an action as follows:

$$a_t = f\left[\sum_{i=1}^{N_h} w_t(i) x_t(i) + n_t\right], \qquad (1)$$

where n_t is a random noise and f is a unipolar sigmoid function. The SAS element's outputs $a_t = [0.2,\ 1]$ are the ratio of sending rates. The flowchart of an individual CMCC is shown in Fig. 2.

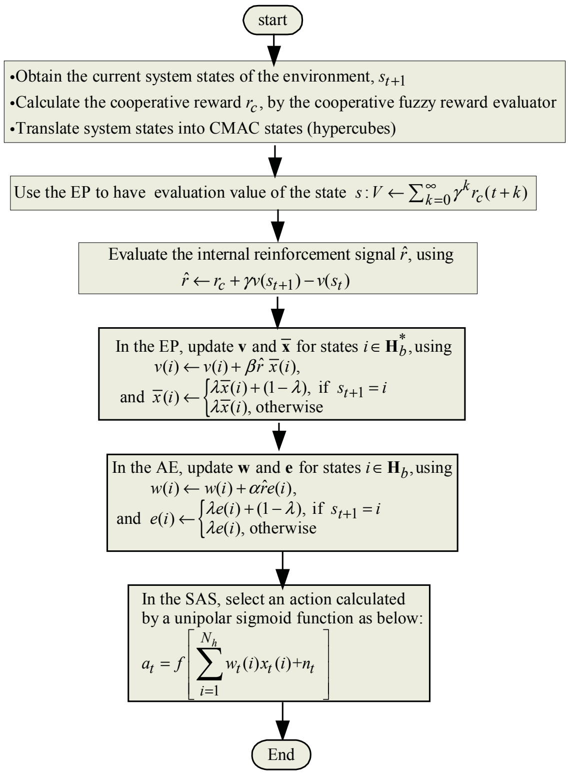

start
•Obtain the current system states of the environment, s_{t+1}
•Calculate the cooperative reward r_c, by the cooperative fuzzy reward evaluator
•Translate system states into CMAC states (hypercubes)
Use the EP to have evaluation value of the state $s: V \leftarrow \sum_{k=0}^{\infty} \gamma^k r_c(t+k)$
Evaluate the internal reinforcement signal $\hat{r}$, using $\hat{r} \leftarrow r_c + \gamma v(s_{t+1}) - v(s_t)$
In the EP, update $\mathbf{v}$ and $\overline{\mathbf{x}}$ for states $i \in \mathbf{H}_b^*$, using $v(i) \leftarrow v(i) + \beta\hat{r}\,\overline{x}(i)$, and $\overline{x}(i) \leftarrow \begin{cases}\lambda\overline{x}(i) + (1-\lambda), \text{ if } s_{t+1} = i \\ \lambda\overline{x}(i), \text{ otherwise}\end{cases}$
In the AE, update $\mathbf{w}$ and $\mathbf{e}$ for states $i \in \mathbf{H}_b$, using $w(i) \leftarrow w(i) + \alpha\hat{r}e(i)$, and $e(i) \leftarrow \begin{cases}\lambda e(i) + (1-\lambda), \text{ if } s_{t+1} = i \\ \lambda e(i), \text{ otherwise}\end{cases}$
In the SAS, select an action calculated by a unipolar sigmoid function as below: $a_t = f\left[\sum_{i=1}^{N_h} w_t(i)x_t(i) + n_t\right]$
End

Fig. 2. The flowchart of an individual CMCC.

4 Simulation and Comparisons

In the simulation, the scheme of congestion control agent could be one of the three schemes: AIMD, Independent Multiagent Congestion Controller (IMCC), and CMCC control. We assume that all packets are fixed length of 100 bytes with Poisson arrival, and adopt a finite buffer length of 20 in each node. Loading is defined as a relative value in the ratio of source offered bandwidth to the available bandwidth. Figure 3 shows the configuration of multi-node controller. Figure 4 shows the throughputs of Sw1 controlled by three different kinds of control agents individually. Because of the reactive control, the throughputs of nodes for the AIMD method decrease seriously at loading of about 0.9. Conversely, the IMCC and CMCC methods remain a higher throughput even though the offered loading is over 1.0; that is, the offered loads of the sources will overwhelm the link capacity without appropriate controls. Hence, it demonstrates that CMCC possesses the ability to predict the system behavior in advance. In addition, Figure 5 shows the Packet Loss Ratio (PLR) of Sw1 implemented by four different kinds of methods. It is obvious that PLR of no control is high, even though we adopt the AIMD method. However, using the CMCC method can decrease the PLR enormously with high throughput and low mean delay. However, low packet losses in CMCC result from cooperatively regulating the sources' sending rates with-

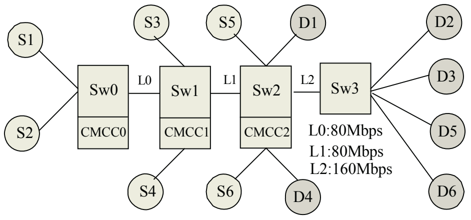


Fig. 3. A high-speed network model with four switches.

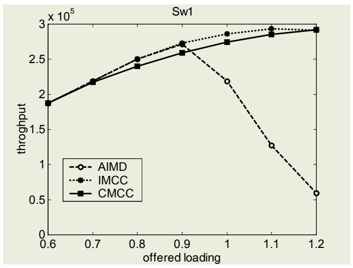


Fig. 4. Comparisons of throughput at Sw1.

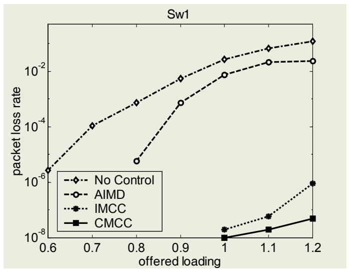


Fig. 5. Comparisons of PLR at Sw1.

out the expense of throughput. Obviously, a low queuing delay is expected as shown in Fig. 6. The reason is that congestion control executed in the precedent nodes can prevent the other node from congestion.

Finally, comparisons of the PMF of distribution of feedback control signal, for IMCC and CMCC at nodes under various offered loading, are shown in Fig. 7. Comprehensibly, CMCC adopts over 90% at a rate of full speed to transmit packets from sources to various destinations, and only a small amount of low rates are used to accommodate traffic to avoid packet loss, so that the throughputs are definitely high. Moreover, the sending rates adopted in CMCC1 are more conservative than those in IMCC1 because IMCC takes an action independently. Meanwhile, as the traffic offered loading is increased, conservative rates are preferred at the expense of throughputs; hence the percentage of higher rate is decreased and that of the lower rate is increased for the purpose of avoiding congestion.

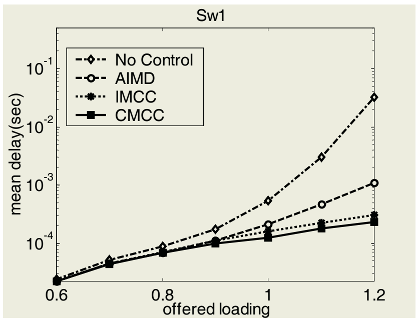


Fig. 6. Comparisons of mean delay at Sw1.

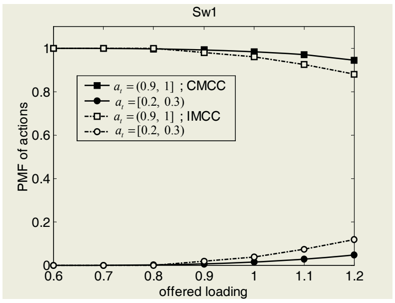


Fig. 7. Comparisons of PMF of actions at Sw1.

5 Conclusions

In modern high-speed networks, most packet losses result from the dropping of packets owing to congested nodes. The reactive congestion scheme AIMD could not accurately respond to a time-varying environment for lack of prediction capability of packet losses. In contrast, the reinforcement learning method can cope with the prediction problems. The proposed CMCC, which is applicable to a network of multi-bottleneck, can respond to networks' dynamics. Through a proper training process and networks' performance evaluated by CRE, CMCC can learn empirically without prior information on the environmental dynamics. Simulation results have shown that the proposed method can increase the utilization of available links' capacity and decrease PLR simultaneously. Therefore, CMCC not only guarantees low PLR for existing links but also achieves high system utilization.

References

1. Gevros, P., Crowcoft, J., Kirstein, P., Bhatti, S.: Congestion Control Mechanisms and the Best Effort Service Model. IEEE Network (2001) 16-26
2. Chiu, D., Jain, R.: Analysis of the Increase and Decrease Algorithm for Congestion Avoidance in Computer Networks. Computer Networks and ISDN Systems (1989) 1-14
3. Lee, S.J. Hou, C.L.: A Neural-Fuzzy System for Congestion Control in ATM Networks. IEEE Trans. on System, Man. and Cybernetics, **30** (2000) 2-9
4. Cheng, R., Chang, C., Lin, L.: A QoS-Provisioning Neural Fuzzy Connection Admission Controller for Multimedia High-Speed Networks. IEEE/ACM Trans. on Networking, **7** (1999) 111-121
5. Sun, R. and Littman, M.L.: Value-function Reinforcement Learning in Markov Games. Journal of Cognitive Systems Research, **2** (2001) 55-66
6. Hwang, K., Lin, C.: Smooth Trajectory Tracking of Three-Link Robot: A self-Organizing CMAC Approach. IEEE Trans. on Systems, Man and Cybernetics, **28** (1998) 680-692

Multi-scale Combination Prediction Model with Least Square Support Vector Machine for Network Traffic

Zunxiong Liu[1,2], Deyun Zhang[1], and Huichuan Liao[2]

[1] School of Electronics and Information Engineering, Xi'an Jiaotong University,
Xi'an, Shaanxi 710049, China
{liuzunx,Dyzhang}@xanet.edu.cn
[2] School of Information Engineering, Huadong Jiaotong University,
Nanchang, Jiangxi 330013, China
lhc@ecjtu.jx.cn

Abstract. A revised multi-scale prediction combination model for network traffic is proposed, where network traffic series are decomposed with stationary wavelet transform, and the different models are built with combinations of wavelet decomposition coefficients. LS-SVM is introduced to predict the coefficients at the expectation point, the prediction value can be obtained by wavelet inversion transform. The simulation experiments with the two traffic traces at different time scale are done with the proposed system, and other predictors. The correlation structure between the prediction point and history data is also explored. The results show that the proposed model improve the computability and achieve a better forecasting accuracy.

1 Introduction

Network traffic prediction is important to network planning, performance evaluation and network management directly. In high-speed network such as ATM, the bandwidth can be allocated based on the accurate traffic forecasting, thus ensuring Qos of the users and accomplishing the preventive congestion control. The traditional linear model oversimplifying the complex temporal correlation presenting in the network traffic [1], artificial neural network(ANN) and its ensembles are applied to network traffic modeling and congestion controlling because of the good capability of nonlinear approximation and learning adaptively [2]. However, neural networks suffer from problems like the existence of local minima and the choice of network structure, so neural networks are limited in such applications as real-time disposition. In order to improve prediction performance in the nonstationary and nonlinear time series, the model combined with wavelet analysis and neural network are also proposed[3], but accompanied with low efficiency due to the inherent flaw from neural networks. After that, Support Vector Machines (SVM), a new learning technique, has been applied in time series prediction due to better generalization ability.

In this paper, the multi-scale combination model based on Least Squares Support Vector Machines(LS-SVM)[4], is proposed to predict one-step head value of network traffic. Our strategies make use of the merits of wavelet analysis and LS-SVM, and approximate the decomposed time subseries at different levels of resolution with coefficient combination models. There wavelet decomposition is implemented with stationary wavelet transform(SWT)[5], an important shift-invariant wavelet transform. The nonstationary and nonlinear signal in time domain is decomposed into the relative stationary wavelet coefficients in frequency domain by way of the wavelet transform, and the property of long-range dependence inherent in the original is not present in the detail coefficients, making it possible that the traditional approaches can be used to approximate the actual correlation structure in the detail coefficients, even the linear models.

The corresponding coefficients at expectation point are forecasted with the use of those models, and the prediction value can be built with wavelet inversion transform. By multi-scale prediction, we mean two facts the traffic data be forecasted at different time scale and that the multi-resolution wavelet analysis be used. We apply the proposed multi-scale model to MPEG video and Ethernet data, and predict the number of bytes per unit time. The other methods, such as autoregression and ANN, are tested in the simulation for comparison.

2 Least Square Support Vector Machine

Least Square Support Vector Machine(LS-SVM) is a new technique for regression[4]. when LS-SVM is used to model network traffic, the input and output variables should be chosen firstly. Given a training data set of *N* points $\{\boldsymbol{x}_k, y_k\}_{k=1}^{N}$ with input data $\boldsymbol{x}_k \in \mathbb{R}^p$ and output data $y_k \in \mathbb{R}$. In order to get the function dependence relation, SVM map the input space into a high-dimension feature space and construct a linear regressor in it. The regression function is expressed with $y = \mathrm{f}(\boldsymbol{x}) = w^T \varphi(\boldsymbol{x})+b$.

In LS-SVM, the model is constructed by solving the following optimal problem:

$$\min_{w,b,e} J_p(\boldsymbol{w},\boldsymbol{e}) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\frac{1}{2}\sum_{i=1}^{N} e_i^2$$
$$\text{s. t. } y_i = \boldsymbol{w}^T\varphi(\boldsymbol{x}_i) + b + e_i, i = 1,\cdots,N \tag{1}$$

with $\varphi(\cdot): \mathbb{R}^n \to \mathbb{R}^{n_h}$ a function which maps the input space into a so-called higher dimensional (possibly infinite dimensional) feature space, weight vector $\boldsymbol{w} \in \mathbb{R}^{n_h}$ in primal weight space, error variables $e_k \in \mathbb{R}$ and bias term b. Note that the cost function J_p consists of a sum squared fitting error(SSE) and a regularization term.

The model should be computed in the dual space, the Lagrangian function can be defined:

$$L(\boldsymbol{w},b,\boldsymbol{e},\boldsymbol{\alpha}) = J_p(\boldsymbol{w},\boldsymbol{e}) - \sum_{i=1}^{N} \alpha_i \left\{\boldsymbol{w}^T\varphi(\boldsymbol{x}_i) + b + e_i - d_i\right\}. \tag{2}$$

where $a_i \in \mathbb{R}$ are Lagrange multipliers. The conditions for optimality are given by

$$\begin{cases} \dfrac{\partial L}{\partial \boldsymbol{w}} = 0 \rightarrow & \boldsymbol{w} = \sum_{i=1}^{N} \alpha_i \varphi(\boldsymbol{x}_i) \\ \dfrac{\partial L}{\partial b} = 0 \rightarrow & \sum_{i=1}^{N} \alpha_i = 0 \\ \dfrac{\partial L}{\partial \boldsymbol{e}_i} = 0 \rightarrow & \alpha_i = Ce_i \qquad\qquad i = 1,\cdots,N \\ \dfrac{\partial L}{\partial \boldsymbol{\alpha}_i} = 0 \rightarrow & \boldsymbol{w}^T \varphi(\boldsymbol{x}_i) + b + e_i - y_i = 0 \end{cases} . \tag{3}$$

After elimination of $\boldsymbol{w}$ and $\boldsymbol{e}$, one obtains the solution

$$\begin{bmatrix} 0 & \vec{\mathbf{1}}^T \\ \vec{\mathbf{1}} & \boldsymbol{K} + C^{-1}\boldsymbol{I} \end{bmatrix} \begin{bmatrix} b \\ \boldsymbol{\alpha} \end{bmatrix} = \begin{bmatrix} 0 \\ \boldsymbol{y} \end{bmatrix} . \tag{4}$$

with $\boldsymbol{y} = [y_1, y_2, \cdots, y_N]^T$, $\alpha = [\alpha_1, \alpha_2, \cdots, \alpha_N]^T$, $\vec{\mathbf{1}} = [1,\cdots 1]^T$, $\boldsymbol{K}_{ij} = \boldsymbol{K}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \varphi(\boldsymbol{x}_i)\ \varphi(\boldsymbol{x}_j)^T$, the equation can be solved by least square. The resulting LS-SVM model becomes

$$y(\boldsymbol{x}) = \sum\nolimits_{k=1}^{N} a_k K(\boldsymbol{x}, \boldsymbol{x}_k) + b \ \cdot \tag{5}$$

where a and a are the solution to equ.4, here RBF kernel $K(\boldsymbol{x}_k, \boldsymbol{x}_l) = \exp(-\|\boldsymbol{x}_k - \boldsymbol{x}_l\|_2^2 / \sigma^2)$

3 Multi-scale Prediction Model

Given the time series $x(k)$, $k = 1, \cdots N$, the aim is to predict the $T - th$ step ahead sample $x(t + T)$ of the series. The working flow for the model run as follow:
Perform SWT on the $x(k)$ to the scale J, getting the approximation coefficients series $a_x(J,)$ and a set of detail series $d_x(j,k)$, $(j = 1, \cdots J)$. The number of resolution level J is empirically determined by way of the inspection of the smoothness of derived scale-coefficient series. Taking the MPEG-4 trace, the medium quality version of Star Wars IV for example, we select the resolution level for wavelet transform as 3, the SWT coefficients series about it are $d_x(1,)$, $d_x(2,)$, $d_x(3,)$ and $a_x(3,)$, respectively.

The four coefficient combination models based on LS-SVM should be built to producing the corresponding SWT coefficients for the expectation point $(t+T)$, where T is the time length of forecast. In Table 1, we present the structure of the used LS-SVM models for above-mentioned trace, among which LS-SVM1, LS-SVM2 and LS-SVM3 combine the SWT coefficients at present point(t) as input variables to produce the detail coefficients. LS-SVM4 for approximation coefficient prediction, its inputs are the p-lagged approximation coefficients, where p is the order of the autoregressive model(AR) that fit the known approximation coefficient.

The expectation value is acquired by performing the inverse stationary wavelet transform on regenerated detail and approximation coefficients.

Table 1. The Structure of LS-SVM Models Used

Models	Input	Output
LS-SVM1	$d(1,t)$, $d(2,t)$, $d(3,t)$	$d(1,t+T)$
LS-SVM2	$d(2,t)$, $d(3,t)$	$d(2,t+T)$
LS-SVM3	$d(2,t)$, $d(3,t)$	$d(3,t+T)$
LS-SVM4	$a(3,t-p)$,... $a(3,t-1)$ $a(3,t)$	$a(3,t+T)$

4 Simulations and Performance

We consider the performance of the proposed system on two network traces containing MPEG-4 video and Ethernet date. The MPEG-4 video trace is the medium quality version of Star Wars IV available at [6]. The Ethernet traffic, collected at Bellcore Morristontown RRESEARCH. All these traffic trace are processed to present the number of byte per unit time, then they are aggregated at different time scales of 1 and 5 seconds. In comparison, the individual LS-SVM predictor, the individual ANN predictor and ANN combination predictor are tested on the same traffic data, where back propogation algorithm is used for ANN, and the inputs for the two individual predictors are the p-lagged normalized traffic data. The ANN combination is produced from the proposed system by replacing LS-SVM with ANN, which structure the same as our proposed system.

In experiment, we use the minimization-maximization normalization method, common in data analysis. As usual, the preprocessed aggregation traffic data and their SWT coefficients are divided into two sets, a training set and a testing sets for each simulation. The training set is used to determine model parameters, and the performance is tested on the testing set.

Note that the lagged traffic data and corresponding coefficients are used for inputs to the models, we apply autoregression (AR) based scheme to decide the order of the autoregressive model(AR). In this scheme, the training set is divided into a modeling and validation set. The modeling set is used to estimate the parameters of a p-th order AR process, where p is the number of lagged values. These parameters are subsequently used to predict the values in the validation set. p is varied and the one giving the best prediction performance is selected as the required lag order p. The p in LS-SVM4(used for approximation coefficient prediction) can be chosen the same as the individual predictor because the approximation coefficients be more smooth and capture the trend of the original signal. The prediction performance measure is the signal-to-noise(SNR) defined as follows:

$$\mathrm{SNR} = 10\log_{10}\left(\frac{\mathrm{E}(\mathrm{x}(\mathrm{n})^2)}{\mathrm{E}((\mathrm{x}(\mathrm{n}) - \hat{\mathrm{x}}(\mathrm{n}))^2)}\right) \text{ dB}. \tag{6}$$

where $E(\cdot)$ is the expected value, $x(n)$ is the actual value, and $\hat{x}(n)$ is the predicted value. As prediction accuracy increases, the SNR becomes larger.

The simulation was with the one-step ahead prediction (T=1), and the performance results in SNR are presented in Tables 2 and 3. In the two tables, the first col-

Table 2. SNR of the predicted MPEG4 video traffic

TS(S)	*P*	AR	Individual Predictor		Combination Predictor	
			ANN	LS-SVM	ANN	LS-SVM
1	8	10.223	10.656	10.995	13.131	14.241
5	3	8.8702	9.4209	9.9324	11.616	14.341

Table 3. SNR of the predicted Ethernet traffic

TS(S)	*p*	AR	Individual Predictor		Combination Predictor	
			ANN	LS-SVM	ANN	LS-SVM
1	13	5.235	5.5828	5.6229	8.9452	10.755
5	7	9.143	9.5996	9.5372	10.004	12.11

umn shows the time scale(TS) in seconds, the second column shows the number of the used lagged value, *p*. The third column shows the result with autoregression for the corresponding data. The forth and fifth column present the results of the individual LS-SVM and ANN predictors. The results for the combination predictors of LS-SVM and ANN are presented in the sixth and seventh columns. The combination predictors with LS-SVM and ANN employ the same network structure, see Table 1. The SWT decomposition scales for the two traffic aggregation data are all set to 3. The ANN models in the experiments are all implemented with the revised back-propagation algorithm, where both the learning rate factor and momentum factor are set to 10^{-4}, The number of training for ANN model is taken 2000 and the transfer functions of the hidden and output neurons are selected as sigmoid function, $f(x) = 1/(1+e^{-x})$.

The approach to determine the number of lagged values for prediction plays an important role in the tests. Moreover there is a key parameter for the LS-SVM models, which is the width for RBF kernel function. The parameter for the individual LS-SVM can be usually selected a relative large value, about between 4 and 10. When it increases, the SNR value increases correspondingly.

According to the above two tables, the performance of the individual models with LS-SVM and ANN are comparable, giving better performance than AR model for the four tests with the two network traffic data. All of the combination predictors have advantage over the individual predictors in accuracy improvement. Especially the LS-SVM combination model outperform the ANN combination in computation efficiency besides prediction accuracy, the results demonstrate that the proposed system be valid. In fig. 1 presents samples of the LS-SVM combination model results in graphical form for the one-step ahead prediction with MPEG4 at time scale 1s.

5 Conclusion

In view of the drawbacks from the conventional methods for network traffic prediction, the muilt-scale combination prediction model with LS-SVM is proposed in the paper, which explores the related theory. The comparative simulation tests were car-

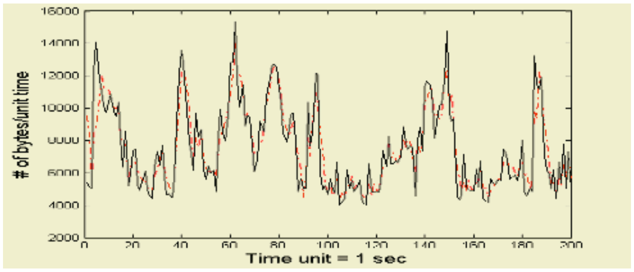


Fig. 1. The actual and predicated traffic for MPEG4 at time scale 1s, the actual values and the predicted values are shown in solid line and dotted lines respectively

ried out with the ethernet and MPEG-4 traffics, demonstrating that the proposed system attains better performance in prediction accuracy and efficiency. The proposed model can play an important role in congestion control in high-speed network, meeting the user Qos requirements.

References

1. Chen, B.S., Peng, S.C., Wang, K.C.: Traffic Modeling, Prediction, and Congestion Control for High-Speed Networks: A Fuzzy AR Approach. IEEE Trans, Fuzzy Systems, 8 (2000) 491-508
2. Khotanzad, A., Sadek, N.: Multi-scale High-Speed Network Traffic Prediction Using Combination of Neural Networks. Proceedings of the International Joint Conference on Neural Networks, **2** (2003) 1071-1075
3. Zhang, B.: Adaptive Neural-Wavelet Model for Short Term Load Forecasting. International Journal of Electric Power Systems Research, **59** (2001) 121-129
4. Suykens, J.A.K., Vandewalle, J.: Least Squares Support Vector Machine Classifiers. Neural Processing Letters, **3** (1999) 293-301
5. Nason, G.P., Silverman, B.W.: The Stationary Wavelet Transform and Some Statistical Applications, In Wavelet and Statistics. Lecture notes in statistics, A. Antoniadis Ed. Springer Verlag, Berlin (1995)
6. Fitzek, F.H.P., Reisslein, M.: MPEG-4 and H.263 Video Traces for Network Performance Evaluation. extended version, Technical Report: TKN-00-06, TU Berlin Department of Electrical Engineering, Telecommunication Networks Group, (2000) http://www-tkn.ee.tu-berlin.de/research/trace/trace.html

Clustering Algorithm Based on Wavelet Neural Network Mobility Prediction in Mobile Ad Hoc Network*

Yanlei Shang, Wei Guo, and Shiduan Cheng

The State Key Lab of Networking and Switching,
Beijing University of Posts and Telecommunications,
Beijing 100876, China
shangyl@bupt.edu.cn

Abstract. In this paper we propose a novel clustering algorithm in mobile ad hoc network. By predicting the mobility of mobile nodes with the wavelet neural network, the resulting clustering algorithm can provide a generic and stable cluster structure for the upper-layer protocols. For this clustering scheme, we give analytical model and performance evaluation.

1 Introduction

A mobile ad hoc network (MANET) is a collection of wireless mobile nodes that dynamically form a network without the need for any pre-existing network infrastructure or central control. In many circumstances, mobile nodes in an ad hoc network are geographically dispersed and multi-hop. They have potential applications in such scenarios as the absence of network infrastructures.

In MANET, the network topology could change rapidly. Network management becomes big challenges. Many researchers have focused their attention on partitioning the ad hoc network into multiple clusters [1]. The clustering algorithms are crucial to ad hoc network. A cluster head should be elected within each cluster to form the upper-layer backbone. With the help of the cluster head, a hierarchical routing protocol can be implemented.

Since the movement is the main cause, we propose a scalable mobility prediction scheme based on the accumulated past behavior. In our work, a novel clustering protocol based on the Wavelet Neural Network [5] mobility Prediction (WNNP) is proposed. The major difference from the previous research is that we design the clustering operation from mobility's point of view, i.e., predicting the nodes mobility by a wavelet neural network. In contrast, most prior work focuses on the algorithm out of regard for management support, lacking an overall cluster stability evaluation. Our goal is to provide a generic and stable cluster structure for the upper-layer protocols.

The remainder of this paper is organized as follows. Section 2 discusses the related work and our motivation. In section 3, we introduce the wavelet neural network prediction model and the clustering algorithm. Performance evaluations are given in section 4. Section 5 concludes the paper.

* This work is supported by the NFSC (No.90204003), the National 863 Project (No. 2003AA121220 and No.2002AA103063), the National 973 project (No.2003CB314806) and The Excellent Young Teachers Program of MOE, P. R. China

2 Related Work and Our Motivation

2.1 Review of Wavelet Neural Network

$\psi(t)$ is a square integrable function, i.e. $\psi(t) \in L^2(R)$. $\Psi(\omega)$ is its corresponding Fourier transform. If

$$\int_{-\infty}^{+\infty} \frac{|\Psi(\omega)|^2}{|\omega|} d\omega < \infty \quad . \tag{1}$$

$\psi(t)$ would be called the wavelet mother function. Make the shift and scale transform on this wavelet mother function, we can get:

$$\psi_{a_k,b_k}(t) = \sqrt{a_k}\psi(a_k t - b_k) \quad a_k \in R^+, \ b_k \in R, \ k \in Z \quad . \tag{2}$$

$\psi_{a_k,b_k}(t)$ is the wavelet basis function depending on the shift factor a_k and the scale factor b_k. We will get a vector family as k varies. If there exist two constants A>0 and B>0 such that for any $g \in L^2(R)$

$$A\|g\|^2 \le \sum_k \left|\left\langle g, \psi_{a_k,b_k} \right\rangle\right|^2 \le B\|g\|^2 \quad g \in L^2(R) \quad . \tag{3}$$

The sequence $\psi_{a_k,b_k}(t)$ is called a frame. A and B will be called the upper and lower limits separately. Function $g(t)$ could be expressed by the linear combination:

$$g(t) = \sum_k w_k \sqrt{a_k}\psi(a_k t - b_k) \quad . \tag{4}$$

The finite sum of the above polynomial could be used in many practical problems to approximate the function $g(t)$ effectively. This equation can also be rewritten by:

$$g(t) = \sum_{k=1}^{N} w_k \psi(a_k(t - b_k)) \quad . \tag{5}$$

It is the wavelet neural network equation. w_k, a_k, b_k are all adaptable parameters. The major difference of the wavelet neural network from the conventional neural network is that the incentive of the hidden layer nodes is the wavelet function rather than the Sigmoid function [5].

2.2 Mobility Prediction

It is important to distinguish between movement history and mobility-model [3]. We will give the definitions firstly.

Definition 1 The *movement history* of a node is defined by a string 'c_1, c_2, c_3…' of symbols. C={c_1,c_2,c_3…c_N} is the set of clusters and ci denotes the Cluster ID (CID) that the mobile node has visited.

Definition 2 The *mobility model* of a mobile node is defined by a function

$$\hat{c}_{N+1} = f(c_1, c_2, c_3, \cdots, c_N) \quad . \tag{6}$$

$c_i (1 \le i \le N)$ is the movement history. $\hat{c}_{N+1}$ is the next cluster the node will visit.

The reason why we could get the future from the past lies in the fact that "history repeats itself." The essence of learning lies in extracting those patterns from history. Learning aids in decision making when reappearance of those patterns are detected.

3 Clustering Algorithm Based on WNNP

3.1 Wavelet Neural Network Mobility Prediction

The wavelet neural network equation can be expressed by:

$$\hat{c}_{N+1} = \sum_{k=1}^{N} w_k \psi\left(a_k \left(C - b_k\right)\right) + \bar{c} \tag{7}$$

Where $C = \left(c_1, c_2, c_3, \cdots, c_N\right)^T$ is the input vector. w_k is weight and $w_k \in R$. The number of the neural cells is N. a_k and b_k are the shift factor and the scale factor separately. $a_k \in R^N$, $b_k \in R^N$. $\bar{c}$ is the estimation of the average of the C.

Suppose there are N sample pairs $\left\{\left(c_i, c_{i+1}\right); c_{i+1} = g\left(c_i\right), i = 1, 2, \cdots, N\right\}$. Let θ denote the vector of w_k, a_k , b_k. Then the equation (9) could be expressed by $g_\theta\left(c_m\right)$. Its objective function is equation (8):

$$E(\theta) = \frac{1}{2} \sum_{i=1}^{N} \left[g_\theta\left(c_i\right) - c_{i+1}\right]^2 \tag{8}$$

To get the optimal value of θ, we minimize the above objective function by the stochastic grads degrade method.

$$\theta(k) = \theta(k-1) - \lambda \nabla\left(\theta(k-1)\right), \quad \nabla E(\theta) = \sum_{i=1}^{N} \left[g_\theta\left(c_i\right) - c_{i+1}\right] \bullet \frac{\partial}{\partial \theta} g_\theta\left(c_i\right) \tag{9}$$

λ is the step size, $\nabla E(\theta)$ is the grad of the objective function $E(\theta)$ about θ. Replacing $g_\theta\left(c_m\right)$ with equation (6), for $k = 1, 2, \cdots, N$ we will get:

$$\begin{cases} \dfrac{\partial E(\theta)}{\partial w_k} = \sum\limits_{i=1}^{N} \left[g_\theta\left(c_i\right) - c_{i+1}\right] \bullet \psi\left(a_k c_i - b_k\right) \\ \dfrac{\partial E(\theta)}{\partial a_k} = \sum\limits_{i=1}^{N} \left[g_\theta\left(c_i\right) - c_{i+1}\right] \bullet w_k \psi'\left(a_k c_i - b_k\right) \bullet c_i \\ \dfrac{\partial E(\theta)}{\partial b_k} = -\sum\limits_{i=1}^{N} \left[g_\theta\left(c_i\right) - c_{i+1}\right] \bullet w_k \psi'\left(a_k c_i - b_k\right) \end{cases} \tag{10}$$

From Equation (10), we would get w_k, a_k, b_k and predict the next cell c_{i+1}.

For lower computing complexity, we implement the WNNP by predicting the horizontal and vertical position separately. WNNP is demonstrated in the Figure 1. In the input, x_i and y_i of the current cluster number i are separated and delayed. The outputs is the predictive results of x_{i+1} and y_{i+1}. The result of the output just is the next cluster number c_{i+1} the mobile node will visit.

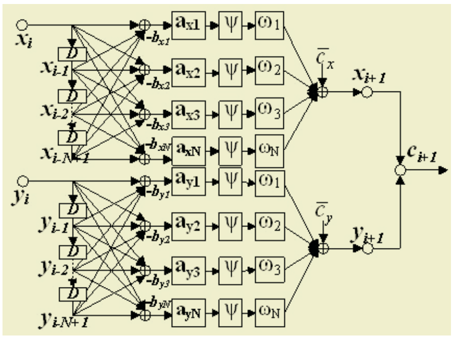


Fig. 1. The Wavelet Neural Network used for the mobility prediction

3.2 Clustering Algorithm

It is important to predict the mobility of the node in MANET. Precise prediction would facilitate clustering and enable the seamless handover of the mobile IP. From the wavelet neural network, we can not only get the next possible cluster, but also a series of cluster number that reflect the node trajectory. We would select the mobile node 'x' with the most probability to cluster 'ci' node as the cluster head.

We defines some terms appeared in this clustering algorithm firstly. Node ID is a string that uniquely identifies a particular node. A cluster consists of a group of nodes. Each node in the network knows its corresponding cluster head. A cluster head is elected in the cluster formation process for each cluster. Each cluster should have only one cluster head. The cluster head has a bi-directional link to every node in the cluster. The routine of the cluster head including:

(1) Maintain the ID of member node in its cluster.
(2) Designate node with maximum visit probability as the Backup cluster head.
(3) Send CH_CHANGE message [4] to the Backup cluster head.
(4) Assign one ID to the coming-in node when a node moving into cluster k.

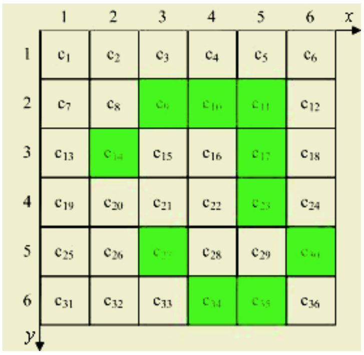
In this proposed clustering algorithm, the member node predicts its mobility by the wavelet neural network and computes its visit probability to the cluster. The backup cluster head should implement the following:

(1) Update and backup the information of the member node in the cluster head.
(2) Upon the reception CH_CHANGE message from the cluster head.

4 Simulation and Evaluation

The simulations compare the performance of our clustering algorithm with the Lowest-ID, maximum connectivity clustering algorithms, in terms of stability of clusters being formed. The cluster stability is measured by determining the number of each mobile node either attempts to become or gives up the cluster head role.

The y-axis of Fig. 3 (left figure) shows the statistical frequency of cluster head changes, and hence measures the stability associated with each clustering algorithm. As it can be seen from Fig. 3, the WNNP clustering algorithm leads to more stable cluster formation. The right figure depicts the average service time of each cluster head. The longer the service time of each cluster head, the better its support for cluster stability is. As it can be seen from the right figure, in the WNNP each cluster head has longer average service time than that of any other algorithm. These simulations are performed for up to 200 mobile nodes. The cluster stability is measured by calculating the total number of cluster head changes.



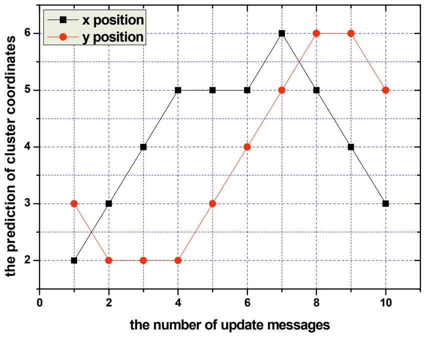
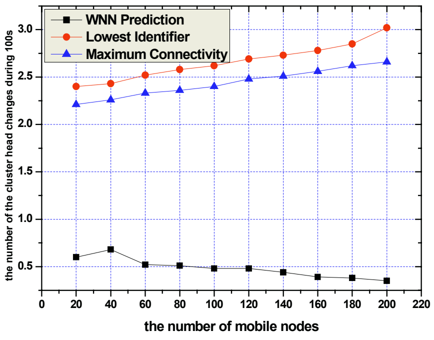


Fig. 2. Movement trajectory and mobility prediction based on WNNP. In the left figure, A node begins moving at the cluster c_{14} (2, 3). The movement history is a set of clusters {c_{14}, c_9, c_{10}, c_{11}, c_{17}, c_{23}, c_{30}, c_{35}, c_{34}, c_{27}}. At last, the node arrived at the cluster c_{27} with position (3, 5). The right figure give the predictions by the horizontal and vertical position separately



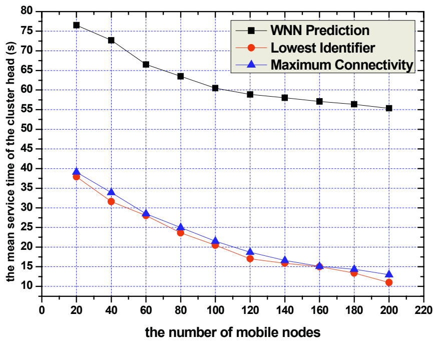


Fig. 3. The simulation results of clustering algorithms. The left figure shows the statistical frequency of cluster head changes. The right figure depicts the average service time. All the simulations are compared with Lowest-ID, maximum connectivity clustering algorithm

5 Conclusions

In this paper we presented a new clustering approach that makes use of intelligent mobility prediction based on the wavelet neural network in mobile ad hoc. We have demonstrated that this clustering scheme results in more stable clusters than other well-known schemes. This stability improvement, however, depends on the accuracy of our mobility prediction. Our future work is going to be on inter-cluster seamless handover of mobile IP and SCTP.

References

1. Hou, T.C.: An Access-based Clustering Protocol for Multihop Wireless Ad Hoc Networks. IEEE Journal on Selected Areas in Communications, **19** (2001) 1201–1210
2. Bhattacharya, A.: LeZi-Update: An Information-Theoretic Approach to Track Mobile Users in PCS Networks. MobiCom, **99** (1999) 1–12
3. Sivavakeesar, S.: Stable Clustering Through Mobility Prediction for Large-Scale Multihop Intelligent Ad Hoc Network. WCNC'04 (2004) 1488–1493
4. Basu, P., Khan, N., Little, D.C.: Mobility Based Metric for Clustering in Mobile Ad Hoc Networks. Workshop on Distributed Computing Systems (2001) 413–418
5. Huang, J.N., You, S.S., Lay, S.R.: The Cascade Correlation Learning: A Projection Pursuit Learning Perspective. IEEE Trans. on Neural Networks, **7** (1996) 278–289

Internet Traffic Prediction by W-Boost: Classification and Regression*

Hanghang Tong[1], Chongrong Li[2], Jingrui He[1], and Yang Chen[1]

[1] Department of Automation, Tsinghua University, Beijing 100084, China
{walkstar98,hejingrui98}@mails.tsinghua.edu.cn
[2] Network Research Center of Tsinghua University, Beijing 100084, China
licr@cernet.edu.cn

Abstract. Internet traffic prediction plays a fundamental role in network design, management, control, and optimization. The self-similar and non-linear nature of network traffic makes highly accurate prediction difficult. In this paper, we proposed a new boosting scheme, namely W-Boost, for traffic prediction from two perspectives: classification and regression. To capture the non-linearity of the traffic while introducing low complexity into the algorithm, 'stump' and piece-wise-constant function are adopted as weak learners for classification and regression, respectively. Furthermore, a new weight update scheme is proposed to take the advantage of the correlation information within the traffic for both models. Experimental results on real network traffic which exhibits both self-similarity and non-linearity demonstrate the effectiveness of the proposed W-Boost.

1 Introduction

Internet traffic prediction plays a fundamental role in network design, management, control, and optimization [12]. Essentially, the statistics of network traffic itself determines the predictability of network traffic [2], [12]. Two of the most important discoveries of the statistics of Internet traffic over the last ten years are that Internet traffic exhibits self-similarity (in many situations, also referred as long-range dependence) and non-linearity. Since Will E. Leland's initiative work in 1993, many researchers have dedicated themselves to proving that Internet traffic is self-similar [10]. On the other hand, Hansegawa et al in [6] demonstrated that Internet traffic is non-linear by using surrogate method [16]. The discovery of self-similarity and non-linearity of network traffic has brought challenges to traffic prediction [12].

In the past several decades, many methods have been proposed for network traffic prediction. To deal with the self-similar nature of network traffic, the authors in [15] proposed using FARIMA since FARIMA is a behavior model for self-similar time series [4]; the authors in [19] proposed predicting in wavelet domain since wavelet is a natural way to describe the multi-scale characteristic of self-similarity. While these methods do improve the performance of prediction for self-similar time series, they are both time-consuming. To deal with the non-linear nature of network traffic, Arti-

* This work is supported by National Fundamental Research Develop (973) under the contract 2003CB314805.

ficial Neural Network (ANN) is probably the most popular method. ANN can capture any kind of relationship between the output and the input theoretically [6], [9], [11], however, it might suffer from over-fitting [5]. Another kind of prediction method for non-linear time series is support vector regression (SVR) [11] which is based on structural risk minimization. However, the selection of suitable kernel functions and optimal parameters might be very difficult [6].

In our previous work [17], [18], we have introduced boosting technique into traffic prediction by considering it as a regression problem. The initial experimental results demonstrated the prediction performance by Feed-Forward Neural Network (FFNN) can be largely improved by boosting technique. However, some very important issues leave to be solved: 1) Which model, classification or regression, is more suitable for traffic prediction? 2) How to design weak learners? 3) How to update weight distribution? In this paper, all these aspects are investigated under a new boosting scheme, namely W-Boost. Firstly, network traffic prediction is modeled from two perspectives: both classification and regression are considered. To capture the non-linearity of the traffic while introducing low complexity into the algorithm, 'stump' and piecewise-constant function are adopted as weak learners for classification and regression, respectively. Furthermore, a new weight update scheme is proposed for both models, which aims to maximally take the advantage of the correlation information within the traffic. Experimental results on real network traffic which exhibits both self-similarity and non-linearity demonstrate the effectiveness of W-Boost.

The rest of this paper is organized as follows: in Section 2, the proposed W-Boost for traffic prediction is presented in detail. Experimental results are given in Section 3; finally, we conclude the paper in Section 4.

2 W-Boost for Traffic Prediction

As a general method for improving the accuracy of some weak learning algorithms, boosting has been shown to be very effective for classification [1], [8], [14]. When applied to traffic prediction, there are mainly two classes of methods: 1) modifying some specific steps of the existing boosting algorithms for classification so that they are suitable for a regression problem [1], [8]; 2) modeling traffic prediction as a classification problem by introducing an additional variable [13], [14]. Both methods will be investigated in the proposed W-Boost: on the whole, W-Boost shares the similar flow-chart of Ada-Boost as well as Real-AdaBoost. In this section, after giving out the problem definition, we will discuss the details of two key points of W-Boost, which make it different from Both Ada-Boost and Real-AdaBoost: weak learner design and weight update scheme.

2.1 Problem Definition

To predict network traffic, it is generally denoted as a time series: $X = (x_i : i = 0, 1, 2, \cdots)$. The prediction problem can be defined as follows [2]: given the current and past observed values $X_i = (x_{i-p+1}, \cdots, x_{i-1}, x_i)$, predict the future value x_{i+q}, where p is the length of history data used for prediction and q is the prediction step.

Prediction as Regression: In practice, the traffic prediction problem can be considered as a classical regression problem under the assumption that $x_i \in \zeta_2\ (i = 1,2,\cdots)$ [2]. That is, the prediction of x_{i+q} can be written as:

$$\hat{x}_{t+q} = \underset{y \in \xi_2}{\operatorname{argmin}} E\{(y - x_{t+q})^2\}\ . \tag{2}$$

Prediction as Classification: As in [13], traffic prediction can also be modeled as a classification problem by introducing an additional reference variable S:

$$\hat{x}_{t+q} = \inf_{s} \{s : P(Y^* = 1 \mid \{x_{t-p}, \cdots, x_t\}, S) \geq 1/2\}\ . \tag{3}$$

Where S is the reference variable, and $Y = \{0,1\}$ is the output defined on $(\{x_{t-p}, \cdots, x_t\}, S)$ (For details, refer to [13]).

2.2 Weak Learner Design

Like Real-AdaBoost [3], in W-Boost, a weak learner is trained for each feature $x_{t-i}(i = 0, \cdots p)$ and the one with the minimum training error is selected in each iteration. A good weak learner design should consider two factors: 1) to capture the non-linearity within network traffic, the weak learner itself should be non-linear; 2) on the other hand, to achieve good generation performance of the final combined learner, such weak learner should not be too 'complex'. Basted on the above observation, the weak learner in W-Boost is designed as follows:

Weak Learner in Classification: In this case, "stumps" is adopted as the weak classifier. "Stumps" are single-split trees with only two terminal nodes [3].

Weak Learner in Regression: In this case, the piece-wise-constant function (PWC) [16] is adopted as the weak regressor.

2.3 Weight Update Scheme

How to exploit the correlation structure is the key problem in traffic prediction [12]. In [17], we proposed using PCA as a preprocessing step to take advantage of self-similar nature of traffic while avoiding the disadvantage of self-similarity. In W-Boost, by using each feature to training a weak leaner, we could make use of the correlation structure of traffic in a more sophisticated way, which is motivated by our previous work in image classification [7]. Its key point is that, in W-Boost, a set of weights are kept on all examples for each feature component to generate the weak learner. Let $w(i), w(j)\ (i, j = 0, \cdots p; i \neq j)$ denote the present sets of weights for feature component x_{t-i} and x_{t-j}, respectively. Let $w'(i), w'(j)$ denote the updated sets of weights. In W-Boost, they are updated as follows:

1. Suppose $x_{t-i}(i = 0, \cdots p)$ is selected to generate the weak learner in the current iteration. Update $w(i)$ as Ada-Boost (Real-AdaBoost): $w(i) \xrightarrow[\mathrm{Re}al-AdaBoost]{Ada-Boost} w'(i)$;
2. For all $x_{t-j}(j = 0, \cdots p; j \neq i)$, update $w(j)$ as Eq. 4:

$$w'(j) = (1 - \alpha_{ij}) w'(i) + \alpha_{ij} w(j)\ . \tag{4}$$

where $\alpha_{ij} \in [0,1]$ is the parameter, indicating the dependence extent between x_{t-i} and x_{t-j} : if they are independent, α_{ij} =1; else if they are totally dependent, $\alpha_{ij} = 0$. In [7], α_{ij} is estimated by the Kullback-Leibler distance between x_{t-i} and x_{t-j} . For traffic prediction, it can also be approximated by the auto-correlation function (ACF) of the time-series $X_i = (x_{i-p+1}, \cdots, x_{i-1}, x_i)$, i.e. $\alpha_{ij} \approx 1 - ACF(|i-j|)$.

3 Experimental Results

The network traffic that we use is the JPEG and MPEG version of the “Star Wars” video which is widely used to examine the performance of the network traffic prediction algorithms [9]. In our experiment, we divide the original traffic into some traces with equal length 1000. Then we make use of variance-time [10] and surrogate method [16] to test self-similarity and non-linearity of a given trace, respectively. Those traces which exhibit both self-similarity and non-linearity are selected to examine the performance of BBF-PT. Furthermore, each trace is normalized to $[0,1]$ for comparison simplicity.

There are a set of parameters and operations that need to be set in W-Boost:

- ♦ The length of history data p used for predicting is set to be 10;
- ♦ At the current stage, we are concerned with one-step prediction so q is 1;
- ♦ We used the first 50% data in each trace to compose the training set;
- ♦ The length of the reference variable S in classification model is set to be 20;
- ♦ For both classification and regression models, the maximum iteration number and the bin number are determined by cross-validation using the training data;
- ♦ The mean-square-error (MSE) is used to evaluate different methods.

First, both classification model by W-Boost (CMW) and regression model by W-Boost (RMW) are evaluated. The results are compared with those by Support Vector Machine (SVM) and Support Vector Regression (SVR), respectively. It can be seen from Figure 1 that 1) for both CMW and RMW, W-Boost results in better or comparable performance with those by SVM and SVR, respectively; 2) both SVR and RMW outperform SVM and CMW, indicating that for traffic prediction, regression might be more suitable. Then, the weak learner design and weight update scheme in W-Boost are evaluated. In this case, we only present the result by regression model for the limited space. The result are compared with that by what is proposed in our previous work [17], where Feed-forward Neural Network is adopted as the basic weak regressor and Principle Component Analysis is used as a preprocess step (FFNN+PCA). From Figure 2, it can be seen that W-Boost can further boost the prediction performance. Finally, two prediction examples are presented in Figure 3.

4 Conclusions

In this paper, we have proposed a new boosting version, namely W-Boost for self-similar and non-linear network traffic prediction. On the whole, W-Boost shares the similar flow-chart of both Ada-Boost and Real-AdaBoost. W-Boost supports both regression and classification methods for traffic prediction. To capture the nonlinearity of the traffic while introducing low complexity into the algorithm, “stumps” and piece-wise-constant function are adopted as weak learners for classification and regression, respectively. Furthermore, a new weight update scheme is proposed for both models, which aims to maximally take the advantage of the correlation information within the traffic. Experimental results demonstrate that 1) the regression model is more effective for traffic prediction; and 2) both the proposed weaker learner and weight update scheme are effective.

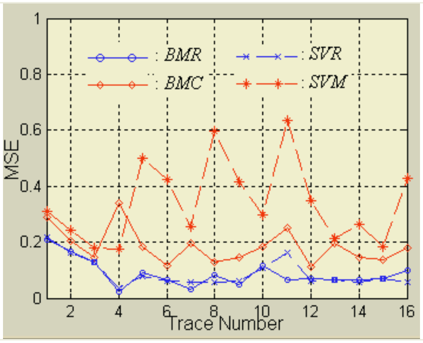


Fig. 1. Evaluation of different models

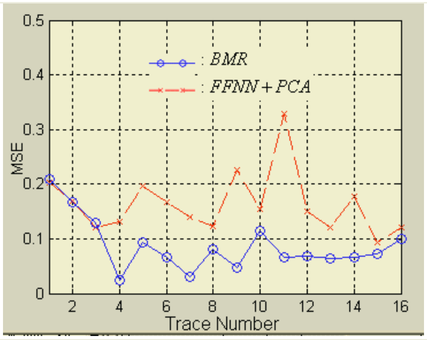


Fig. 2. Evaluation of weak learner design and weight update scheme in W-Boost

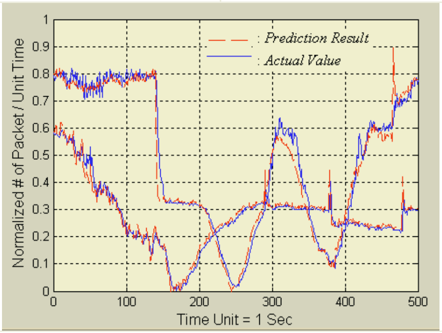


Fig. 3. Prediction example using regression by W-Boost

References

1. Bone, R., Assaad, M., Crucianu, M.: Boosting Recurrent Neural Networks for Time Series Prediction. Proceedings of the International Conference on Artificial Neural Networks and Genetic Algorithms (2003)
2. Brockwell, P., Davis, R.: Time Series: Theory and Methods. Springer-Verlag, New York, 2nd edition (1991)
3. Friedman, J., Hastie, T., Tibshirani, R.: Additive Logistic Regression: A Statistical View of Boosting. The Annual of Statistics, **28** (2000) 337-374
4. Gripenberg, G., Norros, I.: On the Prediction of Fractional Brownian Motion. Journal of Applied Probability, (1996) 400-410

5. Hall, J., Mars, P.: The Limitations of Artificial Neural Networks for Traffic Prediction. IEE Proceedings on Communications (2000) 114-118
6. Hansegawa, M., Wu, G., Mizuno, M.: Applications of Nonlinear Prediction Methods to the Internet Traffic. The 2001 IEEE International Symposium on Circuits and Systems, (2001) 169-172
7. He, J., Li, M., Zhang, H.J., Zhang, C.: W-Boost and Its Application to Web Image Classification. IEEE Int. Conf. on Pattern Recognition (2004) 148-151
8. Kegl, B.: Robust Regression by Boosting the Median. COLT/Kernel, (2003) 258-272
9. Khotanzad, P., Sadek, N.: Multi-Scale High-Speed Network Traffic Prediction Using Combination of Neural Network. IJCNN (2003) 1071-1075
10. Leland, W.E., Taqqu, M.S., Willinger, W., Wilson, D.: On the Self-Similar Nature of Ethernet Traffic. IEEE/ACM Tran. on Networking (1994) 1-15
11. Muller, K.: Predicting Time Series with Support Vector Machines. Proceedings of the International Conference on Artificial Neural Network (1997) 999-1004
12. Ostring, S., Sirisena, H.: The Influence of Long-rang Dependence on Traffic Prediction. IEEE ICC, (2001) 1000-1005
13. Ridgeway,D., Madigan, D., Richardson, T.: Boosting Methodology for Regression Problem. Proc. 7th Int. Workshop on Artificial Intelligence and Statistics (1999) 152-161
14. Schapire, R.E.: The Boosting Approach to Machine Learning: an Overview. MSRI Workshop on Nonliear Estimation and Classification (2002)
15. Shu, Y., Jin, Z., Zhang, L., Wang, L.: Traffic Prediction Using FARIMA Models. IEEE ICC (1999) 891-895
16. Small, M., Yu, D., Harrison, R.G.: Surrogate Test for Pseudoperiodic Time Series Data. Physical Review Letter (2001)
17. Tong, H., Li, C., He, J.: A Boosting-Based Framework for Self-Similar and Non-linear Internet Traffic Prediction. ISNN (2004) 931-936
18. Tong, H., Li, C., He, J.: Boosting Feed-Forward Neural Network for Internet Traffic Prediction. ICMLC (2004)
19. Wang, X., Shan, X.: A Wavelet-based Method to Predict Internet traffic. IEEE International Conference on Communications, Circuits and Systems and West Sino Expositions (2002) 690-694

Fuzzy Neural Network for VBR MPEG Video Traffic Prediction

Xiaoying Liu, Xiaodong Liu, Xiaokang Lin, and Qionghai Dai

Research Center of Broadband Network & Multimedia, Graduate School at Shenzhen, Tsinghua University, Shenzhen, Guangdong 518055, China
liuxy@sz.tsinghua.edu.cn

Abstract. As a main video transmission mode for digital media networks, the capability to predict VBR video traffic can significantly improve the effectiveness of quality of services. Therefore, based on fuzzy theory and neural network, a novel video traffic prediction model is proposed for the complex traffic characteristics of MPEG videos. This model reduces the prediction error and computation. Simulation results show that the proposed method is able to predict the original traffic more accurately and reliably than the conventional AR method.

1 Introduction

The digital media networks has been the present trending of merging networks, such as the internet, ATM, and wireless communication so on, how to provide high-speed transmission for a wide range of quality of services (QoS) is becoming more and more important. Video has now become one of the major components of network multimedia services, MPEG as a very popular standard encoding scheme has been widely used in video applications, and the bit streams of MPEG traffic have a naturally variable bit rate (VBR), therefore, an efficient VBR MPEG video traffic transmission mechanism is vital to a network operation [1].

However, owing to the complex traffic characteristics of MPEG videos, it is very difficult to build a satisfied prediction model of VBR MPEG video traffic with only a few known parameters. To overcome this problem, various dynamic resource allocation methods have been proposed. Traffic prediction is the most generally used technique in dynamic methods, the more accurate the traffic predictions, the better reliable the prevision of QoS. Hence there is an important meaning of VBR MPEG video sources traffic prediction modeling for improving QoS [2].

In recent years, there are mainly several studies on the on-line prediction of VBR MPEG traffic. For example, the recursive least square and the time delay neural network methods are used for JPEG and MPEG video traffic prediction, an extension of the recursive least square method is proposed for multiplexed MPEG video traffic, a linear prediction scheme is based on the least mean square, and an adaptive wavelet prediction method is proposed for fast convergence [3]. In respect that the advanced artificial intelligent technology is good at resolving a kind of inenarrable problems with real-time requirement, compared with the conventional predictions based on mathematical modeling, which is more applied to deal with complex video transmission over the digital media networks. Therefore, the video traffic prediction based on artificial intelligent theory has been the hotspot of research [4].

On the base of MPEG video characteristics, this paper proposes a novel VBR MPEG video traffic prediction model using fuzzy theory and neural network, which integrates the parallel processing and self-learning capabilities of neural network with the expression capability of fuzzy logic for difficult description process. Simulation results show not only the prediction errors of the proposed model are significantly smaller than the conventional AR models [5], but also this model provides an improved video traffic prediction technique.

2 MPEG Video Characteristics

A primary characteristic of a typical MPEG video is the mode in which the picture types are generated. A MPEG video includes three frame types: *I*, *P*, and *B*. Even though not standard requirement, all frames of a given video sequence are mostly arranged in a deterministic normal pattern, which is called a GOP pattern. The character of a GOP pattern is in two parameters: one is the frame distance *N* from one *I*-frame picture to the next *I*-frame picture, the other is the frame distance *M* from one *I*-frame picture to the following *P*-frame picture or two successive *P*-frame pictures. In the processes of visual information compression, since coding type and pseudo-periodical traffic pattern of MPEG are depending on the GOP patterns, VBR MPEG broadcasting videos have complex and burst traffic characteristics, which are not statistically stationary processes.

Generally, most of digitized video sources are mostly coded with the MPEG-2 encoder. For instance, a number of digitized standard test video sequences, which are the most typical ones used by researchers in this area, are coded with the MPEG-2 encoder, then the output streams are saved as disk files and subsequently analyzed. The video sequences as test benchmark are often applied in the study, which include short ones such as “flower garden”, “mobile and calendar” etc., and longer ones such as “star wars”, “soccer” and so on. They are fairly representative for scenes from some feature films and documentaries with medium level of scene changes. After the hybrid algorithm (MC/DPCM/DCT) is adopted, an output bit stream is periodically produced with *I, P* and *B* pictures. The traffic of an *I* frame subsequence, which uses intra-coding without reference to other frames, so only moderate compression is involved. *P* frames are coded using a motion-compensated prediction from a previous *I* or *P* frame, they are more greatly compressed than the *I* frames. Because a *B* frame uses a bidirectional predictive coding scheme, a *B* frame subsequence may require both previous and future *I* or *P* frames with interpolation for the best compression. The video traffic trace is intercepted from “star wars” as illustrated in Fig. 1, which shows the complete coded stream contains sharp and somewhat periodical variations in bit rates. Whereas the separated *I, P* and *B* streams present much more mild bit rate changes in Fig. 2. It can be seen that the prediction is more effective for the separated individual *I, P* and *B* streams than the mixed streams.

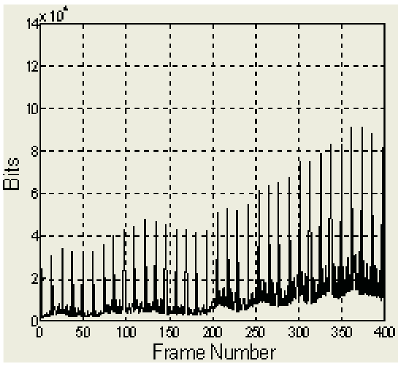


Fig. 1. Mixed stream of MPEG video

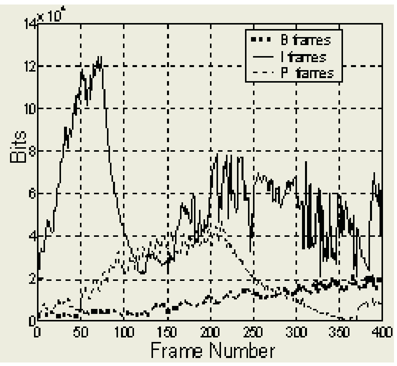


Fig. 2. Separated streams of MPEG video

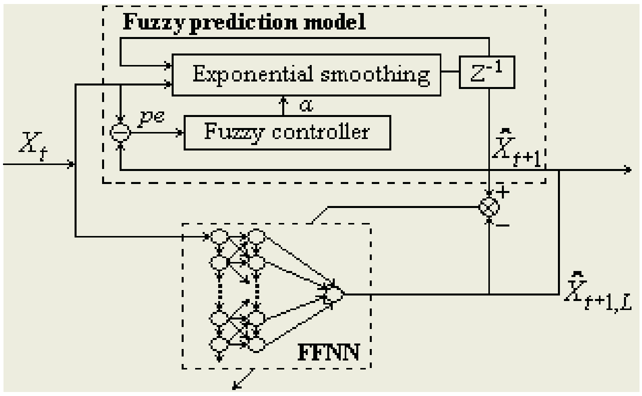


Fig. 3. Video traffic prediction model based on fuzzy model and neural network

3 Fuzzy Neural Network Video Traffic Prediction Model

Fig. 3 shows the architecture of our proposed fuzzy neural network video traffic prediction model, where fuzzy prediction model is used for one-step prediction to improve the prediction accuracy, and a three-layer feedforward neural network (FFNN) is utilized for multi-step prediction to ensure the real-time prediction. Furthermore, the self-learning function is added into this model for video varies.

3.1 Fuzzy Prediction Model

Because general traffic patterns are non-stationary or non-Gaussian processes without intuitive relationship between the parameters and the estimation error, the exact mathematic model couldn't represent them. It is therefore hard to get the satisfied prediction results using the conventional AR models. Since fuzzy theory can change complex prediction problem into ordinary linear programming problem, and the time sequence method can overcome influences of randomicity produced by accidental factors, a fuzzy prediction model is designed by combining these advantages of fuzzy theory with the time sequence method.

If the adjustable parameter is α, the actual frame rate is X_t, then one-step prediction can be given by:

$$\hat{X}_{t+1,1} = \alpha X_t + (1-\alpha)\hat{X}_{t,1} \ . \tag{1}$$

where, $\hat{X}_{t+1,1}$ is the one-step ahead predicted frame rate of X_t (namely the future predicted frame rate of X_t), $\hat{X}_{t,1}$ is the predicted frame rate of X_t (namely the previous predicted frame rate of X_t), α is a smoothing constant as the weight given to the history ($0 \le \alpha \le 1$).

By applying the recursion, multi-step prediction can be derived from (**1**) as follows:

$$\hat{X}_{t+1,L} = \alpha \hat{X}_{t,L-1} + (1-\alpha)\hat{X}_{t,L} \ , \quad L = (1,2,\ldots,n) \ . \tag{2}$$

where, L is the number of prediction steps, $\hat{X}_{t+1,L}$ is the L-step ahead predicted frame rate of X_t, $\hat{X}_{t,L}$ is the L-step predicted frame rate of X_t, $\hat{X}_{t,L-1}$ is the step of $(L-1)$ predicted frame rate of X_t. When L=1, the multi-step prediction (**2**) is equivalent to the one-step prediction (**1**).

Since α is the only controlled parameter, a proportional error of prediction is considered as input and α as output for the fuzzy controller. The proportional error of prediction can be defined as:

$$pe = \left|\hat{X}_{t,1} - X_t\right| \Big/ \hat{X}_{t,1} \ . \tag{3}$$

where, $pe \in [0,1]$ is the proportional error. The smaller proportional error means the more accuracy of prediction. Set pe and α as three levels {*Large, Medium, Small*} in the range of [0,1], then the inferences of α can be expressed by the fuzzy rules.

3.2 Neural Network

Due to the recursive application for multi-step prediction, the computation required for the proposed fuzzy prediction model will increase linearly with the number of prediction steps. In order to limit the computation prediction for real-time operation and keep prediction accuracy at the same time, using powerful computability of neural network with high parallelity, a three-layer FFNN is designed for multi-step prediction. The input sequence is shifted through layer one and the output is given by the single neuron of layer three. Layers one and three are linear transfer functions; layer two as the hidden layer uses hypertangent sigmoid transfer function. The number of neurons in layers one and two are equivalent.

4 Simulation Results

To evaluate the proposed prediction model, standard video sequence is tested as original trace. For example, obtain 3333 frames of *I* pictures from the standard video sequence "star wars" as the original trace, which has a mean $m = 4.401 \times 10^4$ (Bits/Frame) and standard deviation $\sigma = 1.414 \times 10^4$. Relative errors are employed to measure m and σ that are gotten from the prediction trace with which from the original trace, which are denoted by $E(m)$ and $E(\sigma)$ respectively. The experimental results of prediction errors comparing the proposed method with traditional methods are shown in Table 1, the actual and the predicted rates (for example, 20-step ahead prediction) are presented in Fig. 4. Where, AR(1) ~ AR(4) are linear AR prediction models of up to 4th order, FNN is the proposed prediction model.

Table 1. Comparisons of predication errors

Experimental method	One-step prediction		Multi-step prediction (L=20)	
	$E(m)$ (%)	$E(\sigma)$ (%)	$E(m)$ (%)	$E(\sigma)$ (%)
AR(1)	17.79	15.21	31.09	29.45
AR(2)	9.45	10.71	19.78	20.34
AR(3)	2.78	4.47	9.58	11.07
AR(4)	1.73	4.08	7.39	10.46
FNN	0.11	0.50	1.01	2.37

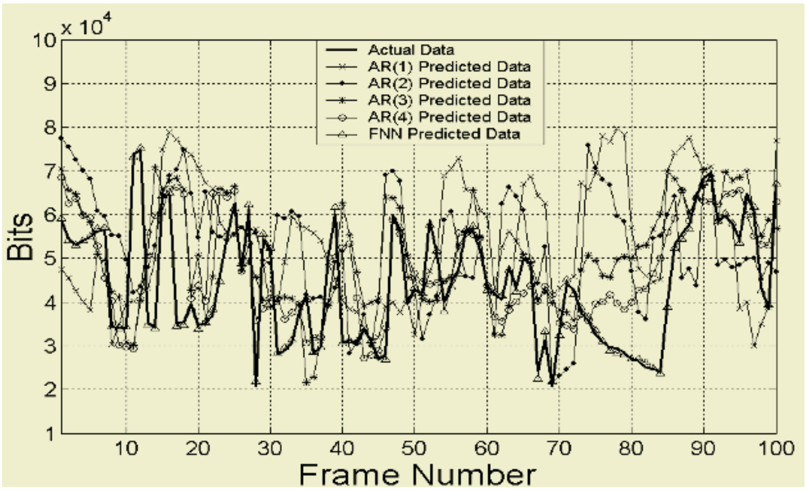


Fig. 4. Comparisons of 20-step ahead prediction results

We can see that higher order AR processes have not a clear advantage over lower order ones, since the prediction errors of AR models have not evidently decreased with the higher order, however the computation complexity has increased because the number of parameters required for AR predictions equal to the order of the AR. Comparing with AR predictions, the prediction errors of FNN have markedly reduced, not only just one parameter α is needed for the prediction, but also the implementation of FFNN meets real-time operation. So this proposed model provides a more exact and simple efficient prediction method for video transmission.

5 Conclusions

Since VBR MPEG video could not be shown as stable normal distribution process, it is difficult to get satisfied prediction results by the conventional linear AR models.

For resolving the video traffic control problem on network transmission, we have designed the intelligent integrated prediction model to predict video traffic based on fuzzy logic and neural network. Through the analysis for MPEG characteristics, using the advantages of artificial intelligent technology to deal with a type of complex and real-time problems, we can obtain more accurate prediction by the proposed intelligent model. Experimental results show that the proposed video traffic prediction scheme generates predicted data very close to actual data and provides improved computing efficiency for real time application.

Acknowledgments

This work was supported by the National Key Natural Science Foundation of China (NSFC 60432030).

References

1. Basir, I., Namuduri, K.R., Pendse, R.: A Light Weight Dynamic Rate Control Scheme for Video Transmission over IP Network. Pattern Recognition Letters, **25** (2004) 817-827
2. Farshchian, M., Cho, S., Pearlman, W.A.: Optimal Error Protection for Real-Time Image and Video Transmission. IEEE Signal Processing Letters, **11** (2004) 780-783
3. Yoo, S.J.: Efficient Traffic Prediction Scheme for Real-Time VBR MPEG Video Transmission Over High-Speed Networks. IEEE Trans. Broadcasting, **48** (2002) 10-18
4. Rikli, N.E.: Modeling Techniques for VBR Video: Feasibility and Limitations. Performance Evaluation, **57** (2004) 57-68
5. Chan, W.S., Cheung, S.H., Wu, K.H.: Multiple Forecasts with Autoregressive Time Series Models: Case Studies. Mathematics and Computers in Simulation, **64** (2004) 421-430

Building an Intrusion Detection System Based on Support Vector Machine and Genetic Algorithm

Rongchang Chen[1], Jeanne Chen[2], Tungshou Chen[3], Chunhung Hsieh[3], Teyu Chen[4], and Kaiyang Wu[3]

[1] Department of Logistics Engineering and Management
National Taichung Institute of Technology
rcchens@ntit.edu.tw
[2] Department of Computer Science and Information Management
Hungkuang University, Taichung, Taiwan 433, China
jeanne@sunrise.hk.edu.tw
[3] Department of Information Management
National Taichung Institute of Technology
No. 129, Sec. 3, Sanmin Rd., Taichung, Taiwan 404, China
{tschen,guu}@ntit.edu.tw
[4] Department of Information Management, Hsiuping Institute of Technology
No.11, Gungye Rd., Dali City, Taichung, Taiwan 412, China

Abstract. Host-based Intrusion Detection System (IDS) utilizes the log files as the data source and is limited by the content of the log files. If the log files were tampered, the IDS cannot accurately detect illegal behaviors. Therefore, the proposed IDS for this paper will create its own data source file. The system is controlled by the Client program and Server program. The client program is responsible for recording a user's behavior in the data source file. The data source file is then transmitted to the server program, which will send it to SVM to be analyzed. The analyzed result will then be transmitted back to the client program. The client program will then decide on the course of actions to take based on the analyzed result. Also, the genetic algorithm is used to optimize information to extract from the data source file so that detection time can be optimized.

1 Introduction

Protecting stored digital data is an age-old problem. The intrusion detection system (IDS) [1],[2],[3] is a popularly used technique to cope with the security issues that include tampering, deletions and piracies. Graham [4] defined intrusion as someone who invaded a computer. A user is defined as legal when authorization is given to access the computer. The usage behavior of the legal user is regarded as the legal user behavior. On the contrary, a person regarded as illegal by IDS is an illegal user and is also called an intruder. The behavior is regarded as the illegal behavior.

In host-based IDS, the log files are the data source files [3] used to analyze intrusion behaviors. The log files usually contain recorded events. If the log files are tampered with, the IDS will assume them to be correct or fail to analyze if the log files were deleted. However, the log files cannot provide information such as, the time a user wanted to use the computer, the kind of programs a user wanted to execute and the typing speed of a user. If the usage behavior of an illegal user is known then a rule from this behavior can be found which can be used by the IDS to detect the user's status.

The proposed IDS for this study is based on a client-server [5] architecture. The users (clients) will each be installed with a client program, while the manager (server) will be installed the server program. There is one server to many clients. The client program collects the behavior information of a user and records them in the data source files. The content of this file includes: the typing speed and the time that the application programs were executed. The data source files will be transmitted to the server program. At the server end, SVM [6],[7],[8] will be used to analyze the behavior of the user and a response is sent back to the client program. The huge amount of information to analyze from data source files could overload the computer process time and cause wastage to its resources [2], [9]. Therefore, the genetic algorithm (GA) [10] will be used to analyze for the application programs a user is permitted. By decreasing the amount of data to be analyzed, the detection rate of IDS is increased and errors reduced.

2 Genetic Algorithm (GA)

The genetic algorithm (GA) used in problem solving requires the variables to be in binary form [10]. Each bit represents a gene. The stringed binary bits (genes) make a chromosome. The genetic composition of each chromosome represents a solution to a problem. The purpose of GA is to find the best solution for a problem from a pool of solutions (chromosomes). A fitness value will be used to measure the “fitness” of a chromosome. The fitness of a chromosome reflects its suitability to produce “fit” offspring and its relative nearness to the solution for the problem at hand.

The evolution process of GA begins by repeating the genetic cycle of manipulating chromosomes from an initial random population of chromosomes to generate new generations consisting of “fit” offspring. Chromosomes will be selected through a special selection process for the mating pool. Parents from the mating pool will then undergo genetic crossovers and mutation to produce the offspring. There are two most-used types of selection process: (1) roulette wheel selection (RWS) and (2) tournament selection (TS). In RWS**,** the sum of fitness of the chromosomes is defined as the circumference of a roulette wheel selection. The chromosome is calculated for its fitness added to the fitness of the members in preceding population that lies within the same fitness for the segment on the roulette. The chromosome with the largest segment on the roulette “wins” the fitness game. The smallest segment is the “loser”. In TS, the chromosomes will be selected for the mating pool from the highest fitness to the lowest in decreasing order. The selection process will use the probability of reproduction (*PoR*) to maintain the size of mating pool to be the same as the population size.

Crossover and Mutation. Once parents are chosen to mate, the crossover process occurs. The crossover process is based on the probability of crossover *PoC*. Not all chromosomes can crossover and are, therefore, eliminated. A normal *PoC* rate is between 0.6 and 1.0 [10]. There are many types of crossover, mainly (a) one-point crossover, (b) two-point crossover and (c) uniform crossover.

Mutation occurs after crossovers. Its sole purpose is to increase GA’s chance towards the best solution. Each bit can be randomly altered with a probability of mutation *PoM*.

3 The Proposed IDS Scheme

The proposed IDS scheme is based on the client-server. At the client end, users' activities will be recorded and transmitted to the server where activities at the client end are being monitored. Activities collected by the client program will be trained for the legal and illegal behaviors. The trained model will be used to detect the behavior of a user. Also, GA is used to increase the system's efficiency by reducing the amount of unnecessary information that is being analyzed by the server.

3.1 IDS Framework

The proposed IDS scheme is client-server based [5], [9]. The main framework is divided into two parts: the client program and the server program. The client program will be installed in the client computers, which will be monitored by the server program. The server program is installed in the server computer, and will monitor activities on the client sites. The client-server programs will bind a socket on TCP, and deliver data via the socket.

Client Program. The client program collects information on user behaviors and feeds them as inputs for LIBSVM [8]. The LIBSVM is a simpler and easier tool to use for SVM. The training data for LIBSVM can be divided into two classes: legal behaviors and illegal behaviors.

The client program collects user behavior every *m* minutes; thereby, generating a data source file for a user per *m* minutes. The content of the data source for a user contains: (a) login by day of the week, (b) login by segment of the day, (c) average typing speed, and (d) application programs executed within the *m* minutes. Day of the week refers to Monday through Sunday; segment of the day refers to morning, noon, evening and midnight; typing speed is the number of characters typed in *m* minutes and; application programs refer to programs the user is authorized to execute.

All these constitute the behavior of an authorized user. The proposed IDS scheme will use the behavior data to determine the status of a user. Only single items are collected for the first three behaviors. The fourth is based on application programs executions, which could be multiple items. Too many of the fourth type could slow down the performance of IDS. Therefore, GA will be used to reduce the number of application programs for analysis. The user status detection rate can be improved as well as reducing analysis errors. The data source will be delivered to the server program through a socket where an analysis on the user's status will be transmitted back. The client program will then decide whether to display a warning message for the illegal user. Fig. 1 illustrates the flow for the client program.

Server Program. The server program monitors the client sites and transmits back to the sites analysis on the data sources it received from clients. The data source is analyzed through LIBSVM to determine a user's status. To increase the success rate for predicting a user's status, more samples of users' behaviors will have to be collected to train models for the predictions. These behaviors can be divided into the legal behaviors and illegal behaviors. The legal behaviors are confirmed behaviors by the client computer administrator. A behavior is considered illegal if it is not from the client computer administrator. Legal and illegal user behaviors are collected and stored in a training file to be used in LIBSVM for training a user model for the client computer administrator.

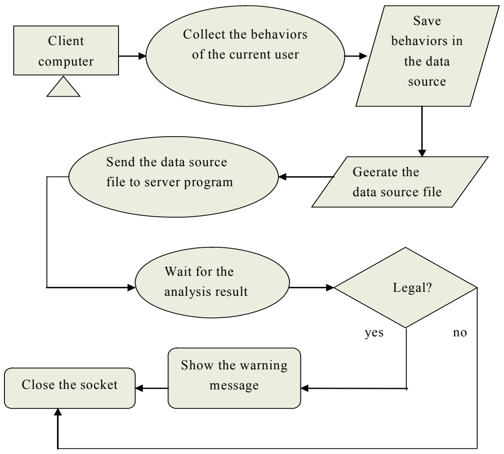

Client computer
Collect the behaviors of the current user
Save behaviors in the data source
Gerate the data source file
Send the data source file to server program
Wait for the analysis result
Legal?
yes
no
Show the warning message
Close the socket

Fig. 1. The flow chart for the client program

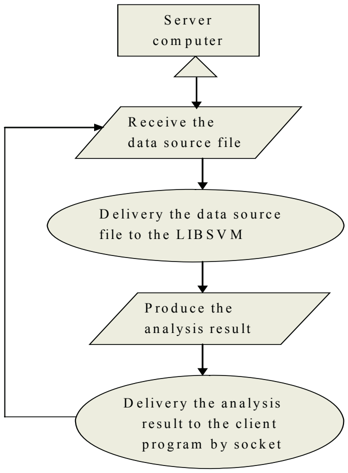

Server computer
Receive the data source file
Delivery the data source file to the LIBSVM
Produce the analysis result
Delivery the analysis result to the client program by socket

Fig. 2. The flow chart of the server program

When the server program receives the data source file, LIBSVM will analyze the data based on the trained model. The analyzed result will be delivered to the client program by the server program. Fig. 2 illustrates the flow of the server program.

3.2 Genetic Algorithm

Genetic algorithm (GA) will be used to improve the performance of the IDS in the proposed scheme. The fourth type of behaviors collected is on the executed application programs. The performance of IDS will be downgraded if too many application programs are detected. Therefore it would be desirable to filter out the insignificant application programs (ones that will not affect the status of a user). If these application programs can be filtered out in advance, a new more efficient model could be trained for analyzing users.

Encoding for GA. Suppose the number of application programs detected by IDS is N. A binary string of length N is generated. Each bit represents a detected application.

Initiate the Population. In the binary string, the bit with value 1 implies that the detected item is to be preserved and 0 implies to be deleted. In addition, the population size is set to M. There are M strings to be processed by GA. The number of the parent generations is M and the number of the offspring generations is M, too.

Fitness Function. The fitness function is used to improve the detection rate. Eq. (1) provides calculation with attributes: detection rate acc(x), error rate error(x), the number of the items deleted from the fourth behavior cost(x), and weights *Wacc*, *Werr*, *Wcost*.

$$\text{fitness}(x) = Wacc \times \text{acc}(x) + Wcost \times \text{cost}(x) - Werr \times \text{error}(x) \qquad (1)$$

Crossover and Mutation. The tournament selection is used here. Suppose the probability reproduction of *PoR*. There are (*PoR*×*M*) chromosomes that have the biggest fitness value in the crossover pool. GA will choose (1-*PoR*)×*M* strings from high to low values for the crossover pool.

After reproduction, there would be *M* strings in the crossover pool *M*/2 coupling of the chromosomes. The coupled crossovers will be based on the probability of crossover *PoC*. If coupling does not crossover, these two strings will remain unchanged. If crossover occurred, crossover is random.

The final step is to carry out mutation. The probability of mutation is set as *PoM*. There are *M*×*N*×*PoM* (round-off) bits to mutate.

Define the Stop Rule. The stop rule is used to limit repeats of genetic cycle. Suppose that the repeating times is *S*. The GA process will stop at *S* repeated genetic cycle. The biggest fitness value will result in the best solution for the *S* child generations. The new data source will be used for training the model for analyzing the user's status. The new model will improve the performance of the IDS.

4 Experimental Results

Two sets of experiments were performed; where one is for the original IDS with one run and the second one is for the IDS with GA for 5 runs. The experimental site is at the laboratory of National Taichung Institute of Technology. The client program is set to record user behavior every 5 minutes interval. If the interval is too short, the server program is unable to detect the behavior because LIBSVM requires time to analyze the data source file. If the interval is too long, the intruder gets away from the client computer before a warning message could be displayed.

Information on behavior is collected for the administrator. The total number of behaviors collected was 1248. The number of the behaviors that do not belong to administration was 2381. Therefore, the total number of behaviors collected was 3629. A program, which was installed in the client end, was used to collect the behaviors. The behaviors were input to LIBSVM to train a model which can be used to analyze whether the user's status in the server is administrator.

Each behavior of the administrator is stored as one data source file. Therefore, for every data source an additional illegal data source file will be generated as test attacks for the proposed IDS scheme.

The original number of items to analyze is 77. The fourth to seventieth items belong to the fourth type application program. Tests were performed for both the legal and illegal data source files for the proposed IDS scheme. The experimental results are as shown in Table 1.

Table 1. The comparison of performance with GA and without GA

	Illegal behavior detection rate	Average error rate	Number of analysis items
Without GA	90 %	10 %	77
With GA	95 %	2.5 %	51

The illegal behavior detection rate is 90% and the average error rate is 10% before the employment of GA. To improve the detection rate and reduce the error rate, we propose to remove non-useful information from popular programs, such as IE, Word, Dreye, winamp, ftp and so on, which are used by both the administrator and the user. Detecting these application programs will downgrade the performance.

After using GA to delete the fourth type analysis items that are not so important, the performance of the proposed IDS was improved, as shown in Table 1. The illegal behavior detection rate improved from 90% to 95% and the average error rate decreased from 10% to 2.5%. The GA process had apparently filtered out some non-useful fourth type analysis items. The non-useful fourth type analysis items are the application programs frequently executed neither by the administrator nor another user, or executed concurrently by the administrator and another user. These application programs are IE, Word, System, Idle and more. There are 26 fourth type analysis items deleted with 51 remaining. There are two categories to the remaining analysis items. The first category is the application programs frequently executed by the administrator while another user does not frequently execute these programs; e.g. vb.net. However, the second category is the exact opposite. That is the administrator does not frequently execute the application programs while another user frequently executes them.

5 Conclusions

The proposed IDS scheme can be improved by directly collecting the user's behaviors and then apply SVM to determine the status of the user. It can avoid tamper attacks on the data source files since they were generated by IDS. In addition, GA can be applied to improve the performance of IDS. From the experimental results, GA can improve the detection rate, reduce the error rate and remove non-useful analysis items. Processing time by IDS is optimized.

References

1. Denning, D.E.: An Intrusion Detection Model. IEEE Trans. Soft. Eng., (SE-13), **2** (1987) 222-232
2. Helmer, G., Wong, J.S.K., Honavar, V., and Miller, L.: Automated Discovery of Concise Predictive Rules for Intrusion Detection, J. Sys. & Soft. (2002) 165-175
3. Jha, S., and Hassan, M.: Building Agents for Rule-based Intrusion Detection System," Comp. Comm. (2002) 1366-1373
4. Graham, R.: http://www.robertgraham.com/pubs/network-intrusion-detection.html, March (2000)
5. Forouzan, B.A.: TCP/IP Protocol Suite. McGraw Hill Pub (2000)
6. Schwenker, F.: Hierarchical Support Vector Machines for Multi-Class Pattern Recognition. IEEE Knowl.-Based Intel. Eng. Sys. & Allied Tech., **2** (2000) 561-565
7. Hsu, C.W., and Lin, C.J.: A Comparison of Methods for Multi-Class Support Vector Machines. IEEE Trans, Neural Networks, **2** (2002) 415-425
8. http://www.csie.ntu.edu.tw/~cjlin/libsvm/
9. Michalski, R.S., Bratko, I., and Kubat, M.: Machine Learning and Data Mining: Methods and Applications, Wiley Pub (1998)
10. Man, K.F., Tang, K.S., and Kwong, S.: Genetic Algorithms – Concepts and Design, Springer (1999) 1-30

Fusions of GA and SVM for Anomaly Detection in Intrusion Detection System

Dong Seong Kim, Ha-Nam Nguyen, Syng-Yup Ohn, and Jong Sou Park

Computer Engineering Department, Hankuk Aviation University
{dskim,nghanam,syohn,jspark}@hau.ac.kr

Abstract. It is important problems to increase the detection rates and reduce false positive rates in Intrusion Detection System (IDS). These problems can be viewed as optimization problems for features and parameters for a detection model in IDS. This paper proposes fusions of Genetic Algorithm (GA) and Support Vector Machines (SVM) for efficient optimization of both features and parameters for detection models. Our method provides optimal anomaly detection model which is capable to minimize amounts of features and maximize the detection rates. In experiments, we show that the proposed method is efficient way of selecting important features as well as optimizing the parameters for detection model and provides more stable detection rates.

1 Introduction

In this paper, we propose fusions of Genetic Algorithm (GA) and Support Vector Machines (SVM) for anomaly detection in Intrusion Detection System (IDS). SVM is relatively a novel classification technique and has been shown higher performance than traditional learning methods in many applications such as bioinformatics and pattern recognitions [16]. Therefore, several researchers have applied SVM to IDS in network security field. Fugate *et al.* [4] have adopted SVM for anomaly detection in computer intrusion detection. Kim *et al.* [8], [14] have also proposed host and network-based IDS using both SVM and data-mining techniques. Mukkamala *et al.* [11] have also applied SVM to network-based IDS and it has shown better performance than neural network. Hu *et al.* [5] have proposed host-based anomaly detection method using RSVM. Despite the advantages of SVM based IDSs in terms of detection rates, there are still rooms for further enhancement. First, SVM based IDSs need to select optimal features among whole feature of audit data. When the numbers of features of audit data become very large, the detection rates of IDS can be degraded because it should process the large number of features of vast amount of audit data. For this reason, Mukkamala *et al.* [12] have tried to figure out important features so as to minimize processing burden and maximize detection rates. Furthermore, as networks become faster, there is an emerging need for security analysis techniques that will be able to keep up with the increased network throughput [9], in this sense, to select *"optimal features"* among whole features is very important. Second, SVM based IDSs need to not only increase detection rates but also guarantee the stability for detection rates. In order to provide these properties, one has to optimize the parameters for kernels in SVM. This paper proposes a fusion method to optimize SVM based IDS through operation of GA. GA is not only one of the most powerful tools for searching in large search space but also imposes few mathematical constraints in the shape of the function to optimize [10]. Our method is different from the feature selection methods based on GA and SVM recently reported [1], [3] in that they used GA only for feature selection and then manually chose the parameters for a kernel function. But our proposed approach simultaneously enables one not only to select *"optimal features"* but also to figure out "*optimal parameters*" for SVM classifier. Additionally, our method guarantees stable detection rates. By exploiting a fusion of GA and SVM, our method is able to minimize the overheads due to the large numbers of features for audit data and maximize detection rates of SVM based IDS. We carry out several experiments on proposed method using KDD 1999 CUP dataset in intrusion detection competition and demonstrate the feasibility of our approach.

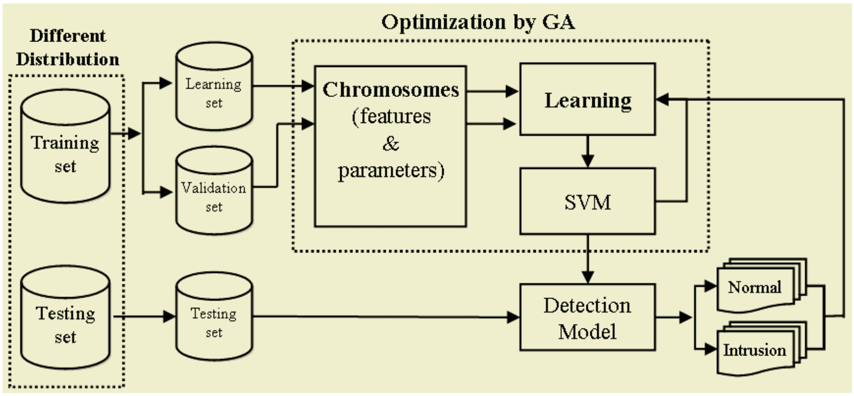


Fig. 1. Overall Structure of Proposed Method

2 Proposed Method

The overall structure and main components of proposed method are depicted in Figure 1. GA builds new chromosomes and searches the *optimal detection model* based on the fitness values (in this work, we use detection rates as fitness values) obtained from the result of SVM classification. A chromosome is decoded into a set of features and parameters for a kernel function to be used by SVM classifier. The SVM is used to estimate the performance of a detection model represented by a chromosome. In order to prevent overfitting problems, *n*-way cross-validation is used and the detection rates acquired as the results of *n* tests are averaged so as to obtain a fitness value.

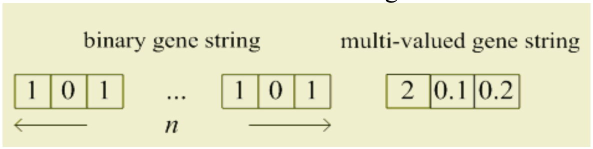


Fig. 2. Structure of a chromosome used in GA procedure

According to *no free lunch theorem* [2] on machine learning, there is no superior kernel function in general, and the performance of a kernel function rather depends on applications. Also, the parameters in a kernel function play the important role of representing the structure of a sample space. The optimal set of the parameters maximizing the detection performance can be selected by a machine learning method. In a learning step, the structure of a sample space is learned by a kernel function, and the knowledge of a sample space is contained in the set of parameters. Furthermore, the optimal set of features also should be chosen in the learning step. In our method, GA technique is adopted to obtain the set of features as well as the optimal parameters for a kernel function. Simulating a genetic procedure, GA creates improved detection models containing a set of features and parameters by the iterative process of reproduction, evaluation, and selection process. At the end of learning step, this method builds the *optimal detection model* consists of a set of features and parameters for a kernel function. The *optimal detection model* is used to classify new pattern samples in classification step [13]. To evaluate the feasibility of our approach, several experiments are carried out using KDD Cup 1999 data [6], [7]. The following section presents the results of experiments and their analysis.

3 Experiments

3.1 Experimental Dataset and Environments

This paper used KDD Cup 1999 data [6], and Stolfo *et al.* [7] defined higher-level features that help in distinguishing normal connections from attacks. We randomly extracted samples for building learning set, validation set and test set from training dataset and testing set respectively. We only used Denial of Service (DoS) type attacks among whole attacks [13]. In learning and validation step, we randomly split the set of labeled training sample into two parts: learning set is used as the traditional training set for adjusting the parameters in SVM. Validation set is used to estimate the generalization error during learning step. In order to achieve both low generation error and high detection rates, we adopted simple generalization method: 10-fold cross validation with 2500 samples. We estimate how well this built model copes with novel attacks since there are 14 new attacks in testing set which are not included in training set. The detection rates indicate the accuracy of classification. For genetic procedure, we used tournament method for selection process.

3.2 Experiments and Analysis

The results of cross validation are shown in Figure 3. Because of characteristics of GA, the detection rates monotonously increase when all of three kernel functions are used in learning step. While GA was executed for 20 generations, the learning process using SVM with neural kernel function achieved the highest detection rates.

And the number of features and parameters became optimized as the results of genetic procedure. These features can be considered as most important features to detect DoS type attacks. The optimal parameters are different according to the kernel function in SVM classifier. The result of all experiments is summarized in Figure 4. Most of the classification problems using SVM, radial kernel is selected and has showed a good performance, however, in our experiments indicate that neural kernel function for SVM shows better results. The reason why is that, according to *no free lunch*

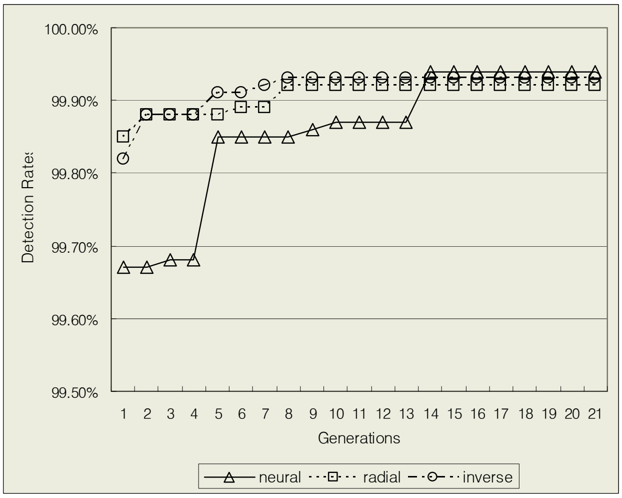


Fig. 3. The results of cross validation: detection rates versus generations of GA procedure

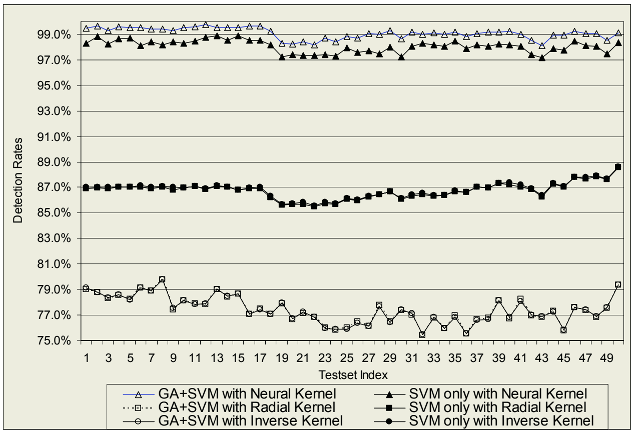


Fig. 4. Testing results: detection rates versus test set indexes

theorem [2] on machine learning, there is no superior kernel function in general, and the performance of a kernel function rather depends on applications. In our experiments, it is most effective for detecting DoS types of attack to use neural kernel function in SVM and such optimal features selected by GA operation. In case of detection model using neural kernel function can provide at least 98% detection rates. And it also shows the possibility that can cope with novel attacks since it shows detection rates more than 98% during testing step. Because testing set includes 14 additional attack types which are not included in training set. In other words, our method has the potential of detecting previously unknown DoS types of attacks. And our method provides higher detection rates than the approaches that only adopt SVM for IDS. The detection rates of our method shows the better performance than the KDD '99

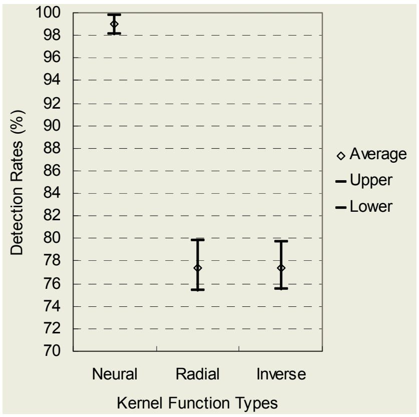


Fig. 5. Detection Rates for each kernel function with feature selection in testing step.

contest winner [15]. Moreover, according to Figure 5, our method guarantees the stability for detection rates. Our method shows the feasibility that is not only able to select *"optimal feature set"* for audit data but also figure out *"optimal parameters"* for a kernel function in SVM based detection model. Therefore, our method can be effective for optimizing IDS based on SVM classifier.

4 Conclusions

This paper proposed a fusion method for optimizing IDS. Several researchers have proposed SVM based IDSs [4], [5], [8], [11], [14], however, SVM based IDSs have still rooms for improvement. We adopted GA which is able to provide fast and excellent optimization. By using GA for both feature selection and optimization of parameters for SVM classifier, our method is not only able to select *optimal features* but also find out *optimal parameters* for a kernel function in SVM. So our method improves previously proposed SVM based IDS. We demonstrated the feasibility of proposed method by carrying out several experiments on KDD Cup 1999 intrusion detection dataset. The experimental results showed that our method guarantees stability for intrusion detection rates and has the potential of detecting previously unknown types of attacks.

Acknowledgements

This research was supported by the MIC (Ministry of Information and Communication), Korea, under the ITRC (Information Technology Research Center) support program supervised by the IITA (Institute of Information Technology Assessment).

References

1. Chen, X.: Gene Selection for Cancer Classification Using Bootstrapped Genetic Algorithms and Support Vector Machines. The Computational Systems Bioinformatics Conference (2003) 504–505.

2. Duda, R.O., et al.: Pattern Classification. 2nd edn. Wiley Interscience Inc (2001)
3. Frohlich, H., et al.: Feature Selection for Support Vector Machines by Means of Genetic Algorithm, Tools with Artificial Intelligence. (2003) 142–148.
4. Fugate, M., et al.: Anomaly Detection Enhanced Classification in Computer Intrusion Detection. Lecture Notes in Computer Science, Vol. 2388. Springer-Verlag, Berlin Heidelberg New York (2002) 186–197
5. Hu, W., et al.: Robust Support Vector Machines for Anomaly Detection in Computer Security. Proc. of Int. Conf. on Machine Learning and Applications 2003, CSREA Press (2003) 168–174
6. KDD-CUP-99 Task Description: http://kdd.ics.uci.edu/databases/kddcup99/task.html
7. KDD Cup 1999 Data.: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html
8. Kim, D.S., Park, J.S.: Network-based Intrusion Detection with Support Vector Machines. Lecture Notes in Computer Science, Vol. 2662. Springer-Verlag, Berlin Heidelberg New York (2003) 747–756
9. Kruegel, C., et al.: Stateful Intrusion Detection for High-Speed Networks, Proc. of the IEEE Symposium on Research on Security and Privacy (2002) 285–293
10. Mitchell, M.: Introduction to Genetic Algorithms, MIT press (1999)
11. Mukkamala, S., et al.: Intrusion Detection Using Neural Networks and Support Vector Machines. Proc. of IEEE Int. Joint Conf. on Neural Networks (2002) 1702–1707
12. Mukkamala, S., Sung, A.H.: Feature Selection for Intrusion Detection Using Neural Networks and Support Vector Machines. Proc. of the 82nd Annual Meeting of the Transportation Research Board, National Academics (2003)
13. Ohn, S.-Y., et al.: Determining Optimal Decision Model for Support Vector Machine by Genetic Algorithm. Lecture Notes in Computer Science, Vol. 3314. Springer-Verlag, Berlin Heidelberg New York (2004) 895–902
14. Park, J.S., et al: Using Support Vector Machine to Detect the Host-based Intrusion IRC Int. Conf. on Internet Information Retrieval (2002) 172–178
15. Pfahringer, B.: Winning the KDD99 Classification Cup, http://www.ai.univie.ac.at/~bernhard/kddcup99.html (1999)
16. Vapnik, V.: The Nature of Statistical Learning Theory. Springer, Berlin Heidelberg New York (1995)

A Genetic SOM Clustering Algorithm for Intrusion Detection

Zhenying Ma

Department of Computer Science and Engineering, Chongqing University,
Chongqing 400044, China
mazhenying@vip.sina.com, xfliao@cqu.edu.cn

Abstract. By combining SOMs network and genetic algorithms, a genetic SOM (Self-Organizing Map) clustering algorithm for intrusion detection is proposed in this paper. In our algorithms, genetic algorithm is used to train the synaptic weights of SOMs. Computer experiments show that GSOMC produces good results on small data sets. Some discussions of the number of clusters *K* and future work is also given.

1 Introduction

The previous works in intrusion detection mainly focused on learning knowledge from the labeled data [1]-[4]. When detecting unknown attacks, these algorithms need to be retrained with new labeled samples. Unfortunately, it is expensive to label huge amount of audit or network data. In general, clustering is the unsupervised classification of input items into groups (clusters) without any prior knowledge [5]. It is promising to detect unknown attacks in intrusion detection automatically. Furthermore, the clustered data can be analyzed for more information, i.e. the signature of new attacks.

Self-organizing neural networks represent a type of useful clustering techniques by competitive learning. In intrusion detection, SOMs are also proved to be promising techniques [7]-[11]. Unfortunately SOM clustering can be viewed as *K*-means types and it is a hill-climbing method [6]. It depends largely on a selection of the initial synaptic weights and the input patterns, otherwise it fails to converge or converge to a local optimum. To solve the above problem, a genetic SOM clustering (GSOMC) method is proposed in this paper. In GSOMC, GAs are used to optimize the synaptic weights of SOM neural network, i.e., SOM adjusts synaptic weights by GAs operations instead of the classical learning rule. The computer experiments show that the GSOMC produces very good results.

The remaining parts of this paper are arranged as follows. Section 2 presents the related work of intrusion detection based on SOMs. The proposed GSOMC method is described in Section 3. The experimental results on 1999 KDD cup data and some discussions of choosing the number of clusters are in Section 4. Some conclusions and future work are given in Section 5.

2 Related Works

A prototype UNIX anomaly detection system is proposed in [8], which monitors host user behaviors based on user profiles established by SOMs. In [7], training and testing data extracted from log files of an UNIX host are used and the representation of time, network parameters and SOM architecture are discussed. Hierarchical SOM approach for intrusion detection is also proposed in [12]. In [10], SOMs are used to map the network connections onto 2-dimentional surfaces, which were displayed to the network administrator. The SOMs are trained by normal traffic in [11]. Based on SOMs and MLP Multilayer Perceptron, a hybrid model was proposed in [9]. The shortages of these methods are that SOM clustering fails to converge or converge to a local optimum. It is possible to view the clustering problem as an optimization problem that finds an optimal partition of the inputs. Hence, the use of optimization (search) algorithms in SOMs clustering is a good choice [13].

3 Methodologies

3.1 Chromosome Coding

In the proposed method, the chromosome structure is constructed according to the representatives of SOMs, i.e. a chromosome presents a possible combination of synaptic weights of neurons in SOMs. Let the number of the input items, the chromosome length (number of clusters) and the population length equal to *N*, *K* and μ respectively. Then the chromosome set *P* (population) is described as $P = \{p_{ij} \mid i = 1 \cdots \mu, j = 1 \cdots K\}$. The chromosome are coded with decimal integers in [1, *N*], i.e. $1 \le p_{ij} \le N$ which presents a candidate solution of representatives of SOMs, i.e., let the *K* equal to 3, $p_i = \{108{,}8{,}25\}$ which presents that the i^{th} solution is the No.108, No.8, and No.25 session in the input data set are selected as representatives and other sessions will be clustered around them..

3.2 Chromosomes Initiating

Typical GA usually begins with randomly generated chromosomes. Consider SOMs clustering, using the presentation of chromosome structure described above, the search space of GA is N^K. When the *N* is a large number, this is too huge to handle even for GA. So in the proposed method the generation of the initial population is controlled to promise that two candidates are completely different, i.e. $\forall p_i, p_j \in P, p_i \cap p_i = \varnothing$.The aim to do this control is to include as much as possible different representatives in the beginning of search process in order to find the optimum as fast as possible. Generation Maximum g is set to 30, the crossover rate δ is set to 0.8, mutation rate η is set to 0.01, and population μ is set dynamically. Let the number of the input items equal to *N*, Let the chromosome length equal to *K*, the population is computed as an integer near to $\frac{N}{K}$. Because, $\forall p_i, p_j \in P, p_i \cap p_i = \varnothing$, most of the sessions in the data set can be included in initial solutions as representatives, i.e. most of the sessions have chances to be representatives. By this way the representatives in one solution are not distributed distinctly but in the whole population they do. Therefore though crossover and mutation operation, the optimum can be found quickly.

3.3 Fitness Evaluation

3.3.1 Genetic SOMs

The genetic SOMs uses *K* representatives coded in one chromosome as synaptic weight vectors $\{w_i \mid i = 1 \cdots K\}$, i.e. let $p_i = \{108, 8, 25\}$, this presents that the i^{th} solution is that the No.108, No.8, and No.25 session in the input data set are selected as representatives and other sessions will be clustered around them. Then this chromosome is sent to SOMs, and it is explained with synaptic weight vectors set to be $w_i = \{x_{ij} \mid i = 108, 8, 25, j = 1 \cdots p\}$ where *p* denotes the dimensionality of observation. And the intrusion detection data sets are the input observation vectors $x(n) = (x_1(n), \cdots x_p(n))^T$. The output of the variant SOMs is the results of assignments of the inputs to one of the clusters that representatives presents by competitive learning, i.e. $y(x(n)) = c$ where c= $\arg\min_i \|x - w_i\|$ with $\|.\|$ denoting the Euclidean distance between any two vectors. The results of SOMs are sent to the evaluation function $f(p_i)$ to compute the fitness of the chromosome. GA is used to help SOMs to learn until the optimum is find at last, i.e. the optimum $\{w_i \mid i = 1, \cdots, K\}$ is improved by evolutionary strategies such as crossover and mutation of the chromosomes.

3.3.2 Fitness Function

The fitness function represents the goodness degree of a solution and it decides the probability that this solution is selected to the next generation. In GSOMC, the fitness is computed by sum-of-squares function which presents the square root of the sum of the distances between all the inputs and their nearest representatives, i.e. let $p_i = (p_{ij} \mid j = 1 \cdots K\}$ be a chromosome string, which denote *K* representatives (clusters or patterns), let $x(n) = (x_1(n), \cdots x_p(n))^T$ $n = 1 \cdots N$ be the input data set, $y(x(n)) = c$ denote the assignment of one of the *K* clusters by SOMs, the fitness function is described as follows.

$$f(p_i) = \frac{1}{sqrt(\sum_{n=1}^{N} \|x(n) \ - y(x(n) \ )\|^2)}. \tag{1}$$

Eq.(1) shows that more close *K* representatives in one chromosome are to the inputs, higher the fitness and this chromosome can achieve.

3.4 Crossover Operator

Crossover is a probabilistic process that exchanges information between two parents for generating two children. In GSOMC, the single-point crossover with a fixed crossover probability is used. For chromosomes of length *K*, a random integer, called the crossover point, is generated in the range [1, *K*]. The portions of the chromosomes lying to the right of the crossover point are exchanged to produce two offspring.

3.5 Mutation Operator

Mutation is also a probabilistic process that takes a single parent and modifies some genes in the chromosome. In GSOMC, the change is the replacement of old representatives by new ones in a random location with fixed probability. Here new representatives refer to those that are never selected in any initial individuals.

3.6 Stop Criterion

The GSOMC will be terminated if $g > G_{\max}$ or the representatives do not change any more (the algorithm converges to an optimum).

4 Experiment Results and Discussion

4.1 Data Formalization

The KDD Cup 1999 Data [14] is used as the experimental data set There are totally 41 features of each session in KDD Cup 99, and they are on very different scales. Let *A*= (2, 5679, 0.5), *B*= (6, 3427, 0.6), under Euclidean metric, the distance of *A* and *B* will be dominated by the second feature. Furthermore, some features are STRING types. Therefore, in GSOMC, we divide 41 features into 4 categories (‘Boolean’, ‘String’, ‘Count’, ‘Rate’) and each category of features are pre-processed differently.

Table 1. 4 categories of 41 features of KDD-99 session

boolean	string	count	Rate
land	protocol_type	duration	serror_rate
logged_in	service	src_bytes	srv_serror_rate
su_attempted	flag	dst_bytes	rerror_rate
is_host_login		wrong_fragment	srv_rerror_rate
is_guest_login		urgent	same_srv_rate
root_shell		hot	diff_srv_rate.
		num_failed_logins	srv_diff_host_rate
		num_compromised	dst_host_same_srv_rate
		num_root	dst_host_diff_srv_rate
		num_file_creations	dst_host_same_src_port_rate
		num_shells	dst_host_srv_diff_host_rate
		num_access_files	dst_host_serror_rate
		num_outbound_cmds	dst_host_srv_serror_rate
		count	dst_host_rerror_rate
		srv_count	dst_host_srv_rerror_rate
		dst_host_count	
		dst_host_srv_count	

Category ‘Boolean’ is 0 for ‘no’ and ‘1’ for ‘yes’; ‘Count’ is integer number and ‘Rate’ is float number scale on [0,1]. For 'count' category, data are formalized so that the values are converted onto [0, 1] scale. Let the input data set be $X = \{x_{ij} \mid i = 1 \cdots N, j = 1 \cdots P\}$, where *N* is the number of total sessions and *P* is the number of features, which is 41 in KDD cup 99 data sets. Let α^{th} feature be 'Count' type, and the data formalization of α^{th} feature is computed as E.q(2), For 'Rate' and

'Boolean' category, the data remain unchanged. Features of 'string' category are not used directly to evaluate the Euclidean distances between sessions, but to analyze features of clusters.

$$x'_{\alpha j} = \frac{x_{\alpha j}}{\max\limits_{j=1}^{P}(x_{\alpha j})}. \tag{2}$$

4.2 Detection Results

The explanation of Fig.1 is that a large *K* presents a wide range selection of representatives (cluster samples) which leads to more precise clustering than a small *K*. Furthermore, because the number of normal sessions is much larger than abnormal sessions (10 times), if *K* is too small, the abnormal sessions will have little chances to be representatives. This is why the detection rate is only about 35% when *K* is 5. GSOMC gets very low false-positive rate (below 5%) in spite of the value of *K*. The explanation is that the number of normal sessions is much larger than abnormal sessions (10 times), so in *K* representatives the normal ones always take advantage over abnormal ones in quantity. That is why the normal sessions can be easily classified into clusters. Sometimes the results are irregular and hard to explain. For example, when *K*=50, detection rates of data set 1 and 4 are lower than *K*=40, and when *K*=20, false positive rates of data set 1 and 2 are very high while they come very low when *K*=25. We can only explain this as that the GAs are random search algorithms, and the final results depend heavily on the initial selection of solutions (chromosomes) which is why the results vary even for the same *K* value and same data set.

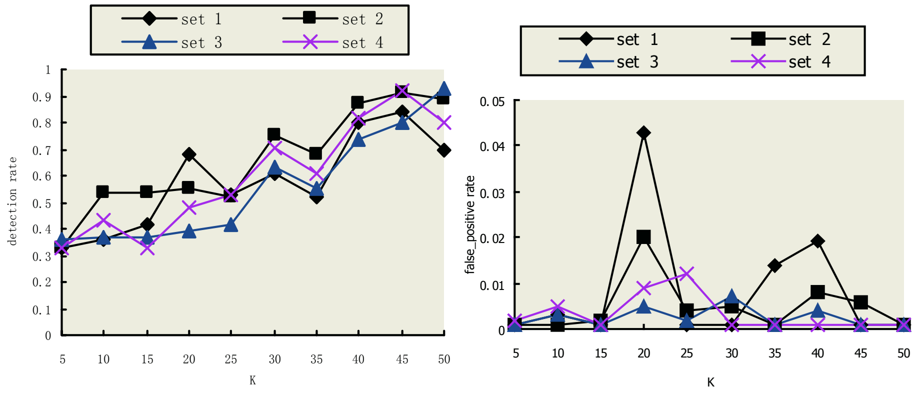


Fig. 1. Detection rate and False positive rate of 4 test data sets

4.3 *K* Representatives

The GSOMC tries to find the optimal *K* representatives in the data set and then cluster other sessions around them. This is similar to *K*-medoids clustering except that GAs are used to optimize the solution. Table 2 records the optimum solution found by GSOMC when *K* is set to 10, 20, 30 and 40 for data set 2. They are good explanation of results that. a larger *K* leads to better clustering results. Assume that in 40 representatives automatically selected by GA 32 abnormal types and normal type are all included, GSOMC may produce a perfect result with a detection rate near 100% and a false positive rate near 0%. Actually this is difficult for 2 reasons. First, because GA is limited for it is a type of random algorithm and the design for the fitness function, the evolution operators and parameters are complex. And second, in the KDD-99 data set, some attack types are very similar (close in Euclidean distance) to normal ones, i.e. they are stealthy attacks that last for a long time period and are not significantly different from normal behavior. It is hard to distinguish these abnormal data from normal data only based on Euclidean distance measure. Hybrid detection techniques need to be developed.

Table 2. Optimum solutions of data set 2

K	10	20	30	40
number of representatives	10	20	30	40
number of clusters	10	20	30	40
number of normal representatives	5	12	20	28
number of abnormal representatives	5	8	10	12
detection rate	54%	55%	75%	80%
false positive rate	0.3%	4.3%	0.1%	1.9%

5 Conclusions

By combining neural network with genetic algorithm, GSOMC has been proposed and proved that it is an efficient clustering method for intrusion detection by experiments on small-scale data sets. But it is still to be improved to adapt to large-scale ones, i.e. to cluster more than 10,000 sessions. The need of pre-defining number of clusters, i.e., *K*, is also a problem. Density-based clustering is a promising way to decide *K* automatically. Furthermore, after clustering process completes, how to explain the clusters with domain knowledge is to be discussed.

Acknowledgements

The work described in this paper was supported by grants from the National Nature Science foundation of China (No. 60271019), the Doctorate Foundation Grants from the Ministry of Education of China (No. 20020611007) and the Natural Science Foundation of Chongqing (No. 8509).

References

1. Lee, W.: A Data Mining Framework for Constructing Features and Models for Intrusion Detection Systems. Ph.D. Thesis, Columbia University, USA (1999)
2. Lee, W., Stolfo, S.J., Mok, K.: Data Mining in Work Flow Environments: Experience in Intrusion Detection. In Proceedings of the 1999 Conference on Knowledge Discovery and Data Mining (KDD-99), San Diego (1999) 114-124
3. Lee, W., Stolfo, S.J.: Data Mining Approaches for Intrusion Detection. In Proceedings of the 1998 USENIX Security Symposium, San Antonio (1998) 79-84

4. Eskin, E.: Anomaly Detection over Noisy Data Using Learned Probability Distributions. In Proceedings of the International Conference on Machine Learning, USA (2000) 255-262
5. Anderberg, M.R.: Cluster Analysis for Application. Academic Press, New York (1973)
6. Kohonen, T.: Self-Organizing Maps, 3rd Edition. Springer - Verlag, ISBN 3-540-67921-9 (2000)
7. Lichodzijewski, P., Zincir-Heywood, A.N., Heywood, M.I.: Host-based Intrusion Detection Using Self-Organizing Maps. IEEE International Joint Conference on Neural Networks (2002) 1714-1719
8. Hoglund, A.J.,Hatonen K,Sorvari A.S.: A computer Host Based User Abnormal Detection System Using the Self Organizing Map. Proceedings of the International Joint Conference on Neural Networks, IEEE IJCNN, **5** (2000) 411-416
9. Cannady, J.: Artificial Neural Networks for Misuse Detection. In Proceedings of the 1998 National Information Systems Security Conference (NISSC'98) (1998) 443-456
10. Girardin, L.: An Eye on Network Intruder Administrator Shootouts. In Proceedings of the workshop on Intrusion Detection and Network Monitoring (ID'99) 19-28
11. Ramadas, M., Ostermann, S., Tjaden, B.: Detecting Anomalous Network Traffic with Self-Organizing Maps. In Recent Advances in Intrusion Detection, 6th International Symposium (2003) 36-54
13. Babu, G.P., Murty, M.N.: Clustering with Evolution Strategies. Pattern Recognition, **27** (1994) 321-329
14. KDD-99 Cup data set. http: // kdd.ics.uci.edu / databases /kddcup99/kddcup99.html (1999)

Intrusion Detection Based on Dynamic Self-organizing Map Neural Network Clustering

Yong Feng, Kaigui Wu, Zhongfu Wu, and Zhongyang Xiong

College of Computer Science and Technology, Chongqing University,
Chongqing 400030, China
fengyongcqu@yahoo.com.cn

Abstract. An approach to network intrusion detection is investigated, based on dynamic self-organizing maps (DSOM) neural network clustering. The basic idea of the method is to produce the cluster by DSOM. With the classified data instances, anomaly data clusters can be easily identified by normal cluster ratio. And then the identified cluster can be used in real data detection. In the traditional clustering-based intrusion detection algorithms, clustering using a simple distance-based metric and detection based on the centers of clusters, which generally degrade detection accuracy and efficiency. Our approach based on DSOM clustering can settle these problems effectively. The experiment result shows that our approach can detect unknown intrusions efficiently in the real network connections.

1 Introduction

Anomaly detection problem can be considered as a two-class classification problem (normal versus abnormal) where samples of only one class (normal class) are used for training. Basically there are three different approaches for anomaly detection: negative characterization [1],[2], positive characterization [3],[4],[5], and artificial anomaly generation [6],[7].

Clustering techniques have been applied successfully to the anomaly detection problem. However, in the traditional clustering-based intrusion detection algorithms, clustering using a simple distance-based metric and detection based on the centers of clusters, which generally degrade detection accuracy and efficiency. In this paper, we present a new type of intrusion detection algorithm, intrusion detection based on dynamic self-organizing maps (DSOM) clustering, to address these problems. The algorithm has three main stages: clustering, labeling clusters, and detection. A spread factor is used in the stage of the clustering, which can control the accuracy of clustering and achieve hierarchical clustering. In the stage of detection, the usage of posteriori probabilities makes the detection independent of the centers of the clusters, so that the detection accuracy and efficiency can be improved. The experiment result shows that our approach can detect unknown intrusions efficiently in the real network connections.

The remainder of the paper is organized as follows. Section 2 presents the detailed algorithms of our approach. Experiment results are reported in section 3. Finally, concluding remarks are made in sections 4.

2 Intrusion Detection Based on DSOM Clustering

Unsupervised anomaly detection algorithms make two assumptions about the data which motivate the general approach [8]. The first assumption is that the number of normal instances vastly outnumbers the number of intrusions. The second assumption is that the intrusions themselves are qualitatively different from the normal instances. Our approach based on the two assumptions has three main steps:

Step1: DSOM clustering
Step2: Labeling clusters
Step3: Detection

2.1 DSOM Clustering Algorithm

The DSOM determines the shapes as well as the size of network during the training of the network, which is a good solution to problems of the SOM [9].

The DSOM is an unsupervised neural network, which is initialized with four nodes and grows nodes to represent the input data [10]. During the node growth, the weight values of the nodes are self-organized according to a similar method as the SOM. The **DSOM clustering algorithm** is described as follows:

1) Initialization phase.
   a) $V' = \{v'_1, v'_2, \ldots\ldots, v'_n\}$ is the input vector sets, and $v'_i = \{x'_{i1}, x'_{i2}, \ldots\ldots, x'_{id}\}$, where $1 \leqq i \leqq n$ and d is dimension of the vector v'_i. Standardizing V' to V, if v'_i is a continuous variable, the method can be described by [11]:

$$x_{ij} = \frac{x'_{ij} - \min(x'_{ij})}{\max(x'_{ij}) - \min(x'_{ij})}, \text{ where } 1 \leqq i \leqq n \text{ and } 1 \leqq j \leqq d. \tag{1}$$

   or we may code v'_i according to binary code, and then processing like (1).
   b) Initialize the weight vectors of the starting nodes with random numbers between 0 and 1.
   c) Calculate the growth threshold (GT) for the given data set.
2) Growing phase.
   a) Present input to the network.
   b) Find the winner neural b such that $\|v_i - w_b\| \leqq \|v_i - w_q\|\ \forall q \in N$,where v_i, w are the input and weight vectors, q is the position vector for nodes, N is the set of natural numbers, and $\|\cdot\|$ is Euclidean distance.
   c) Calculate the error distance E_i between b and v_i, E_i is defined as:

$$E_i = \sum_{j=1}^{d} (x_{i,j} - w_{b,j})^2. \tag{2}$$

   if $E_i \geqq GT$, then turn d) to grow nodes, or turn e) to adjust the weight value of neighborhood.
   d) Grow the new node m, and set $w_m = v_i$.
   e) The weight adjustment of the neighborhood can be described by:

$$w_j(k+1) = \begin{cases} w_j(k), j \notin N_{k+1}, \\ w_j(k) + LR(k) \times (x_k - w_j(k)), j \in N_{k+1}. \end{cases} \tag{3}$$

when $k \to \infty$ ($k \in N$), the learning rate $LR(k) \to 0$. $w_j(k)$, $w_j(k+1)$ are the weight vectors of the node j before and after the adjustment, and N_{k+1} is the neighborhood of the wining neuron at $(k+1)^{th}$ iteration.

f) Adjustment the learning rate according to the following formula:

$$LR(t+1) = LR(t) \times \alpha \,. \tag{4}$$

where α is the learning rate reduction and is implemented as a constant value $0 < \alpha < 1$.

g) Repeat steps b)–f) until all inputs have been presented and node growth is reduced to a minimum level.

3) Smoothing phase.
   a) Reduce learning rate and fix a small starting neighborhood.
   b) Find the winner and adjust the weights of winner and neighbors in the same way as in growing phase.
4) Produce clusters $\{C_1, C_2, \ldots\ldots, C_S\}$, where S is the number of the clusters.

The nodes generation of DSOM is decided by GT. Therefore, the map growth is depends on the GT value. For the node i to grow a new node, it is required that $0 \leqq GT \leqq E_i$. It can be deduced from (2), $0 \leqq GT \leqq E_i$ and $0 \leqq x_{ij}$, $w_{bj} \leqq 1$ that $0 \leqq GT < d$. Therefore, it becomes necessary to identify a different GT value for data sets with different dimensionality. This becomes a difficult task. The spread factor SF can be used to control and calculate the GT for DSOM. The GT can be defined as:

$$GT = d \times f(SF) \,. \tag{5}$$

where $SF \in R$, $0 \leqq SF \leqq 1$, and $f(SF)$ is a function of SF, which is identified as follows:

$$f(SF) = (1 - SF)^2 / (1 + 1/n(t)) \,. \tag{6}$$

where $n(t)$ is the total number of nodes at t^{th} iteration. **Fig. 1** shows that the different options available to the data analyst with the DSOM, due to the control achieved by the SF.

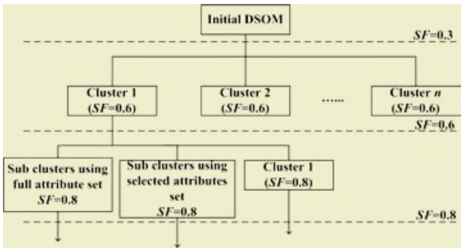


Fig. 1. Using DSOM for hierarchical clustering of a data set

2.2 Labeling Clusters Algorithm

Under the two assumptions of our approach it is highly probable that clusters containing normal data will have a much larger number of instances than would clusters containing anomalies. Therefore, the maximum quantitative difference D_i and the labeling clusters threshold N are defined to label the clusters. The D_i and N can be defined as:

$$D_i = (Q_i - Q_{min})^2/(Q_{max} - Q_{min})^2,\ N = SF/(1 + 1/S)\ . \tag{7}$$

where $0 \leqq D_i \leqq 1$, $0<N<1$. Q_i is the number of instances in C_i, $1 \leqq i \leqq S$. Q_{max} is the maximum of $\{ Q_i \}$. Q_{min} is the minimum of $\{Q_i\}$. S is the number of the clusters. The **labeling clusters algorithm** is described as follows:

1) Calculate the number of instances Q_i, D_i and N for $\{C_1, C_2, \ldots\ldots, C_S\}$.
2) Set $i = 1$, repeat the follows.
3) If $D_i > N$, then C_i is labeled as the 'normal' cluster, or C_i is labeled as the 'anomalous' cluster.
4) Set $i = i + 1$.
5) Until $i > S$.

The labeling clusters algorithm has a good accuracy because of D_i and N are decided by DSOM clustering rather than the parameters inputted by the users.

2.3 Detection Algorithm

Given a network data package X' which would be detected, standardizing X' to X. The X is assigned to the class label C_i such that $P(C_i|X)$ is maximal, $P(C_i|X)$ is a-posteriori probabilities [12]. The method is depends on Bayes theorem. Bayes theorem is defined as follows:

$$P(C_i|X) = P(X|C_i)\cdot P(C_i)\ /\ P(X). \tag{8}$$

where $P(X)$ is constant for all classes, $P(C_i)$ = relative freq of class C_i, C_i such that $P(X|C_i)$ is maximum = C_i such that $P(X|C_i)\cdot P(C_i)$ is maximum. The **detection algorithm** is described as follows:

1) Standardize X' to X.
2) Set $i = 1$, repeat the follows.
3) Calculate $P(X|C_i)\cdot P(C_i)$ for $\{C_1, C_2, \ldots\ldots, C_S\}$.
4) Let $p_i = P(X|C_i)\cdot P(C_i)$, where p_i is array variable.
5) Set $i = i + 1$.
6) Until $i > S$.
7) Let p_j = the maximum of $\{ p_i \}$, where $j \in [1,S]$.
8) If C_j is labeled as the 'normal' cluster, then X' is normal data package, or X' is intrusional data package.

Bayesian classification is the least fault rate rather than other classification algorithms. Detection algorithm of our approach depends on Bayes theorem, which makes the detection independent of the centers of the clusters, so that the detection accuracy and efficiency can be improved.

3 Experiment

The experimental data we used is the KDD Cup 1999 Data [13]. It consists of two datasets: training dataset (KDD-TND) and test dataset (KDD-TTD).

According to the first assumption of the unsupervised anomaly detection algorithms (UADA), we need to generate the training dataset *D* from KDD-TND by filtering it for attacks. *D* consisted of 1% to 1.5% intrusion instances and 98.5% to 99% normal instances.

To evaluate the algorithm we are interested in two major indicators of performance: DR (Detection Rate) and FPR (False Positive Rate). In the experiment, we adopt two test sets: KDD-TND and KDD-TTD. The results are reported in Table 1 and Fig. 2 shows the performance of our approach through the ROC curve.

In reference [14], the experimental data is also the KDD Cup 1999 Data. Table 2 shows the performance of the three algorithms proposed in [14] over KDD-TND. Fig. 3 shows the performance comparing between our approach and the existed UADA over KDD-TND through the ROC curve.

Table 1. The performance of our approach over KDD-TND and KDD-TTD

SF	KDD-TND		KDD-TTD	
	DR	FDR	DR	FDR
0.3	25.34%	0.67%	21.23%	0.11%
0.5	39.28%	6.17%	34.78%	0.91%
0.7	96.87%	9.58%	87.86%	3.41%
0.8	99.32%	13.89%	92.16%	3.67%
0.9	99.93%	20.10%	93.03%	8.24%

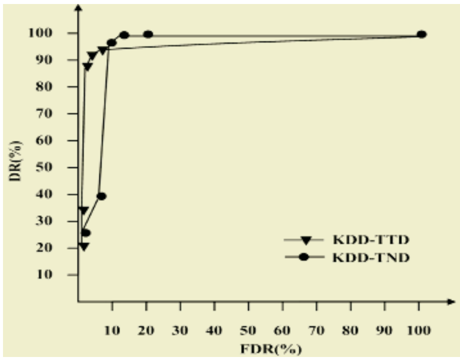


Fig. 2. ROC curves showing the performance of our approach over KDD-TND and KDD-TTD

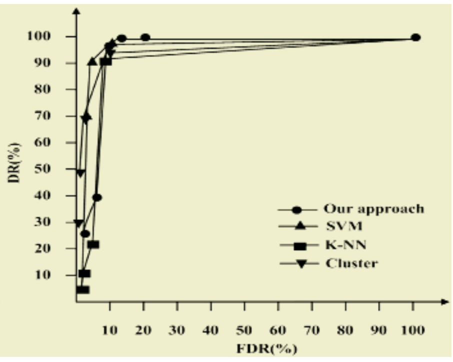


Fig. 3. ROC curves showing the performance comparing between our approach and the existed UADA over KDD-TND

Table 2. The performance of existed UADA over KDD-TND

Algorithm Name	DR	FPR
Cluster	93%, 66%, 47%, 28%	10%, 2%, 1%, 0.5%
K-NN	91%, 23%, 11%, 5%	8%, 6%, 4%, 2%
SVM	98%, 91%, 67%, 5%	10%, 6%, 4%, 3%

4 Conclusions

This paper proposes an unsupervised anomaly detection algorithm based on DSOM clustering and Bayesian classification. The experimental results show the performance of our approach is better than that of the existed algorithms when the detection rate is high.

Acknowledgements

The work described in this paper was partially supported by the 863 Program Project of China (Grant No. 2003AA116010).

References

1. Forrest, S., Perelson, A., Allen, L., Cherukury, R.: Self-Nonself Discrimination in a Computer. In Proceeding of the IEEE Symp. on Research in Security and Privacy (1994)
2. Singh, S.: Anomaly Detection Using Negative Selection Based on the Rcontiguous Matching Rule. In 1st International Conference on Artificial Immune Systems (2002) 99-106
3. Lane, T., Brodley, C.E.: An Application of Machine Learning to Anomaly Detection. In Proc. 20th NIST-NCSC National Information Systems Security Conference (1997)
4. Lane, T., Brodley, C.E.: Sequence Matching and Learning in Anomaly Detection for Computer Security. In AI Approaches to Fraud Detection and Risk Management (Fawcett, Haimowitz, Provost, Stolfo, eds.), AAAI Press (1997) 43-49
5. Mahoney, M., Chan, P.: Learning Nonstationary Models of Normal Network Traffic for Detecting Novel Attacks. In Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2002) 23-26
6. Fan, W., Lee, W., Miller, M., Stolfo, S., Chan, P.: Using Artificial Anomalies to Detect Unknown and Known Network Intrusions. In Proceedings of the 1st IEEE International Conference on Data Mining (2001)
7. Gonzalez, F., Dasgupta, D.: Neuro-Immune and Self-Organizing Map Approaches to Anomaly Detection: A Comparison, In 1st International Conference on Artificial Immune Systems (2002)
8. Portnoy, L., Eskin, E., Stolfo, S. J.: Intrusion Detection with Unlabeled Data Using Clustering. In Proceeding of ACM CSS Workshop on Data Mining Applied to Security, Philadelphia, PA (2001)
9. Kohonen, T.: Self-Organizing Maps, Springer-Verlag, Berlin (1995)
10. Alahakoon, L. D., Halgamuge, S. K., Srinivasan, B.: A Structure Adapting Feature Map for Optimal Cluster Representation. In Proc. Int. Conf. Neural Information Processing (1998) 809-812
11. Han, J., Kamber, M.: Data Mining: Concepts and Techniques. High Education Press, Morgan Kaufman Publishers (2001)
12. Duda, R., Hart, P.: Pattern Classification and Scene Analysis. New York: John Wiley & Sons (1973)
13. KDD99: KDD99 Cup Dataset. http://kdd.ics.uci.edu/databases/kddcup99 (1999)
14. Eskin, E., Arnold, A., Prerau, M.: A Geometric Framework for Unsupervised Anomaly Detection: Detecting Intrusions in Unlabeled Data, Published in Data Mining for Security Applications, Kluwer (2002)

Intrusion Detection Based on MLP Neural Networks and K-Means Algorithm

Hongying Zheng[1], Lin Ni[2], and Di Xiao[2]

[1] College of Computer Science and Engineering, Chongqing University, Chongqing 40030, China
zhenghongy@263.net
[2] College of Mechanic Engineering Chongqing University, Chongqing 40030, China

Abstract. A new intrusion detection technique to classify program behavior as normal or intrusive by using neural network and clustering pretreatment is presented in this paper. In our method, first, we divided the large samples space into subspace using *k*-means clustering; second, a set of neural networks are used to study the every subspace for intrusion detection separately. By this way, we can avoid some inherent problems of neural networks, such as the slow speed of convergence and the burden of computation; On the other hand, during subspace training, because program data, which are in the same subspace, have the similar behavior characters, neural networks can quickly recognize normal or anomalous area of input space; We also note that system call frequency is replaced of system call order in this method, program behavior is represented by frequencies of system calls; Experiment with 1998 DARPA BSM audit data has also shown that the method has good performance.

1 Introduction

Generally speaking, a computer system should provide confidentiality, integrity and assurance. However, due to the increasing connectivity and the vast financial possibilities that are opening up, more and more systems are prone to attack by intruders. These subversion motivations try to exploit flaws in the operating system as well as in application programs. Intrusion Detection Systems (IDS) extract information from a computer or a network of computers, and attempt to detect the presence of intrusions from external sources, as well as system abuses by authorized users [1]. Intrusion detection can be divided into two main categories: Misuse detection and anomaly detection. Misuse detection is trying to discover intrusions by searching for distinguishing patterns or signatures of known attacks, whereas anomaly detection is based on the assumption that intrusive behavior deviates significantly from previously learned normal behavior. Lots of other distinguishing features for IDS, for example the classification into host-based and network-based systems or online and offline systems, are set out in [4], [8], [9]

Many techniques have been applied to the intrusion detection, such as statistical approaches [2], [10], neural networks [3], expert system, rule generation keystroke monitoring and so on [4], [7]. Among these methods, neural networks (NN) were widely considered as an efficient approach to adaptively classify patterns. NN was used to detect anomalous user activities [5]. But long training circles have sometimes hindered the application of NN. Therefore, in order to upgrade the convergence, decrease the computational expenses, recognize the anomalous behavior quickly, we firstly divided the sample space into many subspaces in which samples are similar. Secondly, we use NN technique to train the subspace, because the similarity and the fewer samples in the same subspace, the speed of convergence is rapid and the computational expense is lower. The outline of our procedure is as follows, described as Figure 1, where

$$N = N_1 \cup N_2 \cup \cdots \cup N_n \,. \tag{1}$$

Step 1: pretreatment the samples using *k*-means clustering method;
Step 2: for each subspace, the samples are trained by using NN;
Step 3: integrating the results of each subspace.

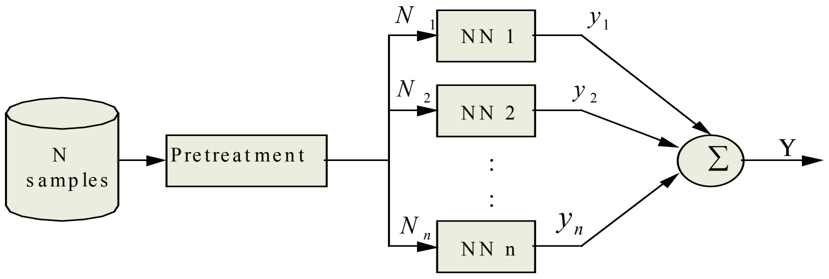


Fig. 1. Procedure of intrusion detection

2 Pretreatment the Samples

In this paper, program behaviors were studied, because intrusions most often occur when programs are misused. Monitoring the program execution and capturing the system calls associated with the program can generate a program profile. Compare the user behavior profiles, program profiles can provide stable tracks for intrusion detection. In detection, the biggest challenge is to choose features that best characterize the user or system usage patterns. We used the frequencies of system calls to characterize program behaviors, counted the occurrence times of the individual system calls during program execution, so we can obtain the program vectors (χ_1, χ_2, $\cdots$, χ_n), where χ_i is defined as the number of occurrence of the system call. Therefore, we can describe the training data set using a system call_by_program matrix in which each entry represents the occurrence of a system call in a program.

Pretreatment means that dividing the large number of vectors into many clusters in which program vectors hold similar system calls. We applied *k*-means clustering technique to pretreat the samples; *k*-means algorithm is the one of the most popular and widely studied clustering methods. Given a set of *n* data points in real *d*-dimensional space $\Re^d$, and an integer *k*, the problem is to assign n observations to k groups so that members of the same group are more "similar" than members of different groups. The following cost function is minimized based on *k*-means algorithm:

$$\mathrm{COST} = \sum_{j=1}^{k} \sum_{i=1}^{nj} \left\| \chi_i - m_j \right\|^2 \,. \tag{2}$$

Where n_j is defined as the number of samples belonging to *jth* cluster, *k* is the number of clusters. The cluster centers are iteratively calculated by means of *k*-means algorithm as follows:

```
Initialize $k$ centers $(m_1, m_2, \cdots, m_k)$ using random sampling where
        $m_j = \chi_i, i \in \{1 \cdots n\}, j \in \{1 \cdots k\}$
Repeat
  For each input vector $\chi_i, i \in \{1 \cdots n\}$
      Assign $\chi_i$ to its nearest cluster as represented by $m_{j*}$ if
      $\|\chi_i - m_{j*}\| < \|\chi_i - m_j\| \quad j*, j \in \{1 \cdots k\}, j* \neq j$
  For each $m_j$, where $j \in \{1 \cdots k\}$
      Update $m_j$ to be the new center of all samples currently in jth
      cluster by
          $m_j = \frac{1}{nj} \sum_{i=1}^{nj} x_i$
  Compute the cost function
          $COST = \sum_{j=1}^{k} \sum_{i=1}^{nj} \|\chi_i - m_j\|^2$
Until the cost function does not change significantly or cluster centers
do not change.
```

After pretreatment by using clustering, large samples were divided into many clusters. Different clusters indicate different data characters, as for a cluster; it may be normal or anomalous data by using same intrusive method. So we can analyze clusters (subspace) separately.

3 Training Subspace Based on Neural Networks

Neural networks have been used extensively in novelty detection. They have the advantage that a very small number of parameters need to be optimized for training networks and no a priori assumptions on the properties of data are made. A number of different architectures and methods have been used for novelty detection [3], these include multi-layer perceptrons, self-organizing maps, radial basis function networks, support vector machines [6], Hopfield networks etc.

The most successful learning algorithm used to train multilayer neural networks is currently the BP algorithm. It is basically a gradient decent algorithm designed to minimize the error function in the weights space. It has been proved that a 3-layer neural network can fulfill the map from *n* dimension to *m* dimension. Therefore, a three-layer network was employed in our study.

In order to construct a network, the number of input and output should be determined besides hidden layer. Because there are distinct system calls observed from different clusters, input nodes are the number of distinct system calls respectively in neural networks. Because the optimal number of hidden nodes was not known before training, neural networks were trained with 5, 10, 15, 20, 25, 30, 40, 50 hidden nodes, the network that classified data most accurately was kept. We used a single output node to represent that input is normal or anomalous and activation function is threshold function.

4 Experiment

For test data set, we defined the binary vector $\overline{V} = \{V_k, \ i = 1,2,\cdots K\}$, where V_k takes value either 0 or 1. When input vector χ_i belong to cluster *k*, $V_k = 1$. Therefore, we integrated results as follows:

$$Y = \sum_{i=1}^{K} V_i Y_i \ . \tag{3}$$

The performance of the improved algorithm is checked to DARPA data for intrusion detection. The DARPA data was labeled with session numbers. Individual sessions can be programmatically extracted from the BSM audit data. Each session consists of one or more processes. A sample system call list (table 1) and the frequencies vector of system calls (vector V) involved in session 2788 are shown as follows.

$$V = (1,72,5,1,22,1,72,90,48,1,3,1,1,4,5,1,1,31,1,1,48,1,1,2,3,3) . \tag{4}$$

Table 1. A sample system call list

chdir	close	execve	fcntl
ioctl	kill	close	mmap
open	setaudit	access	audit
memcntl	exit	fork	login
lstat	munmap	old setgid	old setuid
open	putmsg	setgroups	setpgrp
stat	sysinfo		

In our data set of samples, there are total 42 distinct system calls which means each process is transformed into a vector of size 42, during pretreatment, we divided the samples vectors into two clusters, and learned two clusters using two neural networks separately; because fewer samples in a cluster, the computation and the speed convergence of neural networks have good performance, the trained neural network can offers a desirable attack detection rate (87.6%) with false positive rate (15.2%).

5 Conclusions

In this paper, we presented the application of a simple neural network and *k*-means clustering in order to detect intrusions against systems. Our preliminary experiments with the 1998 DARPA BSM audit data have shown that this approach is able to effectively detect intrusive program behavior and improve the speed of convergence of neural networks. But there are some problems that should be considered, for example the number of clusters predefined may not be optimal and the initial points may lead to non-robustness, further research is in progress to improve the robustness of *k*-means and upgrade the efficiency by parallel detecting attacks.

Acknowledgement

The work described in this paper was supported by grants from the National Natural Science Foundation of China (No. 60271019), the Doctorate Foundation Grants from the Ministry of Education of China (No. 20020611007) and the Natural Science Foundation of Chongqing (No. 8509).

References

1. Sundaram, A.: An Introduction to Intrusion Detection. Special issue on computer security, 2 (1996) 3-7
2. Caberera, J.B.D, Ravichandran, B.: Statistical Traffic Modeling for Network Intrusion Detection. Modeling, Analysis and Simulation of Computer and Telecommunication Systems 2000 Proceedings, 8th International Symposium on, 29 (2000) 466-473
3. Markou, M., Singh, S.: Novelty Detection: A Review—Part 2: Neural Network Based Approaches. Signal Processing, **83** (2003) 2499-2521
4. Axelsson, S.: Research in Intrusion Detection Systems: A Survey Technical Report. Dept. of Computer Engineering, Chalmers University of Technology, Sweden (1998) revised Aug 19, (1999) 98-17
5. Ghosh, A.K.: Detecting Anomalous and Unknown Intrusions Against Programs. Proceedings of IEEE 14th Annual Computer Security Applications Conference, (1998) 259-267
6. Sung, A.H., Mukkamala, S.: Identifying Important Features for Intrusion Detection Using Support Vector Machines and Neural Networks. Applications and the Internet Proceedings. 2003 Symposium on, 27-31 (2003) 209 - 216
7. Liu, Y., Chen, K., Liao, X.F.: A Genetic Clustering Method for Intrusion Detection. Pattern Recognition, **37** (2004) 927-942
8. Schepers, F.: Internet Security Systems (ISS): Network-versus host-based intrusion detection. Information Security Technical Report, **3** (1998) 32-42
9. Bai, Y., Kobayashi, H.: Intrusion Detection Systems: Technology and Development. Advanced Information Networking and Applications, 17th International Conference on, 27-29 (2003) 710 - 715
10. Bykova, M., Ostermann, S., Tjaden, B.: Detecting Network Intrusions via Statistical Analysis of Network Packet Characteristics. IEEE Southeastern Symposium on System Theory (SSST'01) (2001) 309-314

Feature Selection and Intrusion Detection Using Hybrid Flexible Neural Tree

Yuehui Chen[1], Ajith Abraham[2], and Ju Yang[1]

[1] School of Information Science and Engineering
Jinan University, Jinan 250022, China
yhchen@ujn.edu.cn

[2] School of Computer Science and Engineering
Chung-Ang University, Seoul, Republic of Korea
ajith.abraham@ieee.org

Abstract. Current Intrusion Detection Systems (IDS) examine all data features to detect intrusion or misuse patterns. Some of the features may be redundant or contribute little (if anything) to the detection process. The purpose of this study is to identify important input features in building an IDS that is computationally efficient and effective. This paper proposes an IDS model based on general and enhanced Flexible Neural Tree (FNT). Based on the pre-defined instruction/operator sets, a flexible neural tree model can be created and evolved. This framework allows input variables selection, over-layer connections and different activation functions for the various nodes involved. The FNT structure is developed using an evolutionary algorithm and the parameters are optimized by particle swarm optimization algorithm. Empirical results indicate that the proposed method is efficient.

1 Introduction

Intrusion detection is classified into two types: misuse intrusion detection and anomaly intrusion detection. Misuse intrusion detection uses well-defined patterns of the attack that exploit weaknesses in system and application software to identify the intrusions. Anomaly intrusion detection identifies deviations from the normal usage behavior patterns to identify the intrusion.

Various intelligent paradigms namely Neural Networks [1], Support Vector Machine [2], Neuro-Fuzzy systems [3], Linear genetic programming [4] and Decision Trees [7] have been used for intrusion detection. Various data mining techniques have been applied to intrusion detection because it has the advantage of discovering useful knowledge that describes a user's or program's behavior from large audit data sets. This papers proposes a Flexible Neural Tree (FNT) [5] for selecting the input variables and detection of network intrusions. Based on the pre-defined instruction/operator sets, a flexible neural tree model can be created and evolved. FNT allows input variables selection, over-layer connections and different activation functions for different nodes. In our previous work, the hierarchical structure was evolved using Probabilistic Incremental Program Evolution algorithm (PIPE) with specific instructions. In this research, the hierarchical structure is evolved using tree-structure based evolutionary algorithm. The

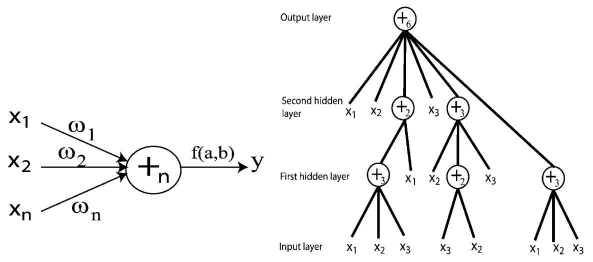


Fig. 1. A flexible neuron operator (left), and a typical representation of the FNT with function instruction set $F = \{+_2, +_3, +_4, +_5, +_6\}$, and terminal instruction set $T = \{x_1, x_2, x_3\}$ (right)

fine tuning of the parameters encoded in the structure is accomplished using particle swarm optimization (PSO). The proposed method interleaves both optimizations. Starting with random structures and corresponding parameters, it first tries to improve the structure and then as soon as an improved structure is found, it fine tunes its parameters. It then goes back to improving the structure again and, fine tunes the structure and rules' parameters. This loop continues until a satisfactory solution is found or a time limit is reached. The novelty of this paper is in the usage of flexible neural tree model for selecting the important features and for detecting intrusions.

2 The Flexible Neural Tree Model

The function set F and terminal instruction set T used for generating a FNT model are described as $S = F \bigcup T = \{+_2, +_3, \ldots, +_N\} \bigcup \{x_1, \ldots, x_n\}$, where $+_i(i = 2, 3, \ldots, N)$ denote non-leaf nodes' instructions and taking i arguments. x_1,x_2,. . .,x_n are leaf nodes' instructions and taking no other arguments. The output of a non-leaf node is calculated as a flexible neuron model (see Fig.1). From this point of view, the instruction $+_i$ is also called a flexible neuron operator with i inputs.

In the creation process of neural tree, if a nonterminal instruction, i.e., $+_i(i = 2, 3, 4, \ldots, N)$ is selected, i real values are randomly generated and used for representing the connection strength between the node $+_i$ and its children. In addition, two adjustable parameters a_i and b_i are randomly created as flexible activation function parameters. For developing the IDS, the flexible activation function $f(a_i, b_i, x) = e^{-(\frac{x-a_i}{b_i})^2}$ is used. The total excitation of $+_n$ is $net_n = \sum_{j=1}^{n} w_j * x_j$, where $x_j(j = 1, 2, \ldots, n)$ are the inputs to node $+_n$. The output of the node $+_n$ is then calculated by $out_n = f(a_n, b_n, net_n) = e^{-(\frac{net_n - a_n}{b_n})^2}$. The overall output of flexible neural tree can be computed from left to right by depth-first method, recursively.

Tree Structure Optimization. Finding an optimal or near-optimal neural tree is formulated as a product of evolution. In this study, the crossover and selection operators used are same as those of standard GP. A number of neural tree mutation operators are developed as follows:

(1) Changing one terminal node: randomly select one terminal node in the neural tree and replace it with another terminal node;

(2) Changing all the terminal nodes: select each and every terminal node in the neural tree and replace it with another terminal node;
(3) Growing: select a random leaf in hidden layer of the neural tree and replace it with a newly generated subtree.
(4) Pruning: randomly select a function node in the neural tree and replace it with a terminal node.

Parameter Optimization with PSO. The Particle Swarm Optimization (PSO) conducts searches using a population of particles which correspond to individuals in evolutionary algorithm (EA). A population of particles is randomly generated initially. Each particle represents a potential solution and has a position represented by a position vector $\mathbf{x_i}$. A swarm of particles moves through the problem space, with the moving velocity of each particle represented by a velocity vector $\mathbf{v_i}$. At each time step, a function f_i representing a quality measure is calculated by using $\mathbf{x_i}$ as input. Each particle keeps track of its own best position, which is associated with the best fitness it has achieved so far in a vector $\mathbf{p_i}$. Furthermore, the best position among all the particles obtained so far in the population is kept track of as $\mathbf{p_g}$. In addition to this global version, another version of PSO keeps track of the best position among all the topological neighbors of a particle. At each time step t, by using the individual best position, $\mathbf{p_i}$, and the global best position, $\mathbf{p_g(t)}$, a new velocity for particle i is updated by

$$\mathbf{v_i(t+1)} = \mathbf{v_i(t)} + c_1\phi_1(\mathbf{p_i(t)} - \mathbf{x_i(t)}) + c_2\phi_2(\mathbf{p_g(t)} - \mathbf{x_i(t)}) \tag{1}$$

where c_1 and c_2 are positive constant and ϕ_1 and ϕ_2 are uniformly distributed random number in [0,1]. The term $\mathbf{v_i}$ is limited to the range of $\pm\mathbf{v_{max}}$. If the velocity violates this limit, it is set to its proper limit. Changing velocity this way enables the particle i to search around its individual best position, $\mathbf{p_i}$, and global best position, $\mathbf{p_g}$. Based on the updated velocities, each particle changes its position according to the following equation:

$$\mathbf{x_i(t+1)} = \mathbf{x_i(t)} + \mathbf{v_i(t+1)}. \tag{2}$$

Procedure of the General Learning Algorithm. The general learning procedure for constructing the FNT model can be described as follows.

1) Create an initial population randomly (FNT trees and its corresponding parameters);
2) Structure optimization is achieved by the neural tree variation operators as described in subsection 2.
3) If a better structure is found, then go to step 4), otherwise go to step 2);
4) Parameter optimization is achieved by the PSO algorithm as described in subsection 2. In this stage, the architecture of FNT model is fixed, and it is the best tree developed during the end of run of the structure search. The parameters (weights and flexible activation function parameters) encoded in the best tree formulate a particle.
5) If the maximum number of local search is reached, or no better parameter vector is found for a significantly long time then go to step 6); otherwise go to step 4);
6) If satisfactory solution is found, then the algorithm is stopped; otherwise go to step 2).

3 Feature Selection and Classification Using FNT Paradigms

The Data Set. The data for our experiments was prepared by the 1998 DARPA intrusion detection evaluation program by MIT Lincoln Lab and contains 24 attack types that could be classified into four main categories namely *Denial of Service (DOS)*, *Remote to User (R2L)*, *User to Root (U2R)* and *Probing*. The data for our experiments contains randomly generated 11982 records having 41 features [6]. The training and test data comprises of 5092 and 6890 records respectively. All the IDS models were trained and tested with the same set of data. Since the data set has five different attack types we performed a 5-class binary classification. The *normal* data belongs to class 1, *Probe* belongs to class 2, *DOS* belongs to class 3, *U2R* belongs to class 4 and *R2L* belongs to class 5.

Feature/Input Selection with FNT. It is often a difficult task to select important variables for any problem, especially when the feature space is large. A fully connected NN classifier usually cannot do this. In the perspective of FNT framework, the nature of model construction procedure allows the FNT to identify important input features in building an IDS that is computationally efficient and effective. The mechanisms of input selection in the FNT constructing procedure are as follows. (1) Initially the input variables are selected to formulate the FNT model with same probabilities; (2) The variables which have more contribution to the objective function will be enhanced and have high opportunity to survive in the next generation by a evolutionary procedure; (3) The evolutionary operators i.e., crossover and mutation, provide a input selection method by which the FNT should select appropriate variables automatically.

Modelling IDS Using FNT with 41 Input-Variables. For this simulation, the original 41 input variables are used for constructing a FNT model. A FNT classifier was constructed using the training data and then the classifier was used on the test data set to classify the data as an attack or normal data. The instruction sets used to create an optimal FNT classifier is $S = F \bigcup T = \{+_5, \ldots, +_{20}\} \bigcup \{x_1, x_2, \ldots, x_{41}\}$. Where $x_i (i = 1, 2, \ldots, 41)$ denotes the 41 features.

The optimal FNTs for classes 1-5 are shown in Figures 2. It should be noted that the important features for constructing the FNT model were formulated in accordance with the procedure mentioned in the previous section. These important variables are shown

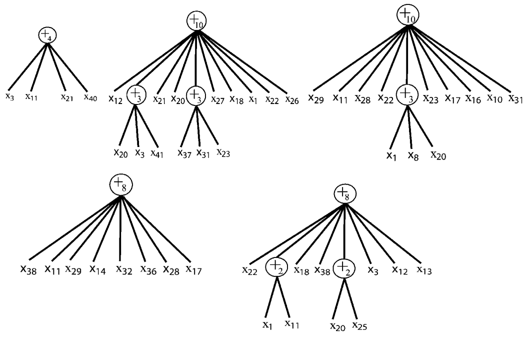

+4
x3 x11 x21 x40
+10
x12 +3 x21 x20 +3 x27 x18 x1 x22 x26
x20 x3 x41
x37 x31 x23
+10
x29 x11 x28 x22 +3 x23 x17 x16 x10 x31
x1 x8 x20
+8
x38 x11 x29 x14 x32 x36 x28 x17
+8
x22 +2 x18 x38 +2 x3 x12 x13
x1 x11
x20 x25

Fig. 2. The evolved FNT for classes 1, 2, 3, 4 and 5 with 41 input variables

Table 1. The important features selected by the FNT algorithm

Class	Important variables
Class 1	$x_3, x_{11}, x_{21}, x_{40}$
Class 2	$x_1, x_3, x_{12}, x_{18}, x_{20}, x_{21}, x_{22}\ x_{23}, x_{26}, x_{27}, x_{31}, x_{37}, x_{41}$
Class 3	$x_1, x_8, x_{10}, x_{11}, x_{16}, x_{17}, x_{20}, x_{12}, x_{23}, x_{28}, x_{29}, x_{31}$
Class 4	$x_{11}, x_{14}, x_{17}, x_{28}, x_{29}, x_{32}, x_{36}, x_{38}$
Class 5	$x_1, x_3, x_{11}, x_{12}, x_{13}, x_{18}, x_{20}, x_{22}, x_{25}, x_{38}$

Table 2. Detection performance using FNT and NN classification models

Attack Class	FNT	NN
Normal	**99.19%**	95.69%
Probe	**98.39%**	95.53%
DOS	**98.75%**	90.41%
U2R	99.70%	**100%**
R2L	**99.09%**	98.10%

Table 3. The false positive/negative errors by the FNT algorithm

Attack Class	False positive error	False negative error
Normal	0.0581%	0.7837%
Probe	1.3943%	0.2160%
DOS	0.6241%	0.6241%
U2R	0.2177%	0.0726%
R2L	0.7547%	0.1597%

in Table 1. Table 2 depicts the detection performance of the FNT by using the original 41 variable data set.

For comparison purpose, a neural network classifier trained by PSO algorithm were constructed using the same training data sets and then the neural network classifier was used on the test data set to detect the different types of attacks. All the input variables were used for the experiments and the results are shown in Table 2.

4 Conclusions

In this paper we presented a Flexible Neural Tree (FNT) model for Intrusion Detection Systems (IDS) with a focus on improving the intrusion detection performance by reducing the input features. We have also demonstrated the performance using different reduced data sets. As evident from Tables 1 and 2, the proposed flexible neural tree approach seems to be very promising. The FNT model was able to reduce the number of variables to 4, 12, 12, 8 and 10 (using 41 input variables) for classes 1-5 respectively.

Using 41 variables, FNT model gave the best accuracy for the detection of most of the classes (except U2R). The achieved false positive/negative errors using 41 data set by the FNT algorithm is depicted in Table 3. The direct NN classifier outperformed the FNT approach for U2R attack only.

References

1. Debar, M., Becke, D., Siboni, A.: A Neural Network Component for an Intrusion Detection System. Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy (1992)
2. Mukkamala, S., Sung, A.H., Abraham, A.: Intrusion Detection Using Ensemble of Soft Computing Paradigms. Advances in Soft Computing, Springer Verlag, Germany (2003) 239-248
3. Shah, K., Dave, N., Chavan, S., Mukherjee, S., Abraham, A., Sanyal, S.: Adaptive Neuro-Fuzzy Intrusion Detection System. IEEE International Conference on ITCC'04, **1** (2004) 70-74
4. Abraham, A.: Evolutionary Computation in Intelligent Web Management. Evolutionary Computing in Data Mining, A. Ghosh and L. Jain (Eds.), Studies in Fuzziness and Soft Computing, Springer Verlag Germany, Chapter **8** (2004) 189-210
5. Chen, Y., Yang, B., Dong, J., Abraham, A.: Time-series Forecasting using Flexible Neural Tree Model. Information Science, (2005) In press
6. KDD cup 99, http://kdd.ics.uci.edu/database/kddcup99/kddcup.data_10_percent.gz
7. Chebrolu, S., Abraham, A., Thomas, J.P.: Feature Detection and Ensemble Design of Intrusion Detection Systems. Computers and security, (http://dx.doi.org/10.1016/j.cose.2004.09.008), In press.
8. Barbara, D., Couto, J., Jajodia, S., Wu, N.: ADAM: A Testbed for Exploring the Use of Data Mining in Intrusion Detection. SIGMOD Record, **30** (2001) 15-24
9. Joo, D., Hong, T., Han, T.: The Neural Network Models for IDS Based on the Asymmetric Costs of False Negative Errors and False Positive Errors. Expert Systems with Applications, **25** (2003) 69-75

Detection of Epileptic Spikes with Empirical Mode Decomposition and Nonlinear Energy Operator

Suyuan Cui, Xiaoli Li, Gaoxiang Ouyang, and Xinping Guan

Institute of Electrical Engineering, Yanshan University, Qinhuangdao, 066004, China
Xlli@ysu.edu.cn

Abstract. Epileptic seizure is a serious brain disease. The characteristic signature of epileptic seizure is interictal spikes and sharp waves. Development of a reliable method to detect spikes from EEG data is of major clinical and theoretical importance. In this paper, a new detection algorithm that combines the Empirical Mode Decomposition (EMD), Hilbert Transformation (HT) and Smoothed Nonlinear Energy Operator (SNEO) is proposed to detect spikes hidden in human EEG data. Finally, the EEG data generated by a nonlinear lumped-parameter cerebral cortex model and real EEG data from human are applied to test the performance of the new detection method. The results show that this method can efficiently detect the spikes hidden in EEG signals.

1 Introduction

About 0.5-1% of the population suffers from epilepsy, which is the most common neurological disease. Epilepsy is the result of abnormal synchronous discharges from large ensembles of neurons in brain structures. Spike as its feature is important for the disease diagnosis. Thus, the spike detection is a very issue in the epilepsy diagnosis.

In this paper, a new algorithm is proposed to deal with the EEG data, in particular the spike detection, which is a combination of Empirical Mode Decomposition (EMD), Hilbert Transformation (HT) [1], Smoothed Nonlinear Energy Operator (SNEO) [2]. In addition, in order to test the performance of this method, a model is developed to generate an EEG signal with spikes. The models to generate epileptic EEG signals often can be divided into micro and macro two types [3]. Here a lumped-parameter cerebral cortex model is addressed to generate spontaneous EEG signals.

The structure of the rest of this paper is arranged as follows. Section 2 introduces a lumped-parameter cerebral cortex model. The new detection algorithm of spike in EEG data is addressed in Section 3. Section 4 is the results of spike detection for the simulated and real EEG signals. The conclusion is given in Section 5.

2 Nonlinear Lumped-Parameter Model

The nonlinear lumped-parameter model was initially proposed by Lopes Da Silva et al (1974) [4], and later was improved and extended by Wending et al [3,5]. In present study, we use Wending's model, which is composed of the model of one neural population and the model of multiple coupled neural population. The model is shown in Fig. 1.

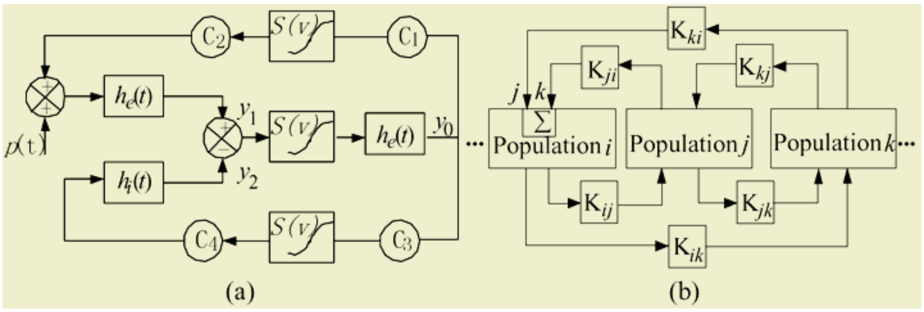


Fig. 1. The model. (a) A population of nonlinear lumped-parameter model. (b) The multiple coupled populations model

The population model (Fig. 1a) contains two interacting subsets: the first is pyramidal cells, which project to and receive feedback (either excitatory or inhibitory) from the second subset; the second is composed of excitatory neurons and inhibitory neurons, which receives excitatory input only [3].

The input $p(t)$ represents the average density of afferent action potentials. Subset 1 is characterized by two second-order dynamic linear transfer functions, which transfers the average pre-synaptic pulse density of afferent action potentials (the input) into an average postsynaptic membrane potential (the output). The impulse responses of excitatory and inhibitory neurons are presented by $h_e(t)$ and $h_i(t)$ respectively. When the neuron is fired, a static nonlinear asymmetric sigmoid function $S(v)$ would send the average postsynaptic potential to the average pulse density of potentials. There is only one linear transfer function from excitatory neuron in subset 2. Parameters C1-C4 are constants, which represent average synaptic number. In the normal situation, excitatory and inhibitory neurons keep a balance. The change of the ratio between excitatory and inhibitory synaptic gains will trigger epileptic seizure and generate sporadic spikes, rhythmic spikes and so on.

Fig.1 (b) is the multiple coupled neural network model [3], which is composed of a few neural networks. The population corresponds different brain areas and their interactions can be linked by the parameters K^{ij}. During the epileptic seizures, the excitatory potentials propagate along axons from a population to another, leading to high voltage exciting states.

3 Detection Method

In this paper, we propose a new algorithm to analyze the simulated and real EEG data. The method is composed of four steps, as shown in Fig.2. (i) The EEG data is decomposed into components called intrinsic mode functions (IMFs) with Empirical Mode Decomposition (EMD). (ii) IMFs with high signal-to-noise ratio (SNR) are sorted out for the detection of spikes. Then, an automatic de-noising process with wavelet is performed for the selected IMFs. (iii) The instantaneous energy of each IMFs are gained by using Hilbert transform (HT), and summed as one signal for the detection of spikes. (iv) Smoothed Nonlinear Energy Operator (SNEO) [6] is used to detect spikes.

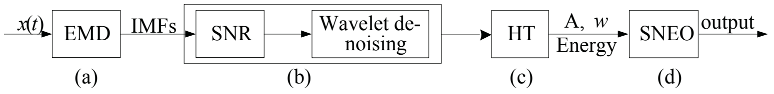


Fig. 2. The block diagram of the detection method

The EMD is first proposed by Huang et al. in 1998 [1]. This novel technique is aimed at decomposing a nonstationary and nonlinear time series into zero-mean AM-FM components. The major advantage of the EMD is that the basis functions derived from the given signal $x(t)$ are not fixed. The principle of decomposition is below. (i) Each IMF has the same numbers of zero crossings and extremes; and (2) IMF is symmetric with respect to the local mean. The effective algorithm of EMD can be found in [6]. For a given EEG data $x(t)$, the simplification of EMD is below:
(1) Initialization: $r_i(t) = x(t), i = 1$; (2) Extraction of the i^{th} IMF: (a) Initialization: $h_0(t) = r_j(t), j = 1$; (b) Calculation of the local minima and maxima of $h_{j-1}(t)$; (c) Interpolation of the local maxima and the local minima by a cubic spline; extraction of the upper and lower envelops of $h_{j-1}(t)$; and calculation of the mean, $m_{j-1}(t)$, of the upper and lower envelopes; (d) $h_j(t) = h_{j-1}(t) - m_{j-1}(t)$; (e) If the criteria proposed in [6] are satisfied, then set $imf_i(t) = h_j(t)$, else back to (b) step with $j = j+1$; (3) $r_i(t) = r_{i-1}(t) - imf_i(t)$. If $r_i(t)$ still has at least 2 extreme then back to 2 with $i = i+1$, else the decomposition stop and $r_n(t)$ is called the residual of the EEG data that also represent the trend of the EEG data.

In order to sort out IMFs with spikes, we propose to calculate the signal-to-noise ratio (SNR) of each IMFs. The IMFs with high SNR are sorted out for further detection of spikes. At the same time, the first IMF (contain high frequency noise) and the residue are also removed. Then, a de-noise method [7] based symlet wavelet is applied to the selected IMFs for removing the noise hidden in IMFs. The next step is to calculate the instantaneous energy of each IMF selected. Herein, Hilbert transform [1] reaches above aim. The equation of HT is below:

$$X(t) = x(t) + jH\{x(t)\} = A(t)e^{j\theta(t)} \tag{1}$$

where $x(t)$ represents IMF, $H\{x(t)\}$ is the Hilbert transform of $x(t)$. $A(t)$ is the instantaneous amplitude, and $\theta(t)$ is the instantaneous phase. Then the instantaneous frequency is $w(t) = d\theta(t)/dt$. Hence, the instantaneous energy of the IMF is $A^2(t)$. Combination of the instantaneous $A^2(t)$ of all of IMFs selected, a new series $y(t)$ is constructed for the detection of spikes.

A smoothed nonlinear energy operator (SNEO) proposed in [2,8] gives the estimate of local energy content of a stochastic signal. The method is applied to detect epileptic spikes in EEG [2]. We convolve energy operator $\Psi[y(t)]$ with a time-domain window, $w(k)$:

$$\Psi_s[y(t)] = \Psi[y(t)] \otimes w = \sum_{k=b}^{k=-b} w(k)\Psi[y(t+k)] \tag{2}$$

where $\otimes$ is the convolution. Herein, A Barrett window function with an integer filter is adopted.

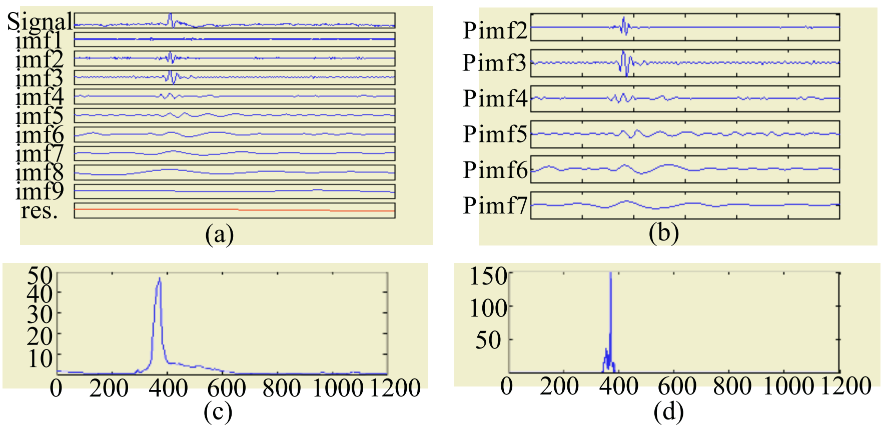
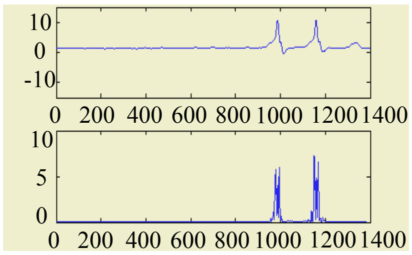


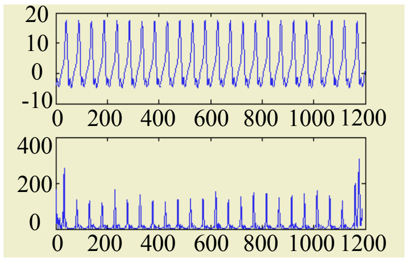
Fig. 3. An example of the detection method. (a) Original signal (top, $x(t)$), the IMFs (imf 1-9) and a residual from EMD method. (b) The selected IMFs with SNR. (c) Combination of the instantaneous energy of the selected IMFs in (b), $y(t)$. (d) The output of SNEO of (c)

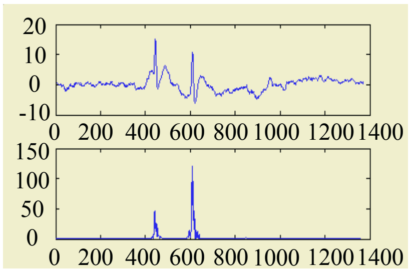
4 Results

Fig.3 provides a detail procedure of the new method of spike detection. As can be seen in Fig. 3 (a), an original EEG signal ($x(t)$) contains one spike. And it is decomposed into 9 IMFs and a residual by EMD. The IMFs (IMF2, 3, 4, 5, 6, 7) that contain the information of the spike are selected, and pre-processed by using wavelet-based de-noise, as shown in Fig.3 (b)). Then, the instantaneous energy of each IMF can be calculated by using HT. One can notice the combination of the instantaneous energy of the selected IMFs, $y(t)$, is significant in the location of the spike (see Fig. 3 (c)). After the energy operator SNEO for $y(t)$, the spike can be easily detected by setting up a threshold.

To test the performance of the new detection method, a lot of simulated and real EEG signals are used. The simulated EEG data are generated from the multiple coupled population model. The increase of the ratio A/B (A and B represent the excitatory and inhibitory synaptic gains, respectively) results in the spikes in the EEG signals. Other parameters can be found in [3]. In this paper, the parameter A is progressively increased, from 3.6 to 3.8. The generated EEG signals are shown in Fig.4 (a) (A=3.6) and (b) (A=3.8). They are sporadic or rhythmic spikes, respectively. The real EEG signals can be obtained from Internet [9], which include sharp waves, sharp-slow waves and a sequence of spikes waves. The real EEG signals are recorded from juvenile epileptic patients according to the international 10-20 standard system with the reference average electrode. In this paper, two typical EEG data are selected, as shown in Fig.4 (c) and (d). It is obvious for either simulated or real signals the new method can indicate the spikes efficiently and accurately.


(a) Simulated EEG signals: Sporadic spikes


(b) Simulated EEG signals: Rhythmic spikes


(c) Real EEG signals: Sporadic spikes

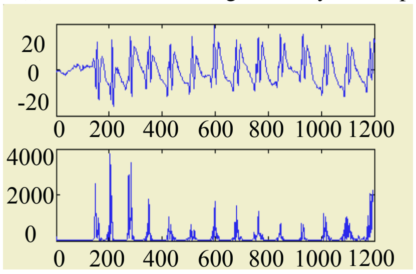

(d) Real EEG signals: Rhythmic spikes

Fig. 4. Detection of the epileptic spikes in simulated and real EEG signals. (a) Simulated EEG signals with sporadic spikes. (top) EEG signal; (bottom) Indication of sporadic spike. (b) Simulated EEG signals with rhythmic spikes. (top) EEG signal; (bottom) Indication of rhythmic spikes. (c) Real EEG signals with sporadic spikes. (top) EEG signal; (bottom) Indication of sporadic spike. (d) Real EEG signals with rhythmic spikes. (top) EEG signal; (bottom) Indication of rhythmic spikes

Table 1. Comparison of performace of three spike detection algorithms in real EEG data

Algorithm	New algorithm	EP	MP
Sensitivity	96.25%	93%	89%
Specificity	95%	89%	86%

The sensitivity and specificity of the new detection method are analyzed by a mount of real EEG signals with various spikes. Real EEG recordings contain 160 epileptic spikes. The sensitivity and specificity of the new method is around 96.25% and 95%, respectively. Other methods: end-predication (EP) and mid-prediction filter (MP) [10] are applied to detect the spikes in the same EEG recordings, the results are shown in Table 1. The new method is the best.

5 Conclusion

Precise spike detection is extremely important for diagnosis of epilepsy [11]. In this paper, a new algorithm of spike detection is proposed to detect the spikes hidden in EEG data. The testing results of the simulated and real EEG data show this new method is the best by comparison with end-predication (EP) and mid-prediction filter (MP). In the near future, the method will be developed to predict epileptic seizure such as in [12].

Acknowledgments

The support of National Science Foundation of China (60274023) is gratefully acknowledged.

References

1. Huang, N.E., Shen, Z.: The Empirical Mode Decomposition and Hilbert Spectrum for Nonlinear and Nonstationary Time Series Analysis. Proc. Roy. Soc. London A, **454** (1998) 903-995
2. Mukhopadhyay, S.: A New Interpretation of Nonlinear Energy Operator and Its Efficacy in Spike Detection. IEEE Transactions on Biomedical Engineering. **45** (1998) 180-187
3. Wendling, F., Bellange, J.J.: Relevance of Nonlinear Lumped-parameter Models in the Analysis of Depth-EEG Epileptic Signals. Biol. Cybernetics (2000) 83-367
4. Lopes da S., F.H., Rotterdam, A.: Models of Neuronal Populations: The Basic Mechanisms of Rhythmicity. Prog.Brain Res, **45** (1974) 281-308
5. Wendling, F., Bartolomei, F.: Epileptic Fast Activity Can Be Explained by a Model of Impaired GABAergic Dendritic Inhibition. European Journal of Neuroscience, **15** (2002) 1499-1508
6. Rilling, G.: On Empirical Mode Decomposition and Its Algorithms, In Proc. IEEE-EURASIP Workshop on Nonlinear Signal and Image Processing. NSIP-03 (2003)
7. Donoho, D.L.: De-noising by Soft-thresholding. IEEE. Trans. Inform. Theory. **41** (1995) 613-627
8. Jones, D.L., Parks, T.W.: A Resolution Comparison of Several Time-frequency Representations. IEEE Trans. Signal Processing. **40** (1992) 413-420
9. URL http://www.republika.pl/eegspike/signals.html
10. Ray, G.C.: An Algorithm to Separate Nonstationary Part of a Signal Using Mid-Prediction Filter.IEEE Trans. Signal Processing. **42** (1994) 2276-2279
11. Li, X.L., Guan, X.P.: Using Damping Time for Epileptic Seizures Detection in EEG. Modelling and Control in Biomedical Systems, Elsevier Ltd, (2003) 255-258
12. Li, X.L., Ouyang, G., Yao, X. and Guan, X.P.: Dynamical Characteristics of Pre-epileptic Seizures in Rats with Recurrence Quantification Analysis, Phy. Let. A **333** (2004) 164-171

Neural Networks for Solving On-Line Outlier Detection Problems

Tianqi Yang[*]

Department of Computer Science, Jinan University,
Guangzhou, Guangdong 510632, China
y_tq@21cn.com

Abstract. To implement on-line process monitoring techniques, a neural network approach to the on-line solution of outlier detection is considered. The first technique is based on the determination of the predictor coefficients on the variables. The coefficients are then solved using least squares (LS) optimization criteria. The second technique is a standard penalty method implemented as an analog neural network. A recursive algorithm is developed for estimating the weights of the ANN and parameters of LS model. The validity and performance of the proposed algorithms has been verified by computer simulation experiments. The analog neural networks are deemed to be particularly well suited for high throughput, real time applications.

1 Introduction

LS plays an important role in systems identification, signal restoration and outlier detection it is required to solve such systems in real time[1]. Unfortunately for applications requiring the solution of such systems within a fraction of a millisecond a standard digital computer can not comply with the desired computation time or alternatively its use is too expensive. One promising approach to solving such problems is to employ artificial neural networks. Neural optimization networks offer evident advantages in solving problems that relate to a high level of computational complexity. These features have recently been shown to have considerable potential for outlier detection. The main purpose of this paper is to review known techniques involving the use of analog neural networks and to propose algorithm which requires fewer processing time and offer improved computational performance.

2 Formulation of the Problem

This section outlines LS approach to the solution of the following classical optimization problem.

A multiple linear regression model can be presented in the matrix notation as[2]:

$$y = X\beta + \varepsilon . \tag{1}$$

where y is a $m\times 1$ vector of the response (dependent) variable; X is a $m\times n$ matrix of explanatory (independent, factors, regressors, predictors) variables; β is a $n\times 1$ vector of regression coefficients parameters and ε is a $m\times 1$ vector containing the error terms with zero mean and an unknown variance σ^2.

[*] Supported by Natural Science Foundation of Department of Education of Guangdong Province and Jinan University.

The purpose of regression analysis is to estimate vector of unknown parameters β. The least squares estimator optimizes parameters by making the sum of squared residuals differences between the observed y_i and the estimated $\hat{y}_i$ as small as possible:

$$\text{Minimize } \sum_i e_i^2 \text{ where: } e_i = y_i - \hat{y}_i$$

The least squares estimator is the optimal one for data with normal error distribution. Each outlier detection measure is designed to detect a specific phenomenon in the data. In a particular application, the analyst does not have to consider all these detection, since there is a great deal of redundancy in them. They are closely related, as they are functions of the same basic building blocks in model construction, i.e., they are characterized by certain function of the LS residuals, ***e*** the diagonal elements of the hat matrix, and/or the elements of the catcher matrix:

A performance measure is defined as

$$J(\beta) = \|e(n,\beta)\|_2^2 . \tag{2}$$

where $e = y - X\beta$; then this is minimized with respect to β, that is,

$$\min_{\beta} J(\beta) = \min_{\beta} \|e(n,\beta)\|_2^2 . \tag{3}$$

where the total number of samples used from the data set $\{(X,y)\}$ denoted by N, is taken as

$$N > 2n$$

Minimizing $\|e(n,\beta)\|_2^2$ with respect to β gives the result as

$$\hat{\beta} = (XX^T)^{-1} Xy . \tag{4}$$

3 An Analog Neural Network Model

The mapping of a constrained optimization problem into an appropriate loss (cost) function is a standard approach in the design of ANN. Consequently the construction of an appropriate energy function E(x) for which the lowest energy state will correspond to the optimal solution *x'* is a pre-requisite for the formulation of the optimization problem in terms of ANN. For the minimization problem, a general energy function, based on the penalty method can be constructed.

The definition of the analog neural network is as Fig1. Let g_i and f_i represent a neuron output with i =1,2,…,n, where n is the number of neurons in the network. The energy function of the network is[3]

$$E = \sum_{j=1}^{M} F\left(\sum_{i=1}^{N} D_{ji} V_i - b_j\right)^2 + \sum_{i=1}^{N} \frac{1}{R_i} \int_0^{V_i} g^{-1}(\tau) d\tau \tag{5}$$

Let us define:

$$D = X , \quad B = y$$

$$D = \{D_{ij}\} \qquad B = [b_1, b_2, \ldots, b_M]^T ,$$

$$D = \{D_{ij}\} \qquad f(t) = k_1 z , \; g(u) = k_2 u .$$

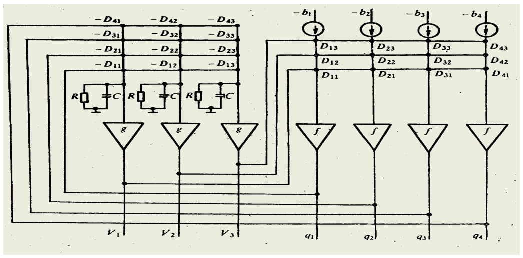


Fig. 1. The definition of the analog neural network

The energy function can be rewritten as

$$E = \frac{1}{2}K_1\left[\|y - XV\|^2 + \frac{1}{2RK_2K_1}\|V\|^2\right]. \tag{6}$$

According to the circuit diagram, we can write the output of neuron as:

$$\begin{cases} C\frac{dU}{dt} = -\frac{U}{R} - X^TQ \\ Q = K_1(XV - y) = K_1(K_2XU - y) \end{cases}. \tag{7}$$

and

$$\frac{dU}{dt} = -\left(\frac{1}{RC}I + K_1K_2X^TX \Big/ C\right)U + K_1X^Ty \ /C \tag{8}$$

When $dE/dt = 0$, then $dU/dt = 0$. Therefore, the output of the neural network is given as

$$V_F = K_2U_F = \left(\frac{1}{RK_1K_2}I + X^TX\right)^{-1}X^Ty \tag{9}$$

The error between $\hat{\beta}$ and V_F is written as

$$\left\|V_F - \hat{\beta}\right\| = \frac{1}{RK_1K_2}\left\|\sum_{i=1}^{r}\frac{1}{\lambda_i + \frac{1}{RK_1K_2}}\frac{1}{\lambda_i}\Lambda_i\Gamma_i^T\right\| \le \frac{1}{1 + RK_1K_2\lambda_{\min}}\|\beta\| \tag{10}$$

$\hat{\beta}$ can be obtained by,

$$\hat{\beta} = V_F = \min_{\alpha\to 0}\left(X^TX + \alpha I\right)^{-1}X^Ty \tag{11}$$

We can adjust resistance R and coefficient K_1, K_2 to Approach $\hat{\beta}$ in high speed.

4 Outliers Measures

d was designed to find out the optimal operating policies to indicate the outliers situations.

$$d = \frac{\sigma_i^2}{\sigma_o^2} + \frac{1}{\sigma_o^2}\left[\beta - \hat{\beta}\right]^T S^{*-1}\left[\beta - \hat{\beta}\right] \tag{12}$$

For character strings, the outliers measures function may be in the form of a pattern string that is used to cover all of the patterns [4]. The outliers increases when the pattern covering all of the strings. The general task of finding an exception set can be

NP-hard (i.e., intractable). A sequential approach is computationally feasible and can be implemented using the algorithm.

The method is an iterative procedure for the determination of a robust covariance matrix. In the first step, the Mahalanobis distance is determined for observation $\hat{\beta}$. S^* is the covariance, calculated using all n observations. Note that the reduced Mahalanobis distance can be calculated by replacing β with the principal component scores and S^* with the covariance of the principal component scores. The n/2 observations with the smallest Mahalanobis distances are determined. Such observations are used to determine the new $\hat{\beta}$ and new covariance S^*. The Mahalanobis distance is recalculated using the new $\hat{\beta}$, the new covariance S^*, and the old β. The iterative procedure continues until $\hat{\beta}$ and S^* stabilize and further iterations can result in higher sensitivity in detecting outliers.

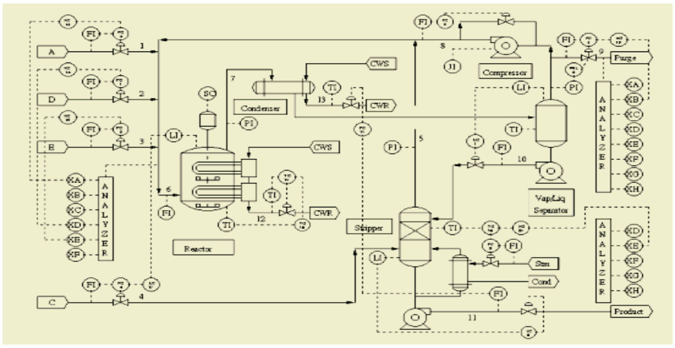


Fig. 2. A process flowsheet

5 Application

Fig. 2 is a process flowsheet based on an industrial process where the components, kinetics, and operating conditions were disguised for proprietary reasons[5]. The gaseous reactants A, C, D, and E and the inert B are fed to the reactor where the liquid products G and H are formed. The reactions in the reactor are irreversible, exothermic, and approximately first-order with respect to the reactant concentrations. The reactor product stream is cooled through a condenser and then fed to a vapor-liquid separator. The vapor exiting the separator is recycled to the reactor feed through a compressor. A portion of the recycle stream is purged to keep the inert and byproducts from accumulating in the process. The condensed components from the separator (Stream 10) are pumped to the stripper. Stream 4 is used to strip the remaining reactants in Stream 10, and is combined with the stream.The products G and H exiting the base of the stripper are sent to a downstream process which are not included in this process. The simulation code allows 20 preprogrammed major process faults, as shown in Table 1. The simulator was used to generate normal data and faulty data. The first simulation run generated 960 observations for the normal operating conditions (Fault 0), with each observation containing 52 process variables. The next 20 simulation runs directly corresponded to Faults 1-20. Each faulty data set contained 480 observations. Twenty case studies were performed, each with the same normal data set (Fault 0) but a different faulty data set. This resulted in 1440 observations for each case study. The outlier detection algorithms were used to identify the most consistent 720 observations and the performance was evaluated in terms of the number of correctly identified normal data in those 720 observations for each case study. All the outlier detection algorithms were programmed using MATLAB6.1. The time trajectories of the LS estimates obtained using the neural network are presented in Fig.3. It can be noticed that LS solutions converge into a 2% envelope in approximately 0.15ms of simulated time.

Table 1 Process outlier for the industrial process

	Outlier Description	Type
1	A/C feed ratio, B composition constant (stream 4).	Step
2	B composition, A/c ratio constant (stream 4).	Step
3	D feed temperature (stream 2).	Step
4	Reactor cooling water inlet temperature.	Step
5	Condenser cooling water inlet temperature.	Step
6	A feed loss (stream 1).	Step
7	C header pressure loss-reduced availability (stream 4).	Step
8	A,B,C feed composition (stream 4).	Random
9	D feed temperature (stream 2).	Random
10	C feed temperature (stream 4).	Random
11	Reactor cooling water inlet temperature.	Random
12	Condenser cooling water inlet temperature.	Random
13	Reaction kinetics.	Slow drift
14	Reactor cooling water valve.	Sticking
15	Condenser cooling water valve.	Sticking
16	Unknown	
17	Unknown	
18	Unknown	
19	Unknown	
20	Unknown	

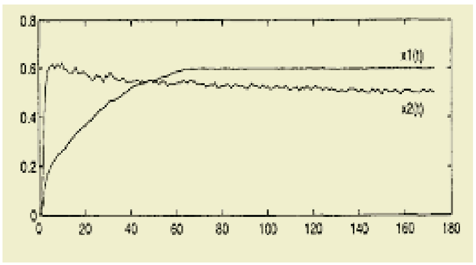


Fig. 3. LS estimates obtained using the neural network

In the presence of outliers, the correlation structure of the normal data was disrupted. As a result, the normal data was not centered at the origin. Because the mixture of the normal data and outliers significantly deviated from a normal distribution and the outliers were extreme, the observations were not evenly distributed inside the confidence region. When outliers are not significantly different from the normal data, the score plot of the first two principal components will be evenly distributed inside the confidence region. An effective outlier detection algorithm should be able to detect most of the outliers.Fig.4 shows The Mahalanobis distances for iterations 1 and 10. Observations 1-960 represent normal data and observations 961-1440 represent Fault 4 data.

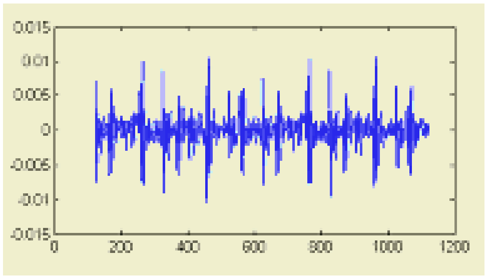


Fig. 4. The Mahalanobis distances for iterations 1 and 10. Observations 1−960 represent normal data and observations 961−1440 represent Fault 4 data. Solid line represents median of the distance

6 Discussions

This paper discussed the analog neural networks methodology for solving outlier detection in real-time. The paper reviewed known methods and presented original techniques for the solution of this problem based on the regularized least squares (LS) approaches. Furthermore, the paper presented a new, simple and efficient adaptive learning algorithm in the form of differential equations. It has been demonstrated that these equations can be implemented using neurons with adaptive weights. The algorithms are deemed to be particularly well suited for real-time and high throughput rate applications. It has been demonstrated that the analog neural networks offer a several orders of magnitude improvements in efficiency over the more conventional digital neural nets.

References

1. Van Huffel, S., Vandewalle, J.: The Total Least Squares Problem: Computational Aspects and Analysis, SIAM, Philadelphia, PA (1991)
2. Li, B., Bart De Moor, B.: Identification of Influential Observations on Total Least Squares Estimates. Linear Algebra and its Applications, **348** (2002) 23–39
3. Foo, D.Y., Anderson, I.R., Takefuji, Y.: Analog Components for the VLSI of Neural Networks. IEEE Circuits Devices, **6** (1990) 18-26
4. Hadi, A.S., Nyquist, H.: Frechet Distance as a Tool for Diagnosing Multivariate Data. Linear Algebra Appl, **289** (1999) 183–201
5. Leo, H., Chiang, R., Pell, J., Seasholtz, B.M.: Exploring Process Data With The Use Of Robust Outlier Detection Algorithms Journal of Process Control, **13** (2003) 437-449
6. Lalor, G.C., Zhang, C.: Multivariate Outlier Detection and Remediation in Geochemical Databases. The Science of the Total Environment, **281** (2001) 99-109
7. Rivals, I., Personnaz, L.: Construction of Confidence Intervals for Neural Networks Based On Least Squares Estimation. Neural Networks, **13** (2000) 463–484

Pedestrian Detection by Multiple Decision-Based Neural Networks

Chen Huang, Guangrong Tang, and Yupin Luo

Dept. of Automation, Tsinghua University,
Beijing 100084, China
may_vv00@mails.tsinghua.edu.cn

Abstract. This paper describes an approach of pedestrian detection for onboard application in night driving. Based on single-frame analysis, two-stage method is designed for detecting pedestrians in the cluttered scenes, which are obtained via a normal camera installed on moving vehicle. In the first stage, bright foreground objects are extracted from dim background as candidates. In the second one, we cascade different-feature-based classifiers, emphasizing shape-based classification. Novel contributions of this paper are, 1) developing the shape representation of candidates; 2) combining multiple Decision-Based Neural Networks for elaborate classification, and further reducing the false alarms. Experiments show that our approach is promising, while the system can achieve real-time detection.

1 Introduction

With the popularization of automobile, intelligent transportation system (ITS) generally focuses more and more attention. As the central function in ITS, pedestrian detection is significant for avoiding traffic accident, especially for night driving.

Comparing with the other targets in the domain of object detection, pedestrian detection is particularly challenging with multifold reasons.

a) Being non-rigid, pedestrian appears widely various body poses, furthermore, the different clothing augments the confusion of appearance.
b) Meanwhile, the environment takes prominent effect, such as the diverse illumination condition for night traffic, and the surrounding clutters in city block, particularly the confusion of human-shaped objects.
c) In addition, the distance and the capability of camera limit the resolution of pedestrian image in the visual analysis.

Pedestrian detection heavily depends on typical features of human body. The features selection and utilization are crucial.

As pedestrian takes on both static and motional features, the detection can be correspondingly based on single-frame analysis or multi-frame analysis.

The static features of pedestrian include position, size, aspect ratio, symmetry, texture, shape, and etc.

Motion is one of the most particular features of pedestrian. Concerning fixed camera, it is effective in multi-frame's associated analysis and tracking, but only to moving people. However, for onboard camera which is moving with vehicle, the relative motion of pedestrian is difficult to be distinguished from the camera's ego-motion. This approach is impracticable and fallible for onboard system.

On the other hand, the available features are affected by selection of camera. Generally, both infrared camera and normal camera can be employed. The former one captures pedestrian as a hotspot, since pedestrian usually emits more heat than the abiotic factors in background, especially at night. Therefore, the use of infrared camera advantages candidate localization. However, the enhancement of hot objects causes sensitiveness to temperature, and the lower resolution of infrared image reduces the available features. Normal camera holds the raw visual scene.

Our system, with a normal camera, carries out a two-stage single-frame visual-based approach for pedestrian detection. In order to get the best use of available features, multiple cascaded classifiers are designed respectively, and the training job is on both entire and partial figures. The combination gives the final judgment.

The outline of this paper is as follow. Section 2 reviews related works on pedestrian detection. Section 3, presents our approach's architecture and implementation. Section 4, shows the experimental results. Section 5 brings the conclusion.

2 The Related Work

Numbers of related work have been done for pedestrian detection by different strategies. Most are concerning daytime detection, as [1]-[4].

The system presented in [1], uses distance transfer-based template (DTBT) matching on shape feature. It scans candidates in the single frame and recognizes subsequently. But exhaustive searching should be time consuming. [2] integrates stereo analysis with Neural Network recognition. Two cameras are installed for the stereo geometry analysis. They train a three-layer feedforward neural network on back-propagation algorithm. In [3], a pedestrian detecting system for daytime detecting is described, which is based on motion analysis and integrates with intensity information. With stationary camera, the system could detect walking humans availably. Focus on shape-based techniques, [4] exploits the morphological features and vertical symmetry of human, to localize and recognize the pedestrian in a specific region.

For night vision, [5] presents a two-stage method, based on infrared-image analysis. Support Vector Machine (SVM) is used in detection stage, and Kalman Filter prediction along with Mean Shift Tracking in tracking stage. However, the employment of infrared camera brings sensitiveness to temperature, which reduces the robustness.

3 Integrated Implementation

With a single normal onboard camera, our system is based on single-frame detection, and extracts static features of pedestrian for detection. Here two-stage approach consists of segmentation and classification stage. Especially, cascaded multiple classifiers are combined in unique hierarchy, and Decision-Based Neural Network algorithm is introduced for main shape-based classification.

3.1 Segmentation

In the first stage, segmentation is according to the brightness of pedestrian against dim background at night. Bright foreground regions are extracted as candidates.

The extraction is achieved by filtering the image separately by two masks of different size, so that the background is smoothed whereas the foreground objects are enhanced respectively.

3.2 Cascaded Multiple Classifiers

Through above segmentation, averagely 45 candidates are picked out per frame, with different size, position, texture, shape and etc. It is quite difficult to build only one general classifier for such a wide range of features. Correspondingly, multiple classifiers disperse the task and difficulty, but the combination is crucial.

We cascade different-feature-based classifiers as filters in series to reject non-objects layer by layer, and eliminable non-objects can be excluded facilely by simple filters. Lastly, shape-based classification gives precise discrimination.

3.3 Shape Feature Extraction

For modeling the shape-based classifier, we normalize each region of interest (ROI), which passes through the former filters, into common size of 20×40 pixels.

The shape representation of foreground object is challenging, cause of the changeful background. Original grayscale image, with intact but complicated details, is easy to induce noise. Binary image makes the shape clearer. However, many factors reduce the effect of binarization and confuse the shape, such as dark clothing, which is potentially confounding with background. Our strategy is mainly on binary recognition, and with grayscale for assistance.

In the step of binarization, a novel adaptive method is introduced, instead of the general one. We divide the image into two parts in adaptive proportion, which is valued on vertical histogram. Fig. 1 shows our binarizing method and the improvement.

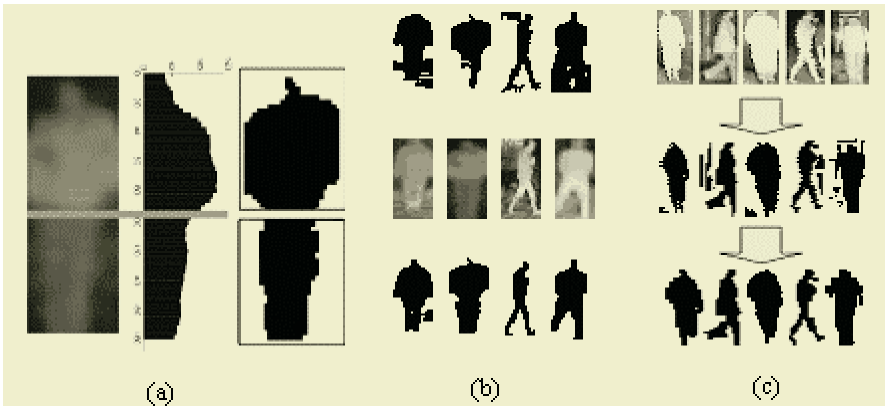


Fig. 1. Adaptive binarization. (a) Binarizing by adaptive split. (b) The comparison with general binarization: the upper one is processed by the general method, and lower one is our outcome. (c) Binarizing steps: firstly, binarizing the gray images by adaptive threshold; then processing on mathematical morphology, and hold the large connected area while filter the noises, finally, normalize the binary image.

3.4 DBNN-Based Hierarchical Classification

Decision-Based Neural Network, as [6] proposed, has hierarchical nonlinear network structure and rigorous convergence, adaptable to multi-clustering. In virtue of that, we design a particular hierarchical compound classifier, with several multiple sub-classifiers. Functionally, it has two cascaded layers.

Furthermore, two partial figures are picked out from the full region for accessorial local recognition. The inputs are divided to distinct pose categories. Details are below.

A). Structure
There is only one sub-classifier modeling on sub-cluster DBNN, whereas the second-layer is composed of multiple parallel sub-classifiers, each based on hidden-node DBNN and representing a distinct pose. The first-layer sub-classifier identifies non-pedestrians, transfers the positive-response inputs to the relative single sub-classifier in second layer for further full and partial figure recognition. Final judgment is produced by combination of the full and partial discriminate outputs.

As non-pedestrian is difficult to characterize, we turn to enhance the pedestrian shape characterization. Instead of general radial basis function (RBF), we adopt elliptic basis function (EBF), for the positive sub-classifiers

$$\psi(x, c_l) = \sum_{k=1} \alpha_{lk}(x_k - c_{lk})^2 + \theta_l \,. \tag{1}$$

B). Training
The training samples are selected from diverse scenes, and labeled manually. The sub-clusters are initialized by K-mean algorithm.

Between the two layers, training is locally individual and globally affiliated. The decision of first-layer refers to winner-take-all rule of DBNN, which is described as

$$l_p = Arg \max_p \psi(x, c_p) \,. \tag{2}$$

Due to low resolution, the indistinct object shape disadvantages classification, especially for the confusion of light with human shape, to which only grayscale information would work out. To utilize the preponderance of grayscale classification, we re-segment grayscale candidate by its binary shape, as illustration in Fig.2, for further texture recognition.

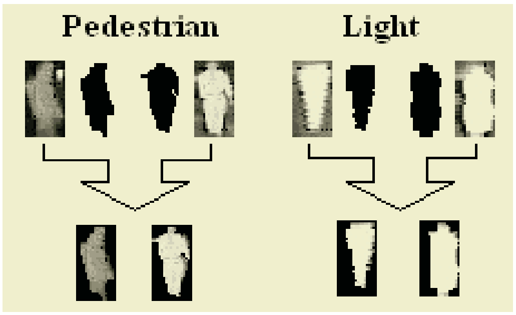


Fig. 2. Re-segment candidate by binary shape for texture recognition.

4 Experimental Results

Each cascaded filters is trained by 2840 positive and 3072 negative samples, whereas all these negative samples and 876 re-selected positive ones join the training of main compound DBNN-based classifier.

We run the system over several video sequences under different scenes. The total 568 frames, 513 of which contain pedestrian, generate 22854 segmented regions. The experiment results are in Table 1. Detection rate is estimated layer by layer.

Table 1. Experiment results for the whole system

		Non-Pedestrian Removed	Pedestrian Passed	Detection Rate
Simple Filters		20091	453	88.30%
	First-layer	2149	407	79.34%
DBNN classifier	Second-layer	94	367	71.54%

The total number of alarms is 374, with 7 false alarms. Detection rate (the number of detected positive inputs to the number of all positive inputs) of the whole system is 71.54%, and the false alarm rate (the number of false alarms to the total number of frames) is 1.23%. In respect that the numerous candidates, which are selected via segmentation stage, the rejection of non-pedestrians is significant. On the experiment results, 89.9% non-pedestrians are removed by cascaded simple filters, which present well performance.

Just for testing DBNN classification alone, other 401 positive and 1351 negative samples are collected random. The detection rate is 81.3%, whereas the false detection rate of negative samples (the number of wrong classified negative inputs to the number of all negative inputs) is 0.15%.

This system is implemented on a PC with Pentium IV 1.5GHz. Average processing speed is about 10pfs, and can runs real-timely on the whole.

5 Conclusions and Future Work

Our implementation is aimed to real-time application, and achievement of high detection rate as well as quite low false alarm rate. Out of this goal, we apply single-frame detection strategy to ensure stability, and static features are extracted for cascaded classification, including size, position, texture, sysmmetry and shape. The adaptive method of shape representation improves robustness of recognition, as well as the compound DBNN-based classifier effects fine classification. The detection allows response to both moving and stationary pedestrians.

Upon being tested on considerable video sequences, of suburban district and urban street scenes, our system can real-timely detect pedestrians, which take on various poses, and be robust to changeful background.

Furthermore, to enhance the detection reliability and development, multi-frame detection for tracking is becoming the focus, and the segmentation still should be refined.

References

1. Gavrila, D. M.: Pedestrian Detection from a Moving Vehicle. In: Proc. of the European Conference on Computer Vision, Dublin (2000) 37-49
2. Zhao, L., Thorpe, C.E.: Stereo- and Neural Network-Based Pedestrian Detection. IEEE Trans. on Intelligent Transportation Systems, **1** (2000)
3. Viola, P., Jones, M.J., Snow, D.: Detecting Pedestrians Using Patterns of Motion and Appearance. In: Proc. of the Ninth IEEE International Conference on Computer Vision (2003)
4. Broggi, A., Bertozzi, M., Fascioli, A., Sechi, M.: Shape-based Pedestrian Detection. In: Proc. IEEE Intelligent Vehicles Symposium (2000) 215-220
5. Xu, F., Fujimura, K.: Pedestrian Detection and Tracking with Night Vision. In: Proc. of IEEE Intelligent Vehicle Symposium (2002)
6. Kung, S.Y., Taur, J.S.: Decision-Based Neural Networks with Signal/Image Classification Applications. IEEE Trans. on Neural Networks (1995) 170-181

A Visual Automatic Incident Detection Method on Freeway Based on RBF and SOFM Neural Networks*

Xuhua Yang, Qiu Guan, Wanliang Wang, and Shengyong Chen

College of Information Engineering,Zhejiang University of Technology,
Hangzhou, Zhejiang 310032, China
xhyang@zjut.edu.cn

Abstract. This paper proposes a novel visual automatic incident detection method on freeway based on RBF and SOFM neural networks. Two stages are involved. First, get the freeway traffic flow model based on the RBF neural networks and use the model to obtain the output prediction. The residuals will be gotten from the comparison between the actual and prediction. Second, use a SOFM neural networks to classify the residuals to detect the incident. Because the SOFM has the character of topological ordering, the winning neuron's running trajectory on SOFM neuron array corresponds to the actual traffic state on freeway. We can observe the trajectory to detect the incident and achieve the visual traffic incident detection.

1 Introduction

This paper proposes a novel visual automatic incident detection method on freeway based on RBF and SOFM neural networks. The RBF neural networks can approximate any nonlinear mapping relationship with high rate of convergence and the SOFM neural networks have the ability to extract a system's feasure and the character of topological ordering. Combining these two neural networks to detect the traffic incident can simply the modeling greatly and achieve the visual traffic incident detection.

2 The Principle of Traffic Incident Detection

The principle of traffic incident detection is shown as figure 1.The input is the traffic parameters which can affect freeway's vehicle flow speed and vehicle flow density (In this paper, vehicle flow speed is the vehicle flow space average speed) .The output is the vehicle flow speed and the vehicle flow density. y is the actual value and $\hat{y}$ is the traffic flow model output prediction. This paper uses the RBF neural networks to get the freeway traffic flow model and use the model to obtain the output prediction.

* This work was supported by the National Natural Science Foundation of China under Grant 60374056 and 60405009

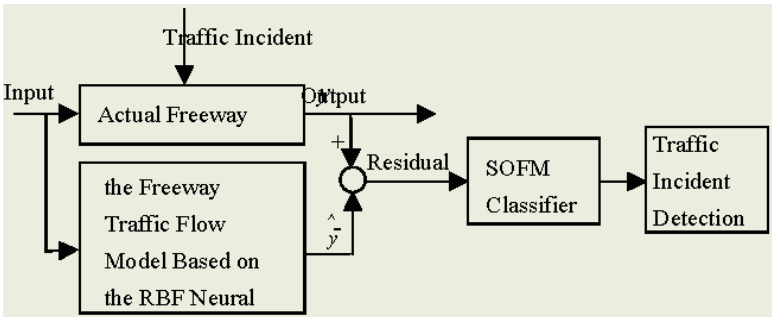

Traffic Incident
Input
Actual Freeway
Output
+
Residual
-
$\hat{y}$
the Freeway Traffic Flow Model Based on the RBF Neural
SOFM Classifier
Traffic Incident Detection

Fig. 1. The Principle of Traffic Incident Detection

Suppose both y and $\hat{y}$ are affected by noises, then

$$y = y_{real} + \xi \,. \tag{1}$$

$$\hat{y} = \hat{y}_{real} + \hat{\xi} \,. \tag{2}$$

In the formula (1) and (2), y_{real} is the real output, ξ is the measure noise, $\hat{y}_{real}$ is the traffic flow model's desired output prediction, $\hat{\xi}$ is the model noise. Suppose ξ and $\hat{\xi}$ are all noises with 0 mean value.

Define residual as formula (3)

$$\varepsilon = y - \hat{y} = (y_{real} - \hat{y}_{real}) + (\xi - \hat{\xi}) \,. \tag{3}$$

The principle of traffic incident detection: When the freeway is in the normal traffic state, the traffic flow model can simulate the actual freeway primarily and $y_{real} \approx \hat{y}_{real}$,then the residuals should meet the distribution with 0 mean value. When there is traffic incident on the freeway, the traffic capacity will fall and the traffic model can not simulate the actual freeway and $y_{real} \neq \hat{y}_{real}$,then the residuals will mutate and should not meet the distribution with 0 mean value. Using a SOFM neural networks to classify characteristics contained in the residuals can detect the traffic incident.

3 The Freeway Traffic Flow Model Based on RBF Neural Networks

3.1 The Learning Algorithm of RBF [1] Neural Networks

The RBF feedforward neural networks achieve a nonlinear mapping as following

$$f_n(x) = W_0 + \sum_{i=1}^{n} W_i \varphi(\|X - c_i\|) \tag{4}$$

$X \in R^n$ is the input vector. $\varphi(\bullet)$ is the radial basis function which achieve a mapping : $R^+ \to R$. $\|\bullet\|$ is the Euclidean norm. W_i is the weight. c_i is the center of the hidden layer ith node. n is the number of centers.

A RBF networks can usually express as

$$f_n(X) = \sum_{i=1}^{n} w_i \varphi([X - c_i]^T \Sigma^{-1} [X - c_i]) \tag{5}$$

We can use the K-means algorithm to select c_i and use a nonlinear programming algorithm to determine Σ. Finally, we can use the least squares method to solve w_i.

3.2 RBF Neural Networks Identify Traffic Flow Model

The neural networks model is a black box model. For the purpose of using the system's information fully, we firstly analyze the traffic flow analytic model[2] which was proposed by Markos Papageorgiou:

Divide a freeway into N sections and each section's length Δ_i is several hundreds meters. In each section, the traffic state can be considered uniform approximately. Namely, the traffic volume, the vehicle flow speed and the vehicle flow density in a section are constant. In each section, the lane number is constant and there is only one entrance or exit. The traffic parameters, such as traffic volume, vehicle flow speed and vehicle flow density, can all be measured. Detection period T is about dozens of seconds.

This traffic flow model is

$$\rho_i(k+1) = \rho_i(k) + \frac{T}{\Delta_i}[q_{i-1}(k) - q_i(k) + r_i(k) - s_i(k)] \tag{6}$$

$$q_i(k) = \alpha\rho_i(k)v_i(k) + (1-\alpha)[\rho_{i+1}(k)v_{i+1}(k) - r_{i+1}(k)] - s_i(k) \tag{7}$$

$$v_i(k+1) = v_i(k) + \frac{T}{\tau}\{v[\rho_i(k)] - v_i(k)\} + \frac{T\xi}{\Delta_i}v_i(k)[v_{i-1}(k) - v_i(k)] - \frac{\gamma T}{\tau\Delta_i}\frac{\rho_{i+1}(k) - \rho_i(k)}{\rho_i(k+1) + \lambda} \tag{8}$$

Where

$$v(\rho) = v_f \exp[-(1/b)(\rho/\rho_{cr})^b] \tag{9}$$

The formula (6)—(9) involves the following variables:
T sampling period: $v_i(k)$ the vehicle flow speed of i th section at KT time;
$\rho_i(k)$ the vehicle flow density of i th section at KT time;
$q_i(k)$ the traffic volume from i th section to $(i+1)$ th section at KT time;
Δ_i the length of i th section; $r_i(k)$ the on-ramp traffic volume for i th section at KT time; $s_i(k)$ the off-ramp traffic volume for i th section at KT time.

Model parameters are $v_f, \rho_{cr}, b, \tau, \gamma, \xi, \lambda, \alpha$.

After analyzing the above analytic model, we can see that there exist nonlinear mapping relationship about vehicle flow speed and vehicle flow density as following:

$$\rho_i(k+1) = f_1[q_{i-1}(k), \rho_i(k), \rho_{i+1}(k), v_{i-1}(k), v_i(k), v_{i+1}(k), r_i(k), r_{i+1}(k), s_i(k)] \quad (10)$$

$$v_i(k+1) = f_2[q_{i-1}(k), \rho_i(k), \rho_{i+1}(k), v_{i-1}(k), v_i(k), v_{i+1}(k), r_i(k), r_{i+1}(k), s_i(k)] \quad (11)$$

Construct a RBF neural networks with 9 inputs and 2 outputs. In this RBF networks, Input variables are $q_{i-1}(k), \rho_i(k), \rho_{i+1}(k), v_{i-1}(k), v_i(k), v_{i+1}(k), r_i(k),$ $r_{i+1}(k), s_i(k)$ and output variables are $\rho_i(k+1)$ and $v_i(k+1)$. Using the freeway's historical traffic data under the normal traffic state to train the RBF networks can get the traffic flow model based on the RBF neural networks.

4 SOFM Neural Networks Detect the Traffic Incident

4.1 The Learning Algorithm of SOFM [3] Neural Networks

(1) Initialize weight w_{ij} which between *ith* input neuron and *jth* output neuron.

(2) Input training samples.

(3) Calculate winning output neuron d_{win}.

$$d_{win} = \min_j \{\sum_{i=0}^{n-1} (x_i(t) - w_{ij}(t))^2\} \quad (12)$$

$w_{ij}(t)$ is the weight between *ith* input neuron and *jth* output neuron at t time.

Adjust weights according to the following formula.

$$w_{ij}(t+1) = w_{ij}(t) + \eta(t)[x_i(t) - w_{ij}(t)]N(j,t) \quad (13)$$

$\eta(t)$ -learning rate, $N(j,t)$ -neighborhood function.

(4) Repeat (2) ~ (4) up to the state in which weights have not obvious change.

4.2 Generating the Residuals

Construct a two dimensions vector which is composed of vehicle flow speed and vehicle flow speed density. Suppose the actual measure vector is $[v \quad \rho]^T$ and the traffic flow model prediction vector is $[\hat{v} \quad \hat{\rho}]^T$,then the residual vector is

$$[\varepsilon_v \quad \varepsilon_\rho]^T = [v \quad \rho]^T - [\hat{v} \quad \hat{\rho}]^T \quad (14)$$

4.3 SOFM Neural Networks Classify the Residuals

Let the residual vector as the input of the SOFM networks and the output layer adopt 10×10 two dimensions array. The. classification involves two stages.

(1) Train the SOFM and get the SOFM model

The training raw data is from this freeway section's traffic historical data including normal state's and incident state's. We can get the residual vector sequence by using the method in *4.2* and use the learning algorithm in *4.1* to train the SOFM .So, we can get the SOFM model.

(2) Apply the SOFM model to the classification
The traffic normal state and incident state correspond to different domain in the output neuron array. In practical application, substitution of the residual into the SOFM model yields a winning neuron. From the neuron's position in the array, we can detect the traffic incident. Furthermore ,Observing the running trajectory of winning neurons can real-time trace the traffic state of the freeway.

5 Simulation Research

Consider a 1800 meters freeway which is divided into 3 sections and each section's length is 600 meters. There are on-ramp and off-ramp on the first and the second section. Suppose this freeway' traffic flow model is the formula (6)-(9) and each section has the same model parameters which are as following:

$$\Delta_i = 0.5km\,,\ \alpha = 0.95\,,\ T = 12s\,,\ \xi = 1\,,\ \tau = 19.4s\,,\ \lambda = 1veh/km$$

$$\gamma = 34.7\ \ km^2/h\,,\ v_f = 98km/h\,,\ b = 3\,,\ \rho_{cr} = 32veh/km$$

This paper researches the traffic detection on the second section.

(1) Getting the traffic flow model based on RBF neural networks
We can get the model by using the method in *3.2*.
(2) Getting residual vector sequence

(a) Getting the residuals of the normal traffic state

Let the ε_v and ε_ρ meet normal distribution with 0 mean. Their variances are 1 and 0.01 respectively.

(b) Getting the residuals of the traffic incident state

Suppose the freeway's normal traffic state is in dynamic balance before the traffic incident and the balance traffic parameters are

$$v_1(0) = 66.9km/h\,,\ \rho_1(0) = 32.7veh/km\,,\ v_2(0) = 68.3km/h\,,$$

$$\rho_2(0) = 33.7veh/km\,,\ v_3(0) = 68.3km/h\,,\ \rho_3(0) = 32veh/km$$

$$r_1(k) = 100veh/h\,,\ s_1(k) = 50veh/h\,,\ (k = 1,2,\cdots)$$

$$r_2(k) = 610veh/h\,,\ s_2(k) = 500veh/h$$

When there is traffic incident on the freeway, the traffic capacity will fall and the parameters of formula (9) will be changed. At the same time, traffic capacity falling means the effective traffic length of this freeway will fall. Namely, the Δ_i of formula (6) and (8) will be decreased. Suppose the traffic incident can cause 3 different level of accidental congestion and the corresponding parameters' transformation is as following.

(i) slight congestion $v_f = 70km/h$, $\rho_{cr} = 25veh/km$, $\Delta_i = 0.55km$

(ii) middle congestion $v_f = 30km/h$, $\rho_{cr} = 15veh/km$, $\Delta_i = 0.50km$

(iii)serious congestion $v_f = 15km/h$, $\rho_{cr} = 10veh/km$, $\Delta_i = 0.45km$

After Updating corresponding parameters, we can use the traffic flow analytic model (formula (6)-(9)) to get the vehicle flow speed and the vehicle flow density of the incident state. So, we can obtain the residuals according to the principle of the figure 1.

(3) Get the SOFM model

The SOFM's output neuron layer array is as figure 2. Each neuron corresponds to a circle and the neuron's sequence number is $[(i-1)*10+j]$ (i, j are the horizontal and vertical ordinate respectively).Using the residual vector sequence which contains the normal and incident states' data to train the SOFM can get the SOFM networks model.

In figure 2,the closed-loop and its interior zone indicate the normal traffic state, the other zone outside the closed-loop indicate the traffic incident state.

(4) Detect the incident

Substitution of the real-time residual into the SOFM model yields a winning neuron. If the winning neuron's position is on the closed-loop or its interior zone, the traffic state is normal. If the position is outside the closed-loop, the traffic is in the incident state .So, we can detect the incident on the freeway.

(5) visual automatic incident detection

Recording the winning neuron's position continuously will produce the running trajectory of winning neurons on the neuron array. If the running trajectory always runs on the closed-loop or its interior zone, the traffic state is normal. If the running trajectory runs outside the closed-loop, the traffic is in the incident state .So, observing the running trajectory of winning neurons on the array can visually trace the traffic state on the freeway and achieve the visual automatic incident detection.

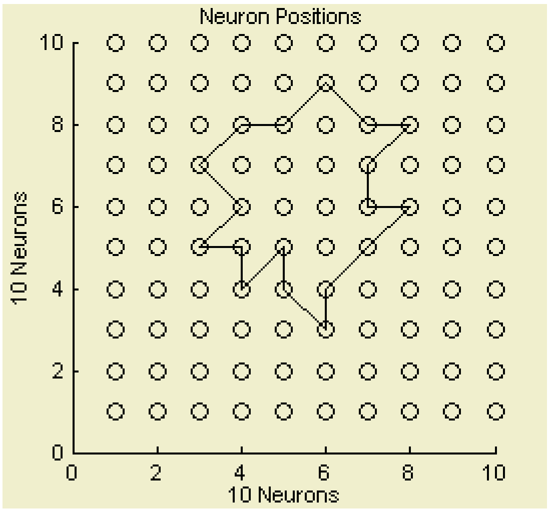


Fig. 2. SOFM output layer neuron array

6 Conclusion

This paper proposes a novel visual automatic incident detection method on freeway based on RBF and SOFM neural networks. This method can detect the traffic incident by observing the winning neuron's running trajectory on SOFM neuron array and achieve the visual automatic incident detection. The simulation research also shows that this method is effective.

References

1. Chen, S., et al.: Orthogonal Least Square Learning Algorithm for Radial Basis Function Networks. IEEE Trans. Neural Networks, **2** (1991) 302-309
2. Shi, Z., Huang, H., Qu, S., Chen, X.: Traffic: Control System Introduction, Science Publishing House (2003)
3. Kohonen, T.: Self-Organizing Maps. NY: Springer–Verlag (1995)

A Self-organizing Map Method for Optical Fiber Fault Detection and Location*

Yi Chai[1,2], Wenzhou Dai[1], Maoyun Guo[1], Shangfu Li[3], and Zhifen Zhang[3]

[1] College of Automation, Chongqing University, Chongqing 400044, China
[2] The Key Laboratory of High Voltage Engineering and Electrical New Technology of Ministry of Education, Chongqing University, Chongqing 400044, China
chaiyi@cqu.edu.cn, dai_7911@yahoo.com.cn, gomorning@sina.com
[3] Xichang Satellite Launch Center, Xichang, Sichuang 615000, China

Abstract. As optical fiber is subject to faults, normal communication will be affected. An intelligent method of detection and location for communication optical fiber is put forward in this paper. According to spatial characters of geographic distributing of optical fiber network, nodes and links topo model of the network is built. Adopting the ANN algorithm in this paper, the nodes are classified according to the structure of optical fiber communication network, an effective ergodicity detection strategy of nodes and links is built, and free optical fiber and optical cable are detected termly. Through the simulation, the method which is put forward in this paper is validated. When optical fiber communication network extend, the method can form a new ergodique detection strategy of nodes and links based on the nodes classified, and the on-line dynamic detection of communication optical fiber can be realized.

1 Introduction

Modernized cable communication is all set up on a high-density, large-capacity, high-speed, interference-proof and confidential cable network. Cable, which is the channel to transmit signal, is the important component of the whole cable communication system. Statistics show the main cause of communication intermission is cable line fault, which takes more than two thirds of transmission fault. Therefore, the maintenance of the line is very important. Search of the fault is one of the important jobs in cable line maintenance, and OTDR is the key instrument to identify the fault. It works as the Rayleigh scattering principle, which analyze the situation of each point by collecting back-direction scattering signal curve.

As communication network scale expands, the number of nodes on network increases. When fault or intermission occurs, passive fault exclusion and inspection and rush repair cannot ensure proper operation. The intelligent initiative detection and maintenance on cable line is right the good method, which examines those disengaged cable periodically, solve problems once they are detected, thus avoid the impact of line fault on communication. In this paper, the nodes are classified dynamically by ANN algorithm, according to the structure of cable communication network; ergodic detection strategy of network is discussed. The research production is essential to

* This work was supported by 863 Program Project of China (Grant No.2003AA132050) and China Postdoctoral Science Foundation (Grant No.2004035154)

establish the intelligent monitoring system of cable network, to realize the general control of maintenance center, and detect and maintain cable network on line.

2 Fault Detection and Location of Communication Cable

It's very important to locate the fault site of communication cable. Rapidly and accurately locating can reduce the time of rush to repair, and decrease the loss. In the common case, people can get the distance from the fault site to the measuring site by OTDR(Optic Time Domain Reflectometer). With the line's property sheet and the connection of map spatial property data, we can locate the spatial site of communication cable fault site, get the detail spatial map intuitively, and the multimedia information of surrounding.

3 Algorithm of Fault Detection and Location Based on SOM Neural Network

3.1 Description of Topo Structure of Cable Communication Network

Optical fiber transmission network is one practical network, which distributes in actual geographical site, and its topo structure use the combination of basic structure or anamorphic topo structure according to the factual complexion of every city, as usual for netted, orbicular, asteroidal and dendriform structure. Figure 1 is a network figure of the cable of a certain city,1-15 represent the nodes of the cable,ONU1-ONu6 represent the optical network unit(ONU),and A-L represent the users on the cable which is connected with ONU by wire.

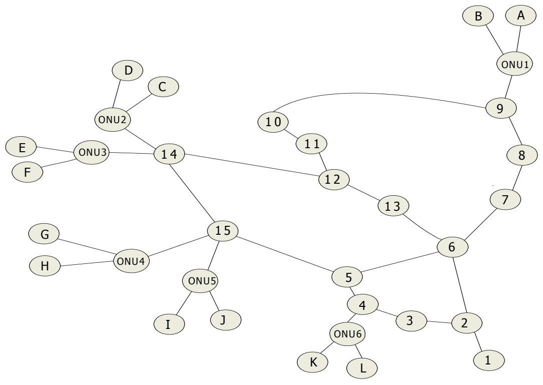


Fig. 1. Data structure of cable network datasets

Analysis and resolution for cable chain is based on network model. network model is a system made of many lines connected each other.. Network data model is one abstract expression about realistic network system, thereinto, lines represent network links and the intersected point represent network node. In the network model, resource and information can be transferred from one node to another node.

IN the process of network model being built, the computer description of cable network structure is that datasets of communication line is computed and stored in the node-chain datasets, and built connected topo relation between nodes and links. The results of topo is that it can create network datasets of optical fiber network, and increase four fields which have the character of topo and network, section name, original node(SmFnode), terminative node(SmTnode) and length of chain in the line-chain table. At the same time, it create node tables as table 1 shows. Node tables exist for the sub dataset of network dataset and include the coordinate of actual geographical sites.

Table 1. Table between network nodes and adjacent links

name	(SmFNode)	(SmTNode)	length	name	(SmFNode)	(SmTNode)	length
1	1	2	d_1	13	10	11	d_{13}
2	2	3	d_2	14	11	12	d_{14}
3	3	4	d_3	15	12	13	d_{15}
4	4	5	d_4	16	6	13	d_{46}
5	5	6	d_5	17	12	14	d_{17}
6	2	6	d_6	18	14	ONU2	d_{18}
7	4	ONU6	d_7	19	14	ONU3	d_{19}
8	6	7	d_8	20	14	15	d_{20}
9	7	8	d_9	21	15	ONU4	d_{21}
10	8	9	d_{10}	22	15	ONU5	d_{22}
11	9	ONU1	d_{11}	23	5	15	d_{23}
12	9	10	d_{12}				

3.2 Algorithm of Fault Detection and Location Based on SOM Neural Network

As communication network scale expands, the number of nodes on network increases,which continually dynamically extend to beside road and to the front of the house, and it must enhance the burden of cable maintenance.

Building the intelligent initiative detection and maintenance system on cable line and using network topo structure to analyze the network performance, we can carry out unite schedule of maintenance center. on-line optical fiber detection and maintenance. Actually, build valid ergodic detection strategy of node-line network and make the system automatically detect the cable line.

3.2.1 Algorithm of Network Nodes Sorted

SOM (Self-Organizing Map) [1] is a competitive learning network, and can self-organize and self-learning without monitored. An important feature of neural networks is the ability to learn from their environment, and through learning to improve performance. SOM's basic idea is that: when a network receive outside input, it can be divided into some unequal areas and neuron can achieve different respond to input pattern by reciprocity and competition each other. After outside input pattern appears, all nerve cells compete each other at one time and try to imitate input signal and make similar inputs explode the same neurons to distinguish different, consequently, complete the function sorted by the similarity of input mode [2].

There are three basic steps involved in the application of the algorithm after initialization: sampling similarity match and updating [3]. These three steps are repeated until formation of the feature map has completed. The algorithm is summarized as follows:

Step1:Initialization. weight vectors wj(0), j=1,…,N(the number of neurons).

Step2:Sampling.

Step3:Similarity Matching, $i(x) = \arg\min \| x(n) - w_j \|, j = 1, \cdots, N$.

Step4:Updating.Adjust the synaptic weight vectors of all neurons by using the update formula

$$w_j(n+1) = \begin{cases} w_j(n) + \eta(n)[x(n) - w_j(n)], j = \Lambda_{i(x)}(n) \\ w_j(n), \qquad\qquad otherwise \end{cases}$$

$\eta(n)$ is learning-rate, $\Lambda_{i(x)}$ is neighborhood function.

Step5:Continuation.Continue with step2 until no noticeable changes in the feature map are observed.

The sort algorithm of SOM can be summarized as follows:

Step1:Input the coordinate of network nodes and the distances between network nodes.

Step2:Initialize SOM Neural Network model and set the numbers of the neuron(the numbers of data).

Step3: Establish SOM Neural Network.

Step4:Train by SOM Neural Network and get weight vectors of all neurons.

Step5:Find and display classified results.

Based on the results sorted by SOM Neural Network, we can divide the whole network nodes into some sorts, then carry out ergodic detection strategy of network to every sort and attain the detection for whole network.

3.2.2 Ergodic Detection and Location Strategy of Network

Optical fiber communication network is made up of nucleus node, concentration node, optical fiber admeasure plank, transmission optical fiber and so on. In analysis, we can transform actual geographical into corresponding network topo structure, which we can analyze network performance based on[4],[5].

After sorting communication cable nodes using SOM, we can get m classified node sets, $NodeSet = \{n_1, n_2, \cdots, n_m\}$ .Compare the classified result of network nodes with the table between network nodes and adjacent links(table 1) and make a node set $n_i, i = 1,2,\cdots,m$ as a node set for detection a time, then make nodes that connect with n_i in table 1 up of a chain detected set, as 1-2-3-4-ONU6 in figure 1.By detecting whether in n_i node set node set and sub node set are connected, we can achieve the detection of every cable section in communication. When communication enlarged, network nodes and links in figure 1 change and node sets in table 1 update at the same time to come into being a new classified node set $n_i, i = 1,2,\cdots,m$. We can change the number of the sort based on the need of optical fiber management and classify right sets. Then we can go on ergodic detection again as above, and attain dynamic on-line detection for communication cable.

The process of detection strategy can be simply described as follow:

Step1: Classify network nodes by SOM.

Step2: Traversing all nodes and arcs in every sort and combining the information OTDR displays, we can detect whether there have faults in cable. If no faults, then display it is normal; otherwise display it is abnormal and transfer to step 3.

Step3: Analyzing the ergodic path and combining OTDR, we can find the fault site and display it.

When the scale of communication network changes, SOM algorithm can dynamical carry out ergodic detection based on network structure, and complete fault detection and location.

4 Simulating Analysis of Cable Communication Network

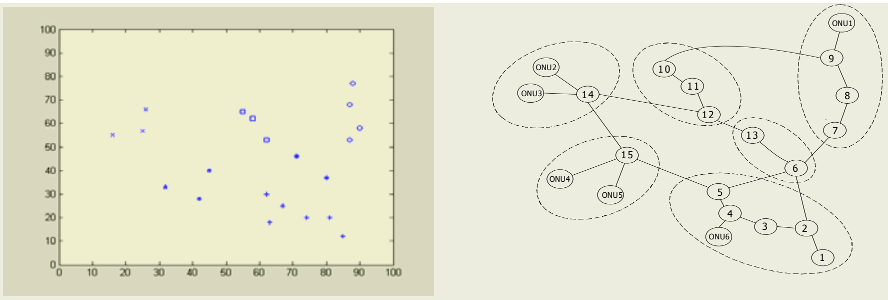
In communication network, every node has at least two line to connect with other nodes and make up of a communication loop, which make communication network be capable of ergodic detection. Table2 show the coordinate of network node in figure 1. Sorting the communication nodes by SOM, we get six classified node sets,

$NodeSet = \{(1,2,3,4,5,ONU6),(6,13),(7,8,9,ONU1),(10,11,12),(14,ONU2,ONU3),$
$(15,ONU4,ONU5)\}$. The classified results show as follow:

Table 2. Network node coordinate

node	(X,Y)	node	(X,Y)	node	(X,Y)
1	85,12	8	90,58	15	45,40
2	81,20	9	87,68	ONU1	88,77
3	74,20	10	55,65	ONU2	26,66
4	67,25	11	58,62	ONU3	25,57
5	62,30	12	62,53	ONU4	32,33
6	80,37	13	71,46	ONU5	42,28
7	87,53	14	16,55	ONU6	63,18

By traversing optical fiber of every node set in figure 2, we can attain the quality status of cable links among nodes. For example, in figure 2, if node 1 represents communication terminal. We implement ergodic detection for classified node set



(a) classify nodes by SOM (b) classified node sets separated by ellipse

Fig. 2. Classified results of network node

*NodeSet*1={1,2,3,4,5, *ONU*6}. First, we traverse the node and the arc in the sort *NodeSet1* and can get two links detection strategy:(1-2-3-4-ONU6-4-5) or (1-2-4-ONU6,1-2-3-4-5);Second, based on the nodes in the sort, detect the links that connect with other sort and get the path((1-2-6),(1-2-3-4-5-15),(1-2-3-4-5-6)).Using the similar disposal method to all sort *NodeSet(i)*,*i=2,3,4,5,6*, the whole network can be detected.

If it exists faults in optical fiber, we can get the data of cable length measured from OTDR from fault point to maintenance center computer room, combining detection links based on links detection strategy, we can analyze and attain the fault site of optical cable.

5 Conclusion

Optical fiber detection is very important for the steady running of optical cable network. On the optical cable network having worked, the maintenance and detection shall run without interrupting the communication, so we usually detect the state of optical fiber by detecting free optical fiber. But, as communication network scale expands, the number of nodes on network increases more and more, the network is more complex. It is a crucial problem to effectively detect communication network. The paper brings forward a test and detection strategy of optical fiber, which can send out the alarm message and confirm the fault site by detecting links. The research production applies on detection and location of cable fault, and emergency process system, which can availably detect optical fiber on-line combining with communication cable GIS system, and improve the ability of fault detection and location, of dealing with broken optical fiber quickly.

References

1. Haykin, S.: Neural Networks a Comprehensive Foundation, Publish House Tsinghua University (2002) 443-483
2. Yan, P., Zhang, C.: Artificial Neural Network and Simulating Evolution Computing. Beijing, Publish House Tsinghua University (2000)
3. Yanagida, T, Miura, T., Shioya, I.: Classifying News Corpus by Self-organizing Maps. Communications. Computers and signal Processing, IEEE Pacific Rim Conference on, **2** (2003) 800-803
4. Ge, M., Du, R., Xu, Y.: Fault Detection Using Hierarchical Self-organizing Map. Robotics, Intelligent Systems and Signal Processing, IEEE International Conference on, **1** (2003) 565-570
5. Neagoe, V.-E., Ropot, A.-D.: Concurrent Self-organizing Maps for Pattern Classification. Cognitive Informatics (2002) 304-312

Anomaly Internet Network Traffic Detection by Kernel Principle Component Classifier*

Hanghang Tong[1], Chongrong Li[2], Jingrui He[1], Jiajian Chen[1], Quang-Anh Tran[2], Haixin Duan[2], and Xing Li[2]

[1] Department of Automation, Tsinghua University, Beijing 100084, China
{walkstar98,hejingrui98}@mails.tsinghua.edu.cn
[2] Network Research Center of Tsinghua University, Beijing 100084, China
{licr,qa,dhx,xing}@cernet.edu.cn

Abstract. As a crucial issue in computer network security, anomaly detection is receiving more and more attention from both application and theoretical point of view. In this paper, a novel anomaly detection scheme is proposed. It can detect anomaly network traffic which has extreme large value on some original feature by the major component, or does not follow the correlation structure of normal traffic by the minor component. By introducing kernel trick, the nonlinearity of network traffic can be well addressed. To save the processing time, a simplified version is also proposed, where only major component is adopted. Experimental results validate the effectiveness of the proposed scheme.

1 Introduction

Intrusion detection has received great attention from researches in the past years [5], [6]. It has been widely recognized that a malicious intrusion or unauthorized use could cause severe damages [11], [13]. According to [2], an intrusion can be defined as "any set of action that attempts to comprise the integrity, confidentiality or availability of information resources". Existing intrusion detection methods can be classified into two categories [13]: misuse detection [8], [9] and anomaly detection [11], [13]. Compared with misuse detection, anomaly detection does not need any prior knowledge of attack, and therefore can detect novel attack types. With the development of Internet, it has become a crucial issue from both application and theoretical point of view [5], [6].

Almost, if not all, existing anomaly detection methods are based on the following assumption [5], [6], [11], [13]: the network traffic with attack has different statistical character compared with that without attack. The main difficulties of anomaly detection lie in two aspects. On one hand, there are a lot of different kinds of attacks. For example, the authors in [4] identified five cases where anomalies present in attack traffic, including user behavior, bug exploits, response anomalies, bugs in the attack and evasion. The statistical nature of attack traffic might vary dramatically throughout different attack types. On the other hand, even for the normal traffic, its statistical nature is very complex. Two of the most important discoveries of the statistics of

* This work is supported by National Fundamental Research Development (973) under the contract 2003CB314805.

anomaly Internet traffic over the last ten years are that Internet traffic exhibits self-similarity [3], [7], [12] (in many situations, also referred as long-range dependence) and non-linearity [1], [12]. The diversity of attack types and the complex statistical nature of network traffic make highly accurate anomaly detection very difficult.

In the past years, many methods have been proposed for anomaly network traffic detection. Statistical-based techniques build a norm profile and make use of statistical tests to perform anomaly detection [5]. A representative work in this category is based on chi-square [15]. More recently, researchers have applied machine learning technique to anomaly detection (See [6] for a detailed review). For example, the authors in [13] proposed using One-Class Support Vector Machine (OCSVM); the authors in [11] proposed a robust principle component classifier (PCC) for anomaly detection and the experimental results on KDD-99 dataset showed its superiority over existing methods. However, as a linear dimension reduction method, PCC cannot capture the non-linearity of network traffic, and therefore, its effectiveness might be compromised.

To deal with the non-linear nature of Internet traffic, in this paper, we introduce kernel trick into principle component classier (PCC) and propose a novel anomaly network traffic detection scheme, namely kernel principle component classifier (KPCC). Like PCC, KPCC can detect anomaly network traffic which has extreme large value on some original feature by the major component, or does not follow the correlation structure of normal traffic by the minor component. As a non-linear dimensionality reduction method, Kernel PCA ensures that the non-linear nature of network traffic can be well addressed. However, PCC cannot achieve this goal. We also proposed a simplified version of KPCC (SKPCC), in which only major component is used. In SKPCC, the processing time for solving eigen-problem can be greatly reduced since only the first several eigen-values and eigen-vectors are needed, which is a desirable property for real applications. Experimental results on DARPA 1999 dataset demonstrate the effectiveness of the proposed methods.

The rest of this paper is organized as follows: in Section 2, we present our Kernel Principle Component Classifier in detail; experimental results are given in Section 3; finally, we conclude the paper in Section 4.

2 Kernel Principle Component Classifier

2.1 Notation

Let $\{X_i, i=1,\cdots,N\} \in R^m$ the training set and $X_{test} \in R^m$ a testing sample;

Let $K=(k(x_i,x_j))_{i,j}$ the dot product matrix defined on training set by a certain type of kernel [10], [14]. Let $\{(\lambda_i,e_i); i=1,\cdots,N\}$ be the eigen-spectrum of K, where is λ_i is the i^{th} largest eigen-value and e_i is the corresponding eigen-vector;

Let $Y_i=[y_i^1,\cdots,y_i^p]^T$ be the principle component of X_i $(i=1,\cdots,N)$, where $y_j(j=1,\cdots,p)$ is the j^{th} principle component. Similarly, Let $Y_{test}=[y_{test}^1,\cdots,y_{test}^p]^T$ be the principle component of X_{test};

Let $\lambda_j (j=1,\cdots,q)$ and be the major eigen-value, and $C_{maj}(X_i)=\sum_{j=1}^{q}\frac{y_i^j}{\lambda_j}$ the major component of X_i. Let $\lambda_{p-r+j} (j=1,\cdots,r)$ and be the minor eigen-value, and $C_{min}(X_i)=\sum_{j=p-r+1}^{r}\frac{y_i^j}{\lambda_j}$ the minor component of X_i.

2.2 Architecture

Like in [13], our system contains three modules as shown in Fig. 1:

- ♦ The collection module sniffs network traffic to 1) form the training set $\{X_i, i=1,\cdots,N\}$, where all traffic are normal and 2) prepare a testing sample X_{test}.
- ♦ The training module generates KPCC or SKPCC by the training set.
- ♦ The Testing module determines whether or not the testing sample X_{test} is attack.

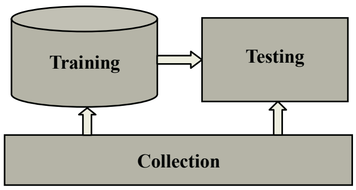


Fig. 1. Architecture of the proposed system

2.3 Algorithm

The network traffic is likely to be an attack if its statistical nature is different from those normal ones. In KPCC, it indicates that such traffic whether 1) has extremely large value on some of its original feature; or 2) it does not the correlation structure of normal traffic. While the former character can be capture by the major component, the latter can be described by the minor component [11]. Based on the above observation, KPCC can be designed as follow:

KPCC for anomaly traffic detection

- ♦ If $C_{maj}(X_{test})=\sum_{j=1}^{q}\frac{y_{test}^j}{\lambda_j}>c_1$, or $C_{min}(X_{test})=\sum_{j=p-r+1}^{r}\frac{y_{test}^j}{\lambda_j}>c_2$, X_{test} is an attack;
- ♦ Else X_{test} is a normal traffic.

Where c_1 and c_2 parameters, which can be determined by the specified false alarm rate (α_1 and α_2) from the training set:

$\alpha_1 = P\left(C_{maj}(X_i)>c_1 \mid X_i \text{ is normal}\right)$; and $\alpha_2 = P\left(C_{min}(X_i)>c_2 \mid X_i \text{ is normal}\right)$.

If the dot product matrix $K=(k(x_i,x_j))_{i,j}$ is replaced by the correlation matrix of $\{X_i, i=1,\cdots,N\}\in R^m$, KPCC is degraded into PCC. However, unlike in PCC, the non-linearity within network traffic can be well described by kernel trick in KPCC.

Note that, in both PCC and KPCC, we need to get the complete eigen-spectrum to get the minor component. When the dimension is high, which is often the case for a real problem, the processing time might be much high. To address this issue, we also propose a simplified version of KPCC (SKPCC), in which only major component is used:

SKPCC for anomaly traffic detection

- ♦ If $C_{maj}(X_{test}) = \sum_{j=1}^{q} \frac{y_{test}^{j}}{\lambda_j} > c_1$, X_{test} is an attack;
- ♦ Else X_{test} is a normal traffic.

3 Experimental Results

DARPA 1999 data (http://www.ll.mit.edu/IST/ideval) is used to evaluate the performance of KPCC and SKPCC. The data in the first week is attack free, while that in second week contains various types of attack as listed in Table 1. As in [13], we use the *inside-tcpdump* dataset in the first week as the training set, and that in the second week as the testing set. To perform a fair comparison, the same types of statistics in [13] are adopted as listed in Table 2, which are generated by TCPSTAT (http://www.frenchfries.net/paul/TCPSTAT).

Table 1. Attack information in the second week

Day	Attack	Destination	Start Time	End Time
Tue	Portsweep	172.16.114.50	21:44:15	22:11:11
	Mailbomb	172.16.112.50	03:25:10	03:35:06
	Ipsweep	172.16.112.0/23	02:05:13	02:29:14
Wed	Satan	172.16.114.50	01:02:09	01:04:38
	Mailbomb	172.16.112.50	02:44:13	02:54:08
	Ipsweep	172.16.112.0/23	09:17:04	09:29:13
Thu	Satan	172.16.114.50	22:33:20	22:35:37
	Portsweep	172.16.114.50	23:50:07	00:07:31
	Neptune	172.16.114.207	00:04:12	00:07:37
	Ipsweep	172.16.112.0/23	05:36:06	05:39:33
Fri	Neptune	172.16.114.50	00:20:11	00:23:36
	Portsweep	172.16.112.50	06:13:02	06:25:06

Table 2. Traffic statistics used in KPCC/SKPCC

TCPSTAT Output Options	Traffic Statistics
%C	# of ICMP packets
%T	# of TCP packets
%U	# of UDP packets
%a	Mean of packet size
%d	Deviation of packet size

There are a set of parameters and operations that need to be set in KPCC and SKPCC:

- p is set so that the major eigen-value accounts for 50% of the total variance;
- r is set so that the minor eigen-value accounts for 0.02% of the total variance;
- The duration to generate the traffic statistics is set to be 300s;
- RBF kernel is adopted to formulate the dot product matrix $K = (k(x_i, x_j))_{i,j}$, that is, $k(x_i, x_j) = \exp(\frac{\|x_i - x_j\|^2}{\sigma})$, where $\sigma = 1$;
- $\alpha_1 = \alpha_2$; $\alpha_1 + \alpha_2$ is the total specified false alarm rate.

The receiver operating characteristic (ROC) curve on the testing set by KPCC and SKPCC is plotted in Fig. 2. It is compared with that by OCSVM [13], and PCC [11]. The recall and precision by KPCC are given in Table 3, with specified total false alarm rate 10%.

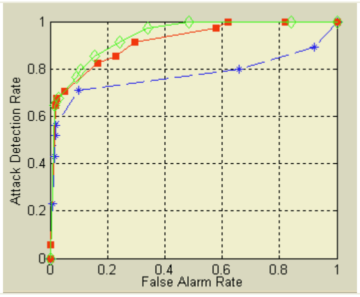


Fig. 2. ROC curve on the testing set by different anomaly detection schemes

Table 3. Recall and Precision of KPCC (10% total specified false alarm rate)

Predicted / Actual	Attack	Normal	Recall
Attack	*TP=29*	*FN=5*	85.3%
Normal	*FP=193*	*TN=1058*	84.6%
Precision	13.1%	99.5%	

4 Conclusion

In this paper, we have proposed a novel anomaly network traffic detection scheme, namely kernel principle component classifier (KPCC). KPCC can capture two kinds of anomaly traffic, which have some extremely large values on some original feature, or do not follow the correlation structure of normal traffic. By introducing kernel trick into existing PCC, KPCC can well address the non-linearity of network traffic. To save the processing time, a simplified version of KPCC is also proposed, where only major component is used. Experimental results demonstrate the effectiveness of the proposed scheme. Future work includes: 1) investigating other kinds of kernel in

KPCC (SKPCC); 2) exploring other traffic statistics; 3) exploring the relationship between KPCC (SKPCC) and spectral methods.

References

1. Hansegawa, M., Wu, G., Mizuno, M.: Applications of Nonlinear Prediction Methods to the Internet Traffic. The 2001 IEEE International Symposium on Circuits and Systems, (2001) 169-172
2. Heady, R., Luger, G., Maccabe, A., Servilla, M.: The Architecture of a Network Level Intrusion Detection System. Tech Report, University of New Mexico, (1990)
3. Leland, W.E., Taqqu, M.S., Willinger, W., Wilson, D.: On the Self-similar Nature of Ethernet Traffic. IEEE/ACM Tran. on Networking, (1994) 1-15
4. Mahoney, M., Chan, P.K.: Learning Nonstationary Models of Normal Network Traffic for Detecting Novel Attacks. SIGKDD, (2002) 376-385
5. Markou, M., Singh, S.: Novelty Detection: A Review Part1: Statistical Approaches. Signal Processing, (2003)
6. Markou, M., Singh, S.: Novelty Detection: A Review Part2: Neural Network-based Approaches. Signal Processing, (2003)
7. Ostring, S., Sirisena, H.: The Influence of Long-rang Dependence on Traffic Prediction. IEEE ICC, (2001) 1000-1005
8. Paxson, V.B.: A System for Detecting Network Intruders in Real-Time. Lawrence Berkley National Laboratory Proceedings, 7'th USENIX Security Symposium, (1998)
9. Roesch, M.: Snort - Lightweight Intrusion Detection for Networks. Proceedings of USENIX Lisa'99, (1999)
10. Scholkopf, B., Smola, A.J., Muller, K.R.: Nonlinear Component Analysis as a Kernel Eigenvalue Problem. Neural Computation, (1998) 1299-1319
11. Shyu, M.L., Chen, S.C., Sarinnapakorn, K., Chang L.W.: A Novel Abnormal Detection Scheme Based on Principle Component classifier. ICDM (2003)
12. Tong, H., Li, C., He, J.: A Boosting-Based Framework for Self-similar and Non-linear Interet Traffic Prediction. ISNN (2004) 931-936
13. Tran, Q.A., Duan, H., and Li, X.: One-Class Support Vector Machine for Anomaly Network Traffic Detection. APAN (2004)
14. Vapnik, V.N.: An Overview of Statistical Learning Theory. IEEE Trans on Neural Networks, (1999) 988-999
15. Ye, N., Chen, Q.: An Anomaly Detection Technique Based on a Chi-Square Statistic for Detecting Intrusions into Information Systems. Quality and Reliability Eng Int'l, (2001) 105-112

Intelligent Hierarchical Intrusion Detection System for Secure Wireless Ad Hoc Network

Peng Fu, Deyun Zhang, Lei Wang, and Zhongxing Duan

Department of Computer Science and Technology, Xi'an Jiaotong University,
Xi'an, Shaanxi 710049, China
fupeng@lzu.edu.cn

Abstract. The nature of mobile ad hoc network (MANETs) makes them vulnerable to security attacks. Intrusion detection is a vital secure scheme for wireless mobile ad hoc networks (MANETs). However, the generalizing ability of current IDS is poor when given less priori knowledge, and a centralized intrusion detection network security solution does not work well in MANETs. In this paper we discuss the characters of security issues in ad hoc networks and propose a Support Vector Machine (SVM) based distributed hierarchical intrusion detection system that adapt to the characters of current ad hoc networks. Performance evaluations based on simulation experiment manifested our proposed approach is efficient and effective.

1 Introduction

Wireless ad hoc network is a kind of new technique of wireless communication for mobile hosts. In an ad hoc network, there is no fixed infrastructure [1] such as base stations or mobile switching centers. Mobile nodes that are within each other's radio range communicate directly via wireless links, while those far apart rely on other nodes to relay messages as routers. While the great flexibility of MANETs also brings many great challenges, the security is an important issue for ad hoc networks, especially for those security-sensitive applications. These networks are vulnerable to attackers due to MANETs' characteristics of open medium, dynamic network topology, and limited bandwidth, distributed cooperation and constrained energy resources. Existing security solutions for conventional network may not be adequate and cannot be applied directly to MANETs. Intrusion Detection and Response System (IDS) [2],[3] construct the vital line defense of an ad hoc network, and the major challenge is how to make sure the IDS can catch the most damaging attacks. In case of MANETs the only available audit data is restricted to the communication activities taking place within the radio range, and any IDS for these types of networks should be made to work with this localized kind of audit data. Anomaly Detection models of IDS cannot be used for MANETs, since the separating line between normalcy and anomaly is obscure. A node that transmits erroneous routing information (fabrication) can be either a compromised or is currently out of sync due to volatile physical movement, and it is difficult to distinguish between false alarms and real intrusions. Therefore, a centralized IDS network security solution cannot work well in MANETs, and there is not sufficient priory knowledge for intrusion detection. And the generalizing ability of current IDS is poor when given less priori knowledge. In this paper, we propose a Support Vector Machine (SVM) based intrusion detection system for MANETs. It is a distributed and hierarchical model. The architecture of our IDS is designed to adapt to the characters of MANETs, such as topology, routing method, secure goals etc.

2 Architecture of Intelligent Hierarchical Intrusion Detection System for Ad Hoc Networks

The architecture of ad hoc network can be classified into two types, flat architecture and hierarchical architecture. Traditional ad hoc networks have flat architecture, which suffers from poor scalability. One way to overcome such the capacity constraints is to use hierarchical network architectures and hierarchical network architecture also has good scalability and flexibility. What this statement means is that every node in the wireless ad hoc network should participate in intrusion detection. Each node is responsible for detecting intrusion locally and independently but neighboring nodes can form an association and collaboratively investigate in a broader range.

2.1 Intelligent Hierarchical Distributed IDS Architecture

Our Hierarchical IDS architecture based showed in Figure 1.

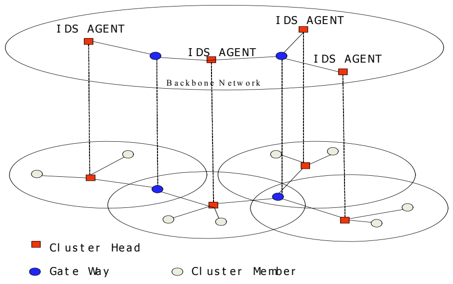


Fig. 1. The Hierarchical Distributed IDS Architecture for Ad hoc network

Each cluster head node within the network has its own individual IDS agent and these agents run independently and monitor user and system activities as well as communication activities within the cluster area. The functions of each IDS agent in those cluster head include local data collection and SVM-Based local intrusion detection. The cluster head also collects detection results from other cluster heads. If an anomaly is detected in the local data or if the evidence is inconclusive, IDS agents on the neighboring cluster head nodes will cooperatively participate in a global intrusion detection scheme. The system proposed is distributed and hierarchical in nature that fit to the ad hoc network's hierarchical topology. The advantage of this architecture is that the data collected by cluster head may be more comprehensive, which enhances the reliability of detection results. But, it is based on hierarchical clustering scheme, and how to select a cluster head effectively in a dynamically changing environment, poses another problem. It should be based on the assumption that the numbers of malicious nodes are few as compared to the network size; otherwise, the scheme fails.

2.2 Support Vector Machines for Intrusion Detection [4]

The basic idea of SVM classifier is to virtually map the audit dataset to a very high dimensional space using kernel function, and then find a hyperplane in the mapped high dimensional space to separate the normal and abnormal patterns. The idea works based on a heuristic that non-separable data in low dimension can be separated by a hyperplane in a sufficiently higher dimension. Here, we think the intrusion detection as the process of pattern recognition, in which the abnormal data patterns are to be recognized from the normal patterns. The hyperplane obtained works as the decision boundary between the two data classes, normal and abnormal. A new unlabeled data point can be classified by putting it into the decision function and using sign function to decide which class it belongs to.

For samples $\Omega = \{(x_i, y_i) \mid i = 1,2,...,N\} \subset R^n \times \{-1,1\}$ and function $< \Phi(x_i), \Phi(x_j) >= K(x_i, x_j)$. Standard SVM can be expressed as following:

Min $Q(\omega,\xi) = \frac{1}{2} \|\omega\|^2 + C\sum_{i=1}^{l} \xi_i$, **s.t.** $y_i[< \omega \cdot \Phi(x_i) > +b] - 1 + \xi_i \geq 0$, $\xi_i \geq 0$, $i = 1,...,l$

(i) In the case of linear separable for SVMs, *K(xi,xj)* is linear transformation, and C=0, $\forall \xi_i$ =0.

(ii) In the case of linear non-separable, *K(xi,xj)* is a non-linear mapping from Ω to a higher dimension space H (so that Dim (H)>Dim(Ω)), and satisfied Mercer condition, thus *K(xi,xj)* is the kernel function, where C>0 is the constant which decides the degree of misclassification, and $\sum_{i=1}^{l} \xi_i$ is the up-bound of all misclassification samples. In fact, it can be formulated as a generic expression: $F_\sigma(\xi) = \sum_{i=1}^{l} \xi_i^\sigma$ .The case σ =1 relates linear cost function, and case σ =2 is corresponding to square cost SVM. Solution to SVM results in the following QP problem:

Min $W(\alpha) = -\sum_{i=1}^{l} \alpha_i + \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j K(x_i \cdot x_j)$, **s.t:** $0 \leq \alpha_i \leq C, i = 1,...,l;$ $\sum_{i=1}^{l} \alpha_i y_i = 0;$

Thus the generic decision hyperplane can be formulated as:

$$f(x) = \text{sgn}(\sum_{S.V.} y_i \alpha_i K(x_i \cdot x) + b). \tag{1}$$

The alternative choice of Kernel function except for the mostly used ones such as Polynomial, Gaussian RBF and Ridge Kernel include: $\| \vec{x} - \vec{y} \|^{2n} \ln(\vec{x} - \vec{y} \|)$ (used in Splines) and $\frac{\sigma}{\pi} \sin c[\frac{\sigma}{\pi}(x - y)]$.

We choose SVM classifier in the system because of that it is a new category of machine learning method, which has been proved to be able to provide a good noise-tolerance performance on pattern recognition problems, it works well on less priori knowledge. Previous study[5] showed that SVM can do the intrusion detection with higher detection accuracy.

2.3 SVM-Based IDS Agent Architecture

The IDS agent consists of following modules showed in Figure 2.

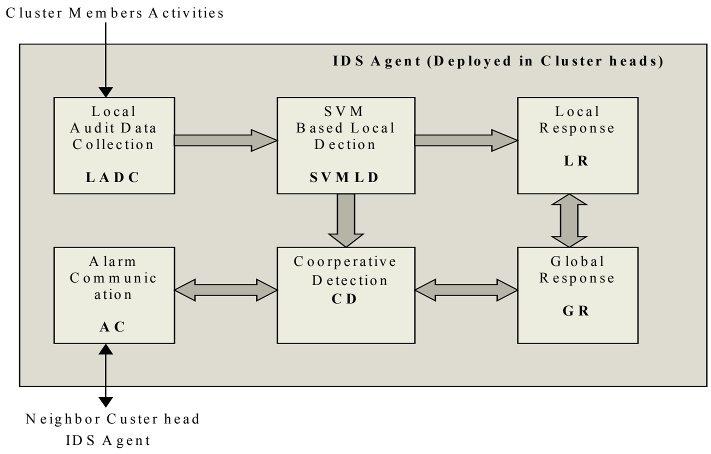

Cluster Members Activities
IDS Agent (Deployed in Cluster heads)
Local Audit Data Collection
LADC
SVM Based Local Dection
SVMLD
Local Response
LR
Alarm Communication
AC
Coorperative Detection
CD
Global Response
GR
Neighbor Custer head IDS Agent

Fig. 2. The IDS agent based on SVM

1) Local Audit Data Collection (LADC): Local Audit Data Collection module gathers streams of real time audit data from user and system activities within the mobile cluster, communication activities by cluster members as well as any communication activities within the radio range of this cluster.

2) SVM-Based Local Detection Engine (SVM-LDE):

The local intrusion detection procedure is divided into two stages:

Stage1: SVMs Training. we should train the SVMs by normal and intrusive data to get the attacks and normal usage patterns. The normal patterns are determined using the trace data from the training process where all activities are normal.

Stage2: Testing. We apply SVMs to the set of intrusion and normal data, and the data is classified by the SVM Classifiers according to the attacks and normal usage patterns, any deviations from the normal profiles are recorded if at all any occur. At last, the classified result is send to the decision-maker to juge. A detection module is to find all the deviation data to distinguish anomalies from normalcy.

3) Cooperative Detection: The cooperative intrusion detection procedure is divided into three stages: **1.** An IDS agent after analyzing its local data concludes that there is an intrusion in its own cluster, and initiate a local response. **2.** An IDS agent after analyzing the local data as well as that from its neighbors that there is an intrusion, and initiate a local response. **3.** Several agents collectively conclude that there is an intrusion, and initiate a globe response.

The Cluster IDS agent negotiation Protocol can include following steps:

Step1: When a cluster head IDS agent cannot decide a suspect action, it sends to its neighboring agent an “intrusion alarm request”.

Step2: Each agent propagates the state information, indicating the possibility of an intrusion to its direct neighbors.

Step3: Each agent then determines whether the majority of the received reports point towards an intrusion, if yes then it concludes that the network is under attack.

Step4: Any agent that detects an intrusion to the network can then initiate the response procedure.

It is a voting scheme to verdict the intrusion behavior. As we know, a compromised node might tend to send misleading data, but which would result in its expulsion from the network in this scheme. Even majority of nodes are compromised, and there exists at least one valid agent, the remedial procedure would be initiated yet, because the agent can decide the intrusion with evidence own. However for compromised node sending audit data doesn't hold any incentives, in doing so it might create a situation, which would result in its expulsion from the network.

4) Intrusion Response. When the abnormal activities are detected, the response initiated locally or globally. Reset the communication channels, identify the compromised nodes and preclude them from the network. The IDS agent can notify the end user to take the necessary action; sends re-authentication requests to all the nodes in the network, to prompt end users to authenticate themselves. Only the re-authenticated nodes participate in negotiating a new communication channel and will recognize each other as legitimate nodes. Thus the malicious nodes can be precluded.

3 Simulation

The ad hoc network is not a kind of practical technique yet, so our research is carry out in simulation environment. The SVM-based intrusion classifiers were implemented using the LIBSVM package[6]. Our simulation bed in ns-2 simulator[7], and the simulation environment description as follow:

Flat Size: 500 x500 meters. Number of Nodes: 50 (8 clusters).
MAC Layer: IEEE 802.11, peer to peer mode Simulation Time: 800 sec
Transport layer: User Datagram Protocol (UDP).
Routing Protocol: AODV Mobility: Random Waypoint
Traffic: 10 constant bit rate (CBR) flows that send 4 packets per second with a packet size of 512 bytes.
Interval time of testing the accuracy rate of intrusion detected: 100 sec.

Here, we focus on a few attacks that have a high severity of negative impact leading to the failure of the network: Black Hole Attack (BHA), Frequent False Routing Requesting (FFRR), Replay Attack.

The accuracy of intrusion detection result showed in Fig. 3 and Fig.4.

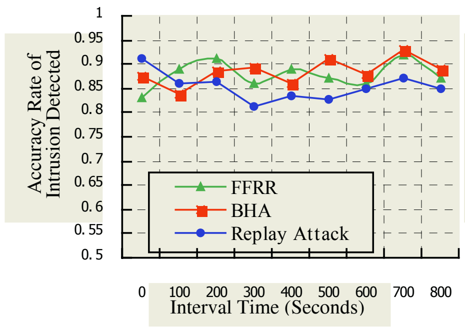


Fig. 3. Accuracy of Intrusion Detection

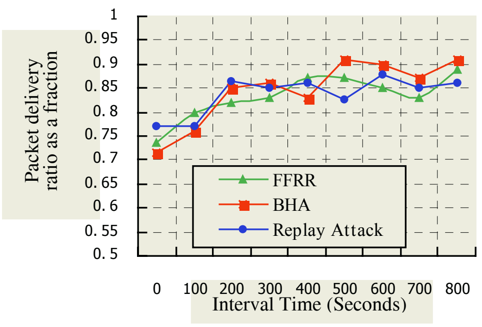


Fig. 4. Packet delivery ratio as intrusion detection

Preliminary results have shown that our IDS simulated in the network with an average accuracy of about 85% and a low degree of false alarms. The very few scenarios wherein intrusion is not detected may be attributed to the extremely high mobility, possible loss of connectivity and interference in the network. Incorporation of our IDS into the MANET does not affect the successful delivery of packets. The slight decrease in PDR values with IDS could be because of non-cooperating compromised nodes as packets to be routed through these nodes or even packets intended for these nodes are not delivered. The system performance is proved to be effective in above attacks situation by simulation.

4 Conclusions

The main challenge in design of the system is for ad hoc networks' vulnerability to security attacks. In this paper, we identify the new challenges and opportunities posed by this new networking environment and explore new IDS architecture to secure its communication. In particular, we take advantage of the inherent topology in ad hoc networks – hierarchical and distributed architecture; and multiple routes between nodes for each node is both host and router – to design the architecture of IDS to against abnormal attacks. We also integrated the new classify method, SVM-based classify method, to build a highly secure and more accurate intrusion detection system, which forms the core of the security framework's second defense line. It can be applied to routing attack and other type attack in other layers.

References

1. IETF: Mobile and Ad Hoc Networks Charter. http://www.ietf.org/html.charters/manet-charter.html (2004)
2. Zhou, L. and Haas, Z.J.: Securing Ad Hoc Networks. IEEE Network Magazine, **13** (1999) 24-30
3. Zhang, Y. and W.L.: Intrusion Detection in Wireless Ad Hoc Networks. In proceeding of the 6th International Conference on Mobile Computing and Networking (MobiCom 2000). Boston, Massachusetts, **8** (2000) 275–283
4. Vapnik, V.: Statistical Learning Theory, Springer. Berlin, Heidelberg New York (1998)
5. Andrew, H., Sung, S., Mukkamala: Identifying Important Features for Intrusion Detection Using Support Vector Machines and Neural Networks. In proceedings of the 2003 Symposium on Applications and the Internet (2003)
6. http://www.csie.ntu.edu.tw/~cjlin/libsvm
7. The Network Simulator – NS2. http://www.isi.edu/nsnam/ns/

A New Approach of Network Intrusion Detection Using HVDM-Based SOM

Lei Wang[1], Yong Yang[2], and Shixin Sun[1]

[1] School of Computer Science and Engineering,
University of Electronic Science and Technology of China, Chengdu, Sichuan 610054, China
dr.wangl@263.net, sxsun@uestc.edu.cn
[2] Suminet Communication Technology(Shanghai) Co., Ltd, Shanghai 200127, China
yangyong1014@vip.sina.com

Abstract. The research of applying Self-organizing Maps for intrusion detection is investigated in this paper. A novel approach is presented for enhancing SOM's abilities of identifying temporal network attacks, which combine with FIR filter. Meanwhile, we reconsider the heterogeneous dataset that composed of network connection's features, and select HVDM as the distance function determining the winning neuron during SOM's training and testing. In the end, KDD benchmark dataset is employed to validate the efficiency of our approach, and the results is detection rates of 96.5%, false positive rates of 6.2%, which accounts for good performance of our new approach in intrusion detection fields.

1 Introduction

Intrusion detection system (IDS) is regarded as the second secure lines after firewalls. Specifically, we define IDS as systems monitoring the status of network without degrading the performance of network infrastructure, whose main functions are auditing, tracing, recognizing, and detecting the unauthorized accesses, or abnormal activities and events. Recent years, network based intrusions take on a trend of uncertainty, complexity and diversity, which brings new challenges to IDS. Classical network based approaches of intrusion detection often rely on either rule-based approaches or anomaly detection. However, both of them have many pending problems to be settled, as [1].

Artificial intelligence (AI) methods had been introduced to design IDS, which are current research hotspot trying to solve the problems of classical approaches. These approaches often focus on identification and generalization of intrusion characters, and have great advantages over classical statistics based methods on detection precision, intelligence, and adaptability.

Self-Organizing Map (SOM) has been applied to IDS for its' powerful and efficient performance in classification problems [2]. Performances of three kinds of neural network with two different coding schemes are compared in [3], including BP, RBF and SOM. [4] studied the approach of using multi-layer SOM for identifying the patterns of intrusions, and proposed corresponding learning algorithm. [5] proposed a DSOM model to handle SOM's deficiency of identifying temporal network attacks, and trained the DSOM combing SOM and BP's learning algorithm.

Above approaches in [3],[4],[5] adopt Euclidean distance for the measure of similarity between input vector and representation vectors during SOM's training phase. However, Euclidean distance doesn't work well when handle heterogeneous data, some preprocessing must be done, as [6]. So, when treat the feature vectors associated with network connections, we need to transform them into forms of linear type, which Euclidean distance can handle directly. Unfortunately, some important information between vectors will lose too.

In this paper, we put forward a novel approach of enhancing SOM's abilities of identifying temporal network attacks, such as DOS and Probing. Meanwhile, we replace Euclidean distance by HVDM as distance metric to determine the winning neuron when training SOM. As far as I know, our approach is firstly proposed, and is expected to improve the performance of IDS.

2 HVDM-Based SOM

2.1 Enhance SOM's Abilities Using FIR Filter

Classical SOM is good at dealing with static patterns. However, it doesn't work very well when handle temporal network attacks, such as Dos and Probing, which are popular in Internet. When we use classical SOM to these scenarios, corresponding false positive rate and false negative rate will both be high. On the other hand, from the analyzing of characters of Dos, DDos and Probing, we can sure that the character of current network connection may be easily judged if we know the characters of all $L-1$ network connections before it. That is to say, we form a feature vector sequence using L feature vectors, and judge the character of current network connection from the sequence. Thus, the feature vector sequence containing temporal information of an attack can be handled by SOM directly.

FIR filter gives us some ideas of forming feature vector sequence. Specifically, we use filters handling original input vectors of SOM and get corresponding feature vector sequence. Supposing the network connection is represented by m-dimension feature vector, we can input each feature into a FIR filter whose order is L, and get a $m \times L$-dimension input vector for SOM actually. Figure 1 demonstrates this process.

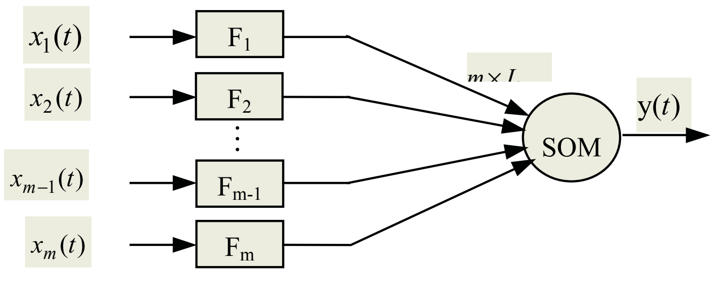


Fig. 1. Using FIR filter dealing with SOM's original input vector

2.2 Extend SOM on Heterogeneous Dataset

Definition 1. Heterogeneous dataset
If X is a m-dimension dataset, $\forall x \in X$, Let $x_i (i = 1,...m)$ be the i th attribute of x. Then X is a heterogeneous dataset if the following are true: 1) the type of $x_i (i = 1,...l)$ is linear. 2) the type of $x_i (i = l+1,...,m)$ is nominal.

KDD-99 dataset belongs to typical heterogeneous dataset (See Sect. 3). Learning algorithm of classical SOM doesn't work well on it for its nominal attributes, in which the measure of distance is based on Euclidean distance. In this paper, we adopt HVDM distance to handle heterogeneous data, which metric is firstly proposed in [7], thus learning algorithm of SOM can be directly extended to heterogeneous datasets.

Definition 2. Normalized linear distance
Let $\forall x, y \in X$, and x_a, y_a be their a th attribute whose type is linear. Then the distance of x, y on the a th attribute is defined as following:

$$normalized - diff_a(x, y) = \frac{|x - y|}{4\sigma_a} . \tag{1}$$

Where σ_a is the variance of X on the a th attribute.

Definition 3. Normalized VDM (value difference metric)
Let $\forall x, y \in X$, and x_a, y_a be their a th attribute whose type is nominal. Then the distance of x, y on the a th attribute is defined as the following:

$$normalized - vdm_a(x, y) = \sqrt{\sum_{c=1}^{C} \left| \frac{N_{a,x,c}}{N_{a,x}} - \frac{N_{a,y,c}}{N_{a,y}} \right|^2} . \tag{2}$$

Where $N_{a,x}$ is the number of instances in the X that have same value for attribute a as x; $N_{a,x,c}$ is the number of instances in X that have same value for attribute a as x and output class c; C is the number of output classes in the problem domain.

Definition 4. HVDM distance
Let $\forall x, y \in X$, then the distance between x and y under the HVDM metric is defined as the following:

$$H(x, y) = \sqrt{\sum_{a=1}^{m} d_a^2(x_a, y_a)} . \tag{3}$$

$$d_a(x, y) = \begin{cases} 1 & \text{If } x \text{ or } y \text{ is unknown;} \\ normalized - vdm_a(x, y) & \text{If } a \text{ is nominal;} \\ normalized - diff_a(x, y) & \text{If } a \text{ is linear.} \end{cases} \tag{4}$$

HVDM treats different types of attributes with different distance metrics when handles heterogeneous datasets, in the end, each attribute contributes fair effects on final distance between vectors. So, HVDM will get more preferable performances than Euclidean and HOEM metric when handles heterogeneous datasets, as [7].

2.3 Learning Algorithm of HVDM-Based SOM

Learning algorithm of SOM in our approach is as the following:

(1) Initialize the network weights $w_j(0)$, original learning rate $\eta(0)$, original neighborhood size $N(0)$;

(2) Present an input vector $x(t)=(x_1(t), x_2(t), \ldots x_{m\times L}(t))^T$ to SOM, which have been disposed by FIR filters;

(3) Calculate the distance between input vector $x(t)$ and each neuron weight $w_j(t)$, and the winning neuron i^* is identified as having the minimal HVDM value, where the HVDM distance is defined as equation (3);

(4) Adjust all weights in the neighborhood of the winning neuron i^*, or

$$w_j(t+1) = w_j(t) + \eta(t)\cdot\Phi(t)\cdot(x(t) - w_j(t))\,. \tag{5}$$

Where $\eta(t)$ is the learning rate at epoch t; $\Phi(t)$ is a suitable neighborhood function, in this case of a Gaussian nature;

(5) Repeat steps (2)-(4) until the convergence criteria are satisfied.

3 Experiments and Results

KDD-99 dataset is used for the validation of our approach, which DARPA provided for evaluating different AI approaches to intrusion detection problem. Each record of the dataset is a network connection represented by 41 features, which can fall into four categories: basic features, content features, time-based traffic features, host-based traffic features. Meanwhile, connections can be categorized as normal or abnormal. Further, abnormal connections have 38 types (24 in training dataset), fall into one of four categories: Dos, Probing, User to Root, and Remote to Local.

We refer related experimental results of [4], training the SOM with 6 basic features. Table 1 lists these basic features. The training of network consists of two phases, both uses “10% KDD” dataset, associated training parameters are listed as table 2.

Table 1. Basic features adopted in our experiments and their types

Feature	Duration	Protocol_type	Service	Flag	Src_byte	Dst_bytes
Type	linear	nominal	nominal	nominal	linear	linear

Table 2. Training parameters of SOM

$\eta(0)$	1.0	**Neuron Relation**	Hexagonal
$N(0)$	7	**Order of FIR Filter**	10
Training Number of First Phase	97,277 (normal)	**Training Number of Second Phase**	100,000 (85,012 is abnormal)

In first phase, only normal connections are used to train SOM, winning neuron for each input is recorded. Figure 2 shows the summary of winning neurons of this phase. It’s easily to know that more times a neuron wins, more likely it can represent normal network connections, otherwise, it may represent abnormal ones, namely may have relations with attacks. In second phase, both normal and abnormal connections are used to train SOM, however, only winning neuron for each abnormal connection is recorded this time. Figure 3 shows the summary of winning neurons of this phase. It's also easily to know that more times a neuron wins, more likely it can represent abnormal network connections, otherwise, it may represent normal ones. Just as we have expected, results of the two phases are consistent with each other, both show the facts that neuron 1,2,11,12,35,36 represent abnormal network connection.

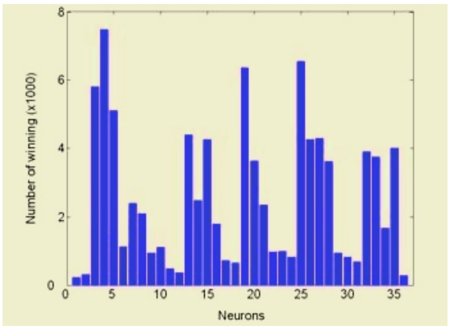


Fig. 2. Summary of the winning neurons in first phase

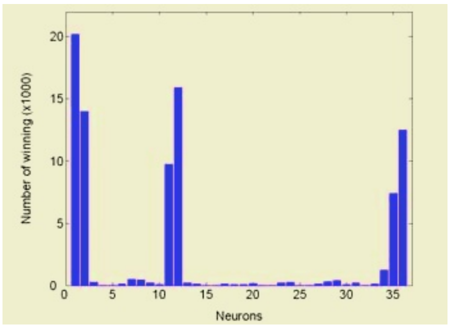


Fig. 3. Summary of the winning neurons in second phase

"Corrected (Test)" dataset is used for the validation of trained SOM. The detailed method is: for each vector extracted from the dataset, we input it to the trained network and determine which neuron wins; if the wined neuron's index is among 1,2,11,12,35,36, we judge the connection associated with the input are some kind attacks. Performance of SOM is evaluated in terms of the false positive rate (FP) and false negative rate (FN) in experiments. Final results we calculated in our experiments are FP of 3.5% and FN of 6.2%. Table 3 shows the statistics of results in our testing for connections containing different kind of attacks.

Table 3. Comparison of the two networks' performance for detection network attacks

Type of Connections		**Normal**	**Dos**	**Probing**	**L2R**	**R2L**
Total Number		18411	77056	4263	9	261
SOM + FIR + HVDM	Failed	1140	2527	250	6	118
	Successful	17271	74529	4013	3	143
	Error rate	6.19%	3.28%	5.86%	66.67%	45.21%
	Sum	6.2%	3.5%			
SOM + Euclidean	Failed	2033	9296	686	6	146
	Successful	16378	67760	3577	3	115
	Error rate	11.04%	12.06%	16.09%	66.67%	55.93%
	Sum	11.0%	12.4%			

To illustrate the efficiency of our approach in enhancing SOM's abilities of identifying temporal network attacks, we also train a SOM with classical learning algorithm (without using FIR filter and HVDM distance metric), and test its' performance for detection network attacks. Comparison of the two networks' performance is listed as table 3. Results show the superiority of our approach, that is to say, our approach can effectively enhance SOM's abilities of identifying temporal network attacks, with very low FP rate and FN rate.

4 Conclusions

In this paper, a novel approach for network intrusion detection is proposed which combines FIR filter and classical SOM to enhance SOM's abilities of identifying temporal network attacks. HVDM distance metric is also used in our approach to determine the winning neuron, thus settles SOM's deficiency on heterogeneous data. Our approach is implemented and test on KDD-99 dataset. Experiment results show that the approach works very well on temporal network attack, such as Dos and Probing, with very low FP rate and FN rate.

Further research is in progress to optimize some parameters of SOM, such as number of neurons, order of filter, learning algorithm, so that lower FP rate and FN rate will be achieved.

References

1. Allen, J., Christie, A., Fithen, W., etc.: State of the Practice of Intrusion Detection Technologies. CMU/SEI-99-TR-028 (2000) 48-58
2. Kohonen, T.: The Self-Organizing Map. Proceedings of the IEEE, **78** (1990) 1464-1480
3. Liu, Z., Florez, G., Bridges, S.M.: A Comparison of Input Representations in Neural Networks: A Case Study in Intrusion Detection. IEEE Proceedings of International Joint Conference on Neural Networks, **2** (2002) 12-17
4. Kayacik, H.G., Zinci-Heywood, A.N., etc.: On the Capability of an SOM based Intrusion Detection System. IEEE Proceedings of the International Joint Conference on Neural Networks, **3** (2003) 20-24
5. Liu, Q., Sylvian, R., et al.: Temporal Sequence Learning and Recognition with Dynamic SOM. IEEE International Joint Conference on Neural Networks, **5** (1999) 10-16
6. Stanfill, C., David, W.: Toward Memory-based Reasoning. Communications of the ACM, **29** (1986) 1213-1228
7. Wilson, D.R., Martinez, T.R.: Improved Heterogeneous Distance Functions. Journal of Artificial Intelligence Research, **6** (1997) 1-34

A Novel Approach to Corona Monitoring*

Chiman Kwan[1], Tao Qian[1], Zhubing Ren[1], Hongda Chen[1], Roger Xu[1], Weijen Lee[2], Hemiao Zhang[2], and Joseph Sheeley[3]

[1] Intelligent Automation, Inc., 15400 Calhoun Drive, Suite 400, Rockville, MD 20855
{hgxu,tqian,ckwan}@i-a-i.com
[2] Electrical Engineering Dept, University of Texas at Arlington,
416 Yates Street, Nedderman Hall, Rm 501, Arlington, TX 76010
lee@exchange.uta.edu
[3] Arnold Air Force Base, Tennessee, TN 37389-9013
Joseph.Sheeley@arnold.af.mil

Abstract. Corona discharge (CD) and partial discharge (PD) indicate early stages of insulation problems in motors. Early detection of CD/PD will enable better coordination and planning of resources such as maintenance personnel, ordering of parts, etc. Most importantly, one can prevent catastrophic failures during normal operations. In this paper, we summarize the application of Support Vector Machine (SVM) to CD/PD monitoring. Hardware testbeds have been developed to emulate CD/PD behaviors and real-time experimental results showed the effectiveness of SVM for fault detection and classification.

1 Introduction

Partial discharges (PD) are both a cause and a symptom of many types of insulation deterioration mechanisms in motors and generators. In decades, on-line PD measurement has been used to find loose, delaminated, overheated, and contaminated defects before these problems lead to failures. As a result, on-line PD monitoring has become an important tool for planning machine maintenance. Many methods are available to measure the PD activity in operating machines. The electrical techniques all rely on monitoring the current and/or voltage pulse created whenever a partial charge occurs. In the literature, it is normal to measure the currents by means of a high frequency current transformer at neutral points [1], or to detect the PD pulses via high voltage capacitors connected to the phase terminals. The capacitor size ranges from 80pF to 1000pF. Those methods are generally expensive due to considerations of the high frequency characteristics, where higher sampling rates are needed to handle them.

In this research, we proposed a novel approach for PD monitoring. Instead of using high frequency analysis, our method is based on the low frequency characteristics of the PD events so that it can be cost-effective in practical applications. Our approach requires less than 50 k sampling and costs less than 3,000 dollars.

In this research, we have successfully achieved the above objectives. There are several key results:

* Research Supported by Air Force under Contract Number F40600-03-M-0008

- Fault classification algorithms
  After the signature extraction process, the signatures are still weak, especially during the CD/PD stage. Effective classification algorithm is absolutely necessary to make the correct classification between normal, CD/PD, and arc faults. We have implemented four classifiers: 1) PCA (Principal Component Analysis)/PSD (Power Spectral Density); 2) GMM (Gaussian Mixture Model); 3) SVM (Support Vector Machine); 4) MLP-NN (Multilayer Perceptron Neural Net). Among these, SVM and MLP-NN achieved the best performance in terms of less false alarms and wrong classifications.
- Testbed development
  In this small feasibility study project, it is too expensive to produce motors with different insulation problems. Here we designed and evaluated two CD/PD and arc producing devices. One is Jacob's ladder and the other one is an arc device with a small spinning motor. The two devices have similar characteristics in CD/PD and arcs. Then we attached the device to a relatively big motor and use the device to emulate CD/PD and arc behavior.
- Real-time experiments
  The fault classification algorithms were implemented in Labview and real-time experiments were successfully performed. The SVM and MLP-NN performed very well in distinguishing between normal, CD, and arc conditions.

This paper is organized as follows. Due to page limitations, here we only focus on results related to SVM. In Section 2, the system architecture and the algorithm will be reviewed. In Section 3, we will describe the arc producing devices, which emulates the CD/PD and arc behavior in a motor. In Section 4, we will first describe the testbed, including the sensors, the data acquisition card, etc. Then we will describe the Labview program. Finally, we will summarize the performance of different classification algorithms. Conclusions will be described in Section 6.

2 CD/PD Classifier Using Support Vector Machine

A good classifier is needed to perform fault classifications. After spending a tremendous amount of time on literature survey and comparing different classifiers, we decide to utilize Support Vector Machine classifiers to perform fault classifications in this study. The advantages of SVM include:

- It is a quadratic learning algorithm; hence, there are no local optima. It can also be formed as linear programming for simplicity.
- Statistical theory gives bounds on the expected performance of a support machine.
- Performance is better than most learning systems for a wide range of applications including automatic target recognition, image detection, and document classification
- Although originally designed for 2-class classification, SVMs have been effectively extended to multiclass classification applications.
- There is no over-training problem as compared to conventional learning classifiers such as neural net or fuzzy logic.

According to references [2], [3], two key elements in the implementation of SVM are the techniques of mathematical programming and kernel functions. The parameters are found by solving a quadratic programming problem with linear equality and inequality constraints; rather than by solving a non-convex, unconstrained optimization problem. The flexibility of kernel functions allows the SVM to search a wide variety of hypothesis spaces. The geometrical interpretation of support vector classification (SVC) is that the algorithm searches for the optimal separating surface, i.e. the hyperplane that is, in a sense, equidistant from the two classes.

The SVM can feed the multi-dimensional input signal to the classifier, so there is no loss of information in the training and testing. The generalization performance and the classification capability of the SVM can easily be controlled by the kernel function and the regularization. The SVM is also faster than GMM in the training and testing. The simulation results showed that SVM performance is better than that of GMM. Fig. 1 shows the overall architecture of the SVM approach. Again Hall effect sensors are used.

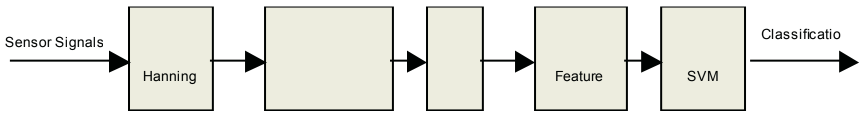


Fig. 1. SVM classification diagram

3 Arc Producing Devices: Jacob's Ladder and Spinning Motor

Due to cost concerns, it was decided not to produce various insulation failures in a real motor. So we adopt an alternative approach, which is to develop a CD/arc producing device to emulate the fault signatures in motors. We have developed two devices: one is the Jacob's ladder and the other one is an arc producing device with a small spinning motor.

First, we verified the similarities between motor arcs and the arcs in Jacob's ladder. The Jacob's ladder has similar arc features as compared to the motor arc faults. The difference is that the arc in the Jacob's ladder restarts periodically and the length of the arc also increases continuously. The motor arc contains background operation current and some harmonics. However, this difference does not affect the common arcing characteristics. Therefore, if an algorithm can detect the common arc patterns, it should be able to detect the arc in the motor. Comparing with high frequency signals, which are easy to get interferences, we are working on the low frequency domain. Moreover, we use the Hall effect sensor with a response bandwidth 50 kHz, which can be sure to catch the current information of the actual arc current of a motor. Second, we have been able to collect synchronized data by using four sensors simultaneously: one antenna sensor, two Hall-effect sensors at low and high voltage side each, and one acoustic sensor. Based on our evaluations, the acoustic sensor is not very effective in detecting the PD. However, Hall Effect sensor and antenna work quite well.

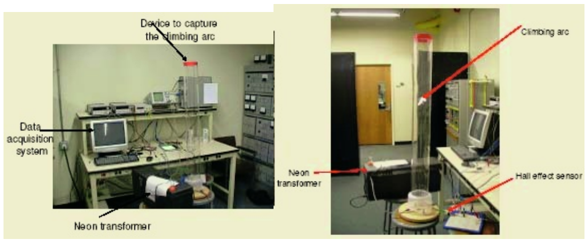


Fig. 2. Jacob Ladder for generating the arcing fault

Fig. 3. Hall effect sensor

3.1 Jacob's Ladder

Partial discharge is very similar to arc faults. To get some insights into the partial discharge, it is therefore very important to analyze the behavior of the arc faults. In this research, we set up a Jacob ladder as shown in Fig. 2. A neon transformer with 50 kV supply voltage and 1 kVA power is used to generate arcs. In-line current with embedded arcing components information is fed through a Hall effect sensor as shown in Fig. 3. Labview with Data Acquisition Card (DAQ) is used to digitize and record the information. The DAQ is able to simultaneously collect 14 channels of data.

3.2 Arc Producing Device with a Small Spinning Motor

Here we developed an alternative device to simulate the arc inside of a motor. We used a stick with a coil, and then it was connected to a motor's shaft and the coil's ends are opposite to each other. Referring to Fig. 4, we put the stick in the middle of two wires' ends. As the voltage of the two wires increases to about 1 kV, arc occurs between the end of the wire and coil. This arcing is discontinuous when the motor is on because the coil is rotating. To emulate corona, we can increase the voltage to some values less than 1 kV before the arc starts.

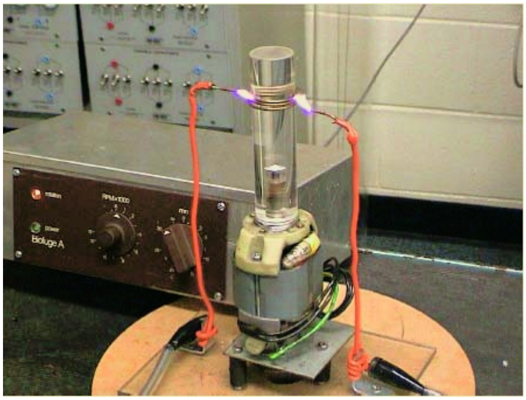


Fig. 4. A second testbed to emulate arcs

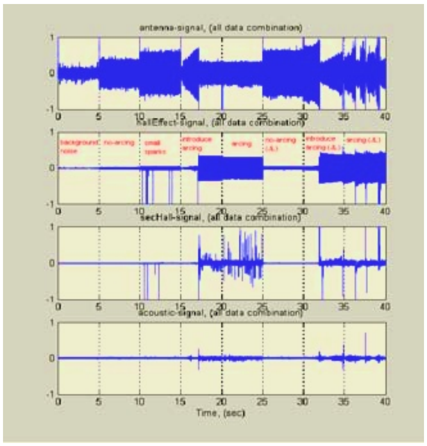


Fig. 5. Signals from different sensors

3.3 Evaluation of the Two Arc Producing Approaches

To evaluate the two approaches of emulating arcs, we consider a whole process of synchronized data with background noise, corona, and arc faults.

Fig. 5 shows the data segments. Each segment is with a period of 5 seconds. The synchronized four-sensor signals include 8 different patterns: (1) no power supply, background noise only; (2) constant transformer voltage without arcing faults; (3) higher transformer voltage to introduce some small discontinuous sparks; (4) an arc fault process, (voltage is from zero to the value where arcs are produced and keeps arcing); (5) arc faults; (6) constant transformer voltage without arc faults via Jacob's ladder; (7) an arc fault process via Jacob's ladder, (voltage is from some value to a value where arcs occur and keeps arcing); (8) arc faults via Jacob's ladder.

It can be seen that both devices generate sensor signals that are similar. Hence both devices are suitable for emulating CD and arcs.

4 Performance of SVM for Fault Classification: Off-Line Studies

We used the same data sets collected on Jan. 3, 2004 to train and test the Support Vector machine (SVM). The results are shown in Fig. 6. There is almost no false alarm in both the training data and testing data.

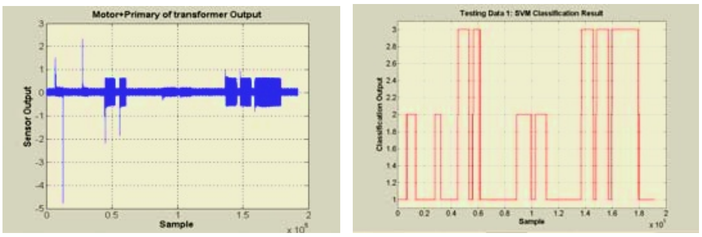


Fig. 6. The testing data 1 and SVM testing results

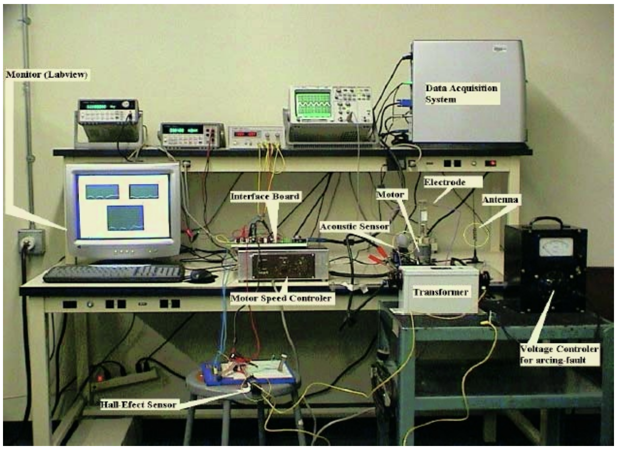


Fig. 7. System setup for partial discharge/arc monitoring

5 Real-Time Experimental Results

5.1 Experimental Setup

The purpose of the experimental setup is to simulate the partial discharge and arc behavior in a motor. The partial discharges and arcs create changes in currents, acoustic emissions, and electromagnetic interference.

Fig. 7 shows the experimental setup. A small motor with a rpm range from 0 to several thousand is used. Two electrodes are used to generate arcs. A voltage supply (0 to 130 volt) is used to provide power to the electrodes. A transformer amplifies the voltage by 100 times is used to amplify the voltage signals. Three sensors are used to collect the partial discharge and arc signatures: a microphone, a Hall effect sensor, and an antenna. The sensors are connected to a data acquisition card in a PC. Labview algorithms are used to directly process the sensor signals and generate fault detection indices, which are shown on the monitor.

5.2 Labview Program

The detection and classification algorithms have been explained in earlier sections. Here we only briefly describe the Labview program.

Fig. 8 shows the overall flow chart of the Labview program. The sensor signals are collected by the data acquisition card and then processed by various algorithms. The processed results are then displayed on the monitor.

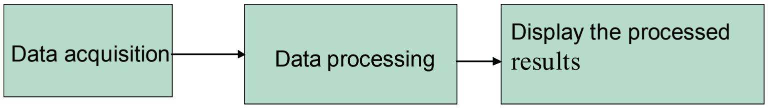
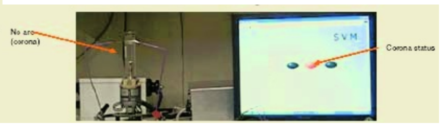
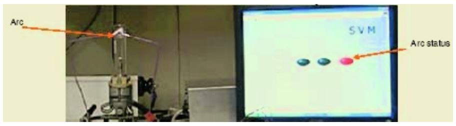


Fig. 8. Real-time PD detection flow chart

(a) System is correctly classified as normal.



(b) System is correctly classified as corona.



(c) System is correctly classified as arc.

Fig. 9. Performance of SVM in real-time

5.3 Performance of Support Vector Machine for Real-Time Monitoring of CD/PD

The SVM classifier was implemented in Labview and real-time monitoring was carried out. Compared to GMM and MLP-NN methods [4], the false alarms and incorrect classifications have been significantly reduced. The overall classification is very robust even during the switching transients. A video was recorded. Here we include a few images captured from and video and show them in Fig. 9.

6 Conclusions

In this research, we have met all the research objectives. A realistic testbed has been designed and built to emulate CD/PD and arc signatures in a motor. Many data sets from normal, corona, and arc conditions have been collected. Various preprocessing algorithms, feature extraction algorithms, and fault classifiers have been customized, implemented, and evaluated. Extensive simulations and experiments were successfully carried out to demonstrate the performance of our fault detection and classification algorithms.

References

1. Stone, G.C., Warren, V., Sedding, H.G., McDermid, W.: Advances in Interpreting Partial Discharge Test Results from Motor and Generator Stator Windings. Iris Power Engineering, Technical Paper (2000)
2. Hsu, C.-W., Lin, C.-J.: A Comparison of Methods for Multiclass Support Vector Machines. IEEE Transaction on Neural Networks, **13** (2002) 415-425
3. Burbidge, R., Buxton, B.: An Introduction to Support Vector Machines for Data Mining. Keynote Papers, Young OR12, University College London (2001)
4. Kwan, C., Qian, T., Ren, Z., Chen, H., Xu, R., Lee, W., Zhang, H.: A Novel Approach to Corona Monitoring, Phase 1 Final Report. March (2004)

Multi-class Probability SVM Fusion Using Fuzzy Integral for Fault Diagnosis*

Zhonghui Hu, Yunze Cai, Xing He, Ye Li, and Xiaoming Xu

Department of Automation, Shanghai Jiaotong University,
Shanghai 200030, China
ZhonghuiHu@msn.com

Abstract. A multi-class probability support vector machine (MPSVM) is designed by training the sigmoid function to mapping the outputs of standard SVMs into posterior probabilities and then combining these learned probability SVMs. The method of using fuzzy integral to combine multiple MPSVMs is proposed to deal with distributed multi-source multi-class problems. The proposed method considers both the evidence provided by each MPSVM and the empirical degree of importance of these MPSVMs in combination process. It is applied to fault diagnosis for a diesel engine. The experimental results show the performance of fault diagnosis can be improved.

1 Introduction

It is in great demand to understand what and how mechanical failure occurs for decreasing the costs of production and maintenance [1],[2]. The fault data are hard to sample and the fault mechanism is complex. The support vector machine (SVM) has good generalization performance [3],[4]. It is a promising method for fault diagnosis.

It is valuable to extend the SVM for binary classification to solve the multi-class problem. Two types of approaches for constructing multi-class SVM (MSVM) have been proposed [5]. One is by constructing and combining several binary classifiers while the other is by directly considering all data in one optimization formulation. Platt [6] described a method for fitting a sigmoid that maps binary-class SVM outputs to posterior probabilities. This method is called as probability SVM (PSVM). We extend the one-against-all MSVM to multi-class PSVM (MPSVM) in this paper.

The methods of combining evidences produced by multiple information sources have been profoundly researched [7],[8]. The fuzzy integral and the associated fuzzy measures are initially introduced by Sugeno [9]. It is first used as a new evidence fusion technique in [10] for information fusion in computer vision, and is also used for multiple neural network fusion in [11],[12],[13]. The fuzzy integral differs from some other theories in that both objective evidence supplied by various sources and the expected worth of subsets of these sources are considered in the fusion process

* This work was supported by the National Key Fundamental Research Program of China (2002cb312200), the National High Technology Research and Development Program of China (2002AA412010), and in part supported by the Natural Science Foundation of China (60174038).

[10],[11]. In this paper, we propose a method that multiple MPSVMs are aggregated by using the Sugeno fuzzy integral, and apply it to fault diagnosis for a diesel engine.

This paper is organized as follows. Subsection 2.1 introduces the PSVM. Subsection 2.2 describes the proposed MPSVM. Section 3 introduces the fuzzy measure and fuzzy integral. In Section 4, we develop the method of combining MPSVMs using the fuzzy integral and illustrate its use. In Section 5, the method proposed is applied to fault diagnosis. We conclude with a discussion in Section 6.

2 Multi-class Probability Support Vector Machines

2.1 Probability Support Vector Machines for Binary Classification

Let the continuous output of a standard SVM, the distance from the unknown example x to the optimal classification hyperplane, be f [6]. A parametric model is used to fit the posterior $P(y=1\,|\,f)$ directly, i.e.,

$$P(y=1\,|\,f)=\frac{1}{1+\exp(Af+B)}\,. \tag{1}$$

where the parameters A and B are adapted to give the best probability outputs. The two parameters are found by minimizing the following cross-entropy error function

$$\min\ -\sum_{\mathrm{i}} \mathrm{t_i}\log(p_i)+(1-t_i)\log(1-p_i)\,. \tag{2}$$

where

$$t_i=\frac{y_i+1}{2}\,. \tag{3}$$

The problem (2) is performed by using a model-trust minimization algorithm [14].

A simple out-of-sample model is used in [6] for preventing overfitting in training sigmoid. Suppose N_+ positive examples and N_- negative examples are observed. The maximum a posteriori probabilities (MAPs) for the target probability of positive examples, and for that of negative examples, are, respectively,

$$t_+=\frac{N_+ +1}{N_+ +2}\,,\ t_-=\frac{1}{N_- +2}\,. \tag{4}$$

Hence, the training set for sigmoid fit is

$$(f_i,t'_i)\,,\ t'_i=\begin{cases} t_+, t_i=1 \\ t_-, t_i=0 \end{cases}, i=1,2,\cdots,l \tag{5}$$

Thus, we can obtain the final decision function of PSVM

$$d(x)=\begin{cases} 1, p(x)\ge 0.5 \\ -1, p(x)<0.5 \end{cases}. \tag{6}$$

2.2 Multi-class Probability Support Vector Machines

Given a training data set $\left\{(x_i, y_i)\right\}_{i=1}^{l}$, where $x_i \in R^n$ and $y_i \in \left\{1, \cdots, K\right\}$. The MPSVM can be constructed by applying the following procedure [4, 6, 15].

1) Construct K binary-class SVM classifiers where $f_k(x)$ ($k = 1, 2, \cdots, K$) separates the training examples of class k from all the other training examples.
2) Training the corresponding sigmoid function using the training set (f_i, t'_i), $i = 1, 2, \cdots, l$, K PSVM classifiers with outputs $p_k(x)$, $k = 1, 2, \cdots, K$ are constructed.
3) Construct the K-class MPSVM classifier by choosing the class corresponding to the maximal probability among $p_k(x)$, $k = 1, 2, \cdots, K$. The final decision function is

$$d(x) = \arg\max\{p_1(x), p_2(x), \cdots, p_K(x)\}\,. \tag{7}$$

3 Fuzzy Measure and Fuzzy Integral

The fuzzy integral and associated fuzzy measures are introduced as below [9, 10, 16].

Definition 1: Let X be a finite set of elements, and B be a Borel field of X. A set function g: $2^X \to [0,1]$ defined on B is called a fuzzy measure if it satisfies

1) Lower and upper boundary condition: $g(\varnothing) = 0$ and $g(X) = 1$;

2) Monotonicity condition: $g(A) \le g(B)$, *if* $A \subset B$, and $A, B \in \mathrm{B}$.

Sugeno [9] proposed a g_λ-fuzzy measure satisfying the λ-rule ($\lambda > -1$, and $\lambda \ne 0$).

$$g(A \cup B) = g(A) + g(B) + \lambda g(A) g(B)\,,\ A, B \subset X \text{ and } A \cap B = \varnothing\,. \tag{8}$$

Definition 2: Let $h : X \to [0,1]$ be a fuzzy subset of X. The fuzzy integral is

$$\int_A h(x) \circ g(\cdot) = \max_{E \subseteq X}[\min(\min_{x \in E} h(x), g(A \cap E))] = \max_{\alpha \in [0,1]}[\min(\alpha, g(A \cap h_\alpha))]\,. \tag{9}$$

where $h_\alpha = \{x \mid h(x) \ge \alpha\}$.

Let $h(y_1) \ge \cdots \ge h(y_n)$ (if not, reorder X). The fuzzy integral, e, is calculated by

$$e = \max_{i=1}^{n}[\min(h(x_i),\ g(A_i))]\,. \tag{10}$$

where $A_i = \{x_1, x_2, \cdots, x_i\}$, and g is a fuzzy measure over X.

For a g_λ-fuzzy measure, the values of $g(A_i)$ can be determined recursively as

$$g(A_1) = g(\{y_1\}) = g^1\,. \tag{11}$$

$$g(A_i) = g^i + g(A_{i-1}) + \lambda g^i g(A_{i-1})\,, \text{ for } 1 < i \le n\,. \tag{12}$$

For a fixed set of $\{g^i\}$, $0 < g^i < 1$, the constant value of λ can be obtained by

$$\lambda + 1 = \prod_{i=1}^{n} (1 + \lambda g^i) . \tag{13}$$

4 Combination of Multiple MPSVMs Using the Fuzzy Integral

Suppose there are N data sources and exist exhaustive K patterns (hypotheses). N MPSVM classifiers can be constructed corresponding to the N data sources. Let $C^* = \{C_1, C_2, \cdots, C_N\}$ be a set of MPSVMs, and $\Omega = \{\omega_1, \omega_2, \cdots, \omega_K\}$ be a class set. Set the probability outputs of the ith MPSVM are $\{p_{i1}, p_{i2}, \cdots, p_{iK}\}$, $i \in \{1, \cdots, N\}$. $p_{ij} : C^* \to [0,1]$ is considered as the partial evaluation of object x for class ω_j [11].

The fuzzy density can be given subjectively by an expert, or objectively by using training data [17, 18]. Two simple methods are described as follows [10],[11],[16],[19].

In the first method, the performance of every MPSVM is under consideration. The degree of importance of the ith MPSVM is determined by

$$g^i = \frac{r_i}{\sum_{m=1}^{N} r_m} d_{sum} . \tag{14}$$

where r_i is the accuracy of the ith MPSVM, r_m is the performance of the mth MPSVM, N is the number of MPSVMs, and d_{sum} is the desired sum.

In the second method, the performance of every MPSVM for each class is under consideration. The fuzzy density of a MPSVM for each class is given by

$$g^{ij} = \frac{r_{ij}}{\sum_{i=1}^{N} r_{ij}} d_{sum} . \tag{15}$$

where g^{ij} is the fuzzy density of the ith MPSVM in the recognition of class ω_j, r_{ij} is the performance of the ith MPSVM classifier for the jth class, and d_{sum} is the desired sum of fuzzy densities.

The proposed method is illustrated in Fig. 1, and also by the following example.

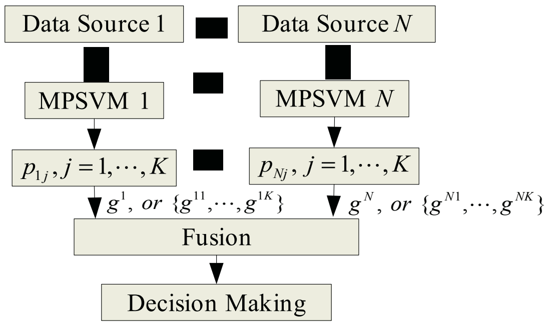

Data Source 1
Data Source N
MPSVM 1
MPSVM N
$p_{1j}, j = 1, \cdots, K$
$p_{Nj}, j = 1, \cdots, K$
g^1, or $\{g^{11}, \cdots, g^{1K}\}$
g^N, or $\{g^{N1}, \cdots, g^{NK}\}$
Fusion
Decision Making

Fig. 1. Multiple MPSVM fusion using the fuzzy integral

Example: Given a three-class classification problem and there exist three data sources. Three MPSVM classifiers corresponding to the three sources are constructed, respectively. The outputs of them for an unknown object x are listed in Table 1.

Table 2 shows the unique root and the degree of importance. The degree of importance of each MPSVM is in the last row, and that of each sub-PSVM is in other rows. With the first method of determining the degree of importance, the results of combination are given in Table 3, which shows the object belongs to class 3.

Table 1. The outputs of MPSVMs for an instance with unknown class label

p_{ij}	MPSVM 1	MPSVM 2	MPSVM 3
Sub-PSVM 1	0.5	0.7	0.2
Sub-PSVM 2	0.9	0.5	0.4
Sub-PSVM 3	0.4	0.6	0.8

Table 2. The unique root of fuzzy measure and the degree of importance of MPSVMs

g_i	MPSVM 1	MPSVM 2	MPSVM 3	Root(λ)
PSVM 1	0.3330	0.3134	0.3526	0.003
PSVM 2	0.3670	0.3058	0.3262	0.003
PSVM 3	0.2914	0.3330	0.3746	0.003
MPSVM	0.3317	0.3155	0.3519	0.003

Table 3. The combination results with the degree of importance of each MPSVM

Class	p_{ij}	$g(A_i)$	$H(E)$	max[$H(E)$]
1	0.7	$g(\{\omega_2\}) = g^2 = 0.3155$	0.3155	0.5
	0.5	$g(\{\omega_2, \omega_1\}) = g^2 + g^1 + \lambda_1 g^2 g^1 = 0.6474$	0.5	
	0.2	$g(\{\omega_1, \omega_2, \omega_3\}) = 1.0$	0.2	
2	0.9	$g(\{\omega_1\}) = g^1 = 0.3317$	0.3317	0.5
	0.5	$g(\{\omega_2, \omega_1\}) = g^2 + g^1 + \lambda_2 g^2 g^1 = 0.6474$	0.5	
	0.4	$g(\{\omega_1, \omega_2, \omega_3\}) = 1.0$	0.4	
3	0.8	$g(\{\omega_3\}) = g^3 = 0.3519$	0.3519	0.6
	0.6	$g(\{\omega_3, \omega_2\}) = g^3 + g^2 + \lambda_3 g^3 g^2 = 0.6677$	0.6	
	0.4	$g(\{\omega_1, \omega_2, \omega_3\}) = 1.0$	0.4	

5 Experimental Results

In [1], the rough set theory is used to diagnose the valve fault for a multi-cylinder diesel engine. Three sampling points are selected for collecting vibration signals. The whole dataset provided in [2] consists of 37 instances. Four states are researched [1]: Normal state; intake valve clearance is too small; intake valve clearance is too large; exhaust valve clearance is too large. At each sampling point, six features of the vibration signals are extracted from both the frequency domain and time domain. Thus, each instance in the dataset is composed of 18 condition attributes and one class attribute (four states). In the distributed schemes, the six features from one sampling point, added the class attribute, form an individual dataset. Therefore, three individual datasets corresponding to the three data sources are constructed.

The cross-validation test is used for showing the performance of rough set theory in fault diagnosis in [1]. The excerpted classification accuracy is listed in Table 4.

In our experiment, 25 instances of the dataset are used as training set and the rest are used as testing set. Choice of the kernel and parameters is via the performance on a validation set. Twenty percent of the training set is used as a validation set.

Table 4. Classification accuracy of each part (TrD: Training data; TD: Testing data)

Data set	1st part—TrD; 2nd part—TD	2nd part—TrD; 1st part—TD
Accuracy (%)	78.95	73.68

The whole experiment is repeated 50 times, where the training set and testing set are randomly selected without replacement every time. The accuracy of the three MPSVMs corresponding to three sampling points is given in the DS1, DS2 and DS3 columns (Table 5). The performance of the proposed method is listed in FM I and II. In FM I method, the degree of importance of each MPSVM is used, whereas in FM II method the degree of importance of every sub-PSVM is used. Table 5 shows that our proposed method improves the accuracy and robustness of fault diagnosis.

Table 5. Comparison of the experimental results (%) (DS1, DS2, DS3: Data source 1, 2, and 3; MV: Majority vote; FM: Fuzzy measure)

Scheme	Data source (MPSVM)			MV for MSVM	MV for MPSVM	Our method	
	DS1	DS2	DS3			FM I	FM II
Average accuracy	90.83	82.00	91.50	93.67	94.33	95.33	95.50
Minimal accuracy	66.67	58.33	75.00	75.00	75.00	75.00	75.00
RMS error	8.95	11.83	5.45	7.63	7.23	6.11	6.12

6 Conclusions

Based on PSVM, the MPSVM classifier using one-against-all strategy is proposed. To deal with distributed multi-source multi-class problem, the fuzzy integral is used to combine multiple MPSVM classifiers constructed corresponding to the information sources. Two methods of determining the degree of importance by using the recognition rate are applied. One method is using the recognition rate of each MPSVM, and another is using the recognition rate of every MPSVM for each class.

The method proposed is applied to fault diagnosis for a diesel engine. The experimental results show that the performance of fault diagnosis is improved.

References

1. Tay, F.E.H., Shen, L.: Fault Diagnosis Based on Rough Set Theory. Engineering Application of Artificial Intelligence, **16** (2003) 39-43

2. Shen, L., Tay, F.E.H., Qu, L., Shen, Y.: Fault Diagnosis Using Rough Sets Theory. Computers in Industry, **43** (2000) 61-72
3. Vapnik, V.N.: The Nature of Statistical Learning Theory. Springer Verlag, New York (1995)
4. Vapnik, V.N.: Statistical Learning Theory. Wiley, New York (1998)
5. Hsu, C.-W., Lin, C.-J.: A Comparison of Methods for Multi-class Support Vector Machines. IEEE Transactions on Neural Networks, 13 (2002) 415-425
6. Platt, J.C.: Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods. In: Smola, A.J., Bartlett, P., Scholkopf, B., Schuur-mans, D. (eds.): Advances in Large Margin Classifiers. MIT Press (1999)
7. Guan, J.W., Bell, D.A.: Evidence Theory and its applications, North-Holland-Amsterdam, New York, Vol. 1 (1992)
8. Cho, S.-B.: Pattern Recognition With Neural Networks Combined by Genetic Algorithm. Fuzzy Sets and Systems, **103** (1999) 339-347
9. Sugeno, M.: Fuzzy Measures and Fuzzy Integrals: a Survey. Fuzzy Automata and Decision Processes. North Holland, Amsterdam (1977) 89-102
10. Tahani, H., Keller, J.M.: Information Fusion in Computer Version Using the Fuzzy Integral. IEEE Transaction on System, Man and Cybernetics, **20** (1990) 733-741
11. Cho, S.-B., Kim, J.H.: Combining Multiple Neural Networks by Fuzzy Integral for Robust Classification. IEEE Transactions on systems, man and cybernetics, **25** (1995) 380-384
12. Cho, S.-B., Kim, J.H.: Multiple Network Fusion Using Fuzzy Logic. IEEE Transactions on Neural Networks, **6** (1995) 497-501.
13. Kumar, A.S., Basu, S.K., Majumdar, K.L.: Robust Classification of Multispectral Data Using Multiple Neural Networks and Fuzzy Integral. IEEE Transactions on Geoscience and Remote Sensing, **35** (1997) 787-790
14. Gill, P.E., Murray, W., Wright, M.H.: Practical Optimization. Academic Press (1981)
15. Platt, J.C., Cristianini, N., Shawe-Taylor, J.: Large Margin DAG's for Multiclass Classification. In: Advances in Neural Information Processing Systems 12. MIT Press, Cambridge, MA (2000) 547-553
16. Pham, T.D.: Combination of Multiple Classifiers using Adaptive Fuzzy Integral. Proceedings of the ICAIS'02
17. Keller, J.M., Gader, P.D., Hocaoğlu, A.K.: Fuzzy Integrals in Image Processing and Recognition. In: Grabisch, M., Murofushi, T., Sugeno, M. (eds.): Fuzzy Measures and Integrals (Theory and Applications). Physica-Verlag Heidelberg, New York (2000) 435-466
18. Keller, J.M., Osborn, J.: Training the Fuzzy Integral. International Journal of Approximate Reasoning, **15** (1996) 1-24
19. Yao, M., He, T., Pan, X., Zhang, R.: A Improved Combination Method of Multiple Classifiers based on Fuzzy Integrals. Proceedings of the WCICA (2004) 2445-2447

A Rapid Response Intelligent Diagnosis Network Using Radial Basis Function Network[*]

Guangrui Wen, Liangsheng Qu, and Xining Zhang

Research Institute of Diagnostics & Cybernetics, College of Mechanical Engineering, Xi'an Jiaotong University, Xi'an 710049, China
grwen@mail.xjtu.edu.cn

Abstract. An intelligent diagnostic system for a large rotor system based on radial basis function network, called rapid response intelligent diagnosis network (RRIDN), is proposed and introduced into practice. In this paper, the principles, model, net architecture, and fault feature selection of RRIDN are discussed in detail. Correct model architecture selection are emphasized in constructing a radial basis neural network of high performance. In order to reduce the amount of real training data, the counterexamples of real data are adopted. Some training and testing results of rapid response intelligent diagnosis networks are given. The practical effects in two chemical complexes are analyzed. Both of them indicate that RRIDN possesses good function.

1 Introduction

Large rotating machinery is key industrial equipment and plays important role in oil refinery, petrochemical plants and power plants, etc.. There are vast demands for fault diagnosis of large rotor system in these enterprises. Really, modern methods based diagnostic systems are now applied and have obtained some effects in practice. However, for long-range continuous running, highly automatic and complex equipment, model based diagnostic systems are very time-consuming in their modeling processes. Further more, traditional pattern recognition based diagnosis systems depend too much on the training data and the actual circumstances. To properly select the essential features is also a problem. Inappropriate choice often leads to very complex decision rules [1], [2]. If the model fails in use or the training data need to be changed, the whole diagnostic system will be paralyzed and require to repeat the task all over again. For diagnostic expert systems, it is well acknowledge acquisition and self-learning are their bottle-necks [3], [4].

In this paper, estimation of probability density function and Bayes rule based radial basis neural networks and its advantages in fault diagnosis of large rotating machinery are introduced. The practice in the chemical complexes shows that the system is fast responsible, operator independent and with better adaptability.

[*] This paper is sponsored by NSFC 50335030.

2 Principle of Radial Basis Function Network

The radial basis function network is a feedforward structure with a radial basis function in the hidden layer. The character is that its response decreases monotonically with distance from a center point. The center, the distance scale, and the precise shape of the radial basis function are the parameters of the model. In this paper we use Gaussian function in the hidden layer. The structure of the radial basis function network is illustrated in Fig. 1.

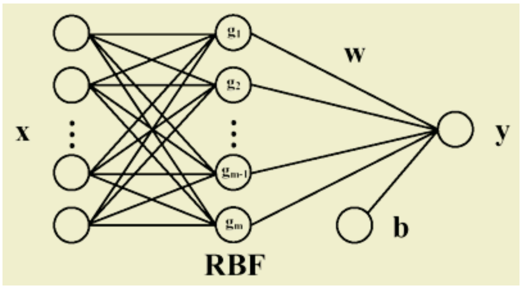


Fig. 1. Structure of radial basis function network.

There are three layers in RBF network. *X* is the input, Y is the output and it is connected with the radial basis neurons in the hidden layer by the weight W. It represents a mapping from $x \in R^n$ to $y \in R$:

$$y = \sum_{i=1}^{m} w_i g_i(\|x - c_i\|) + b \tag{1}$$

where *m* is the number of neurons in the hidden layer, w_i is the weight associated between the *i*th neuron in the hidden layer and the output, and g_i is the Gaussian function represented by:

$$g_i(\|x - x_i\|) = \exp\left(\frac{\|x - x_i\|^2}{2\sigma^2}\right) \tag{2}$$

The parameter σ and x_i are the standard deviation and the center of Gaussian function, which are determined with the weight w_i during the training of network. The network has a universal approximation capability. Theoretically, it can approximate any nonlinear function with sufficient RBF neurons in the hidden layer [5].

3 Model of Rapid Response Intelligent Diagnosis Networks (RRIDN)

As an intelligent and practical diagnostic system for large rotating machinery, completes the whole process beginning from the primary vibration signal picked-up, approaching to the identification of the faults and failure extent. In order to satisfy the demand, the system is divided into five steps: system input unit, signal processing unit, feature extraction unit, fault diagnosis unit and system output unit. The model architecture of RRIDN is shown in Fig.2.

The fault identification unit is the kernel of RRIDN. The performance of this unit is to classify the faults and to evaluate vibration level. The RRIDN fault classifier, as the most important fault-recognizing tool, provides automatic fault classification function. In order to improve the fault classification ability, it is very important to select the fault features of the classifier, apply proper training methods and accumulate sufficient sample signals of high quality with different kinds of faults [6].

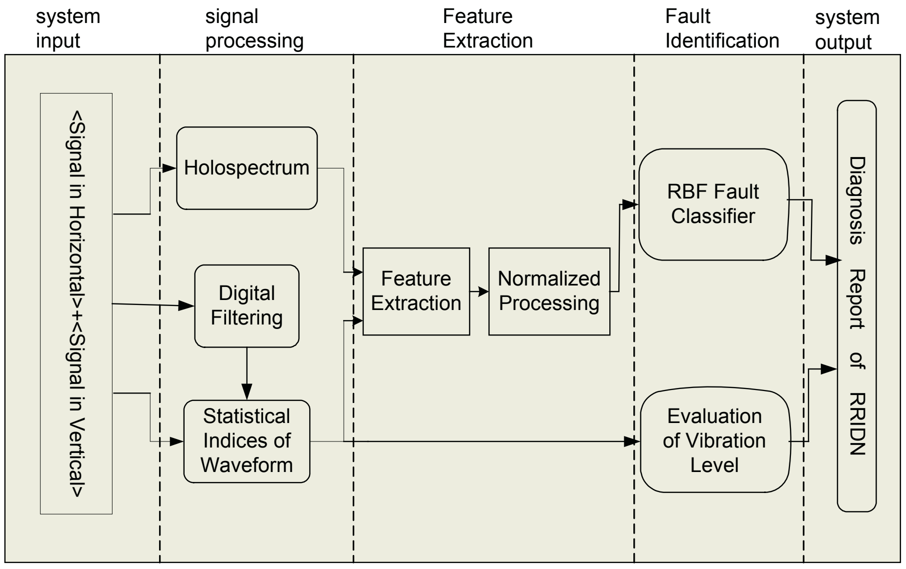


Fig. 2. Model architecture of RRIDN.

4 Fault Feature Extraction

From the point of view of application, RRIDN minimizes the operator interference and operates in a highly automatic and intelligent level. Once the system is initialized, only very few parameters need to be input or confirmed by operator. Majority of the information are automatically picked up by the system, and then processed until diagnostic conclusions without any operator's interference.

Since the fault mechanism of large rotating machinery is very complicate and the categories of the detail faults are numerous, priority is given to the typical faults that occur frequently or can make huge damage to the diagnostic objects. According to the statistical result from some industries, these typical faults and their distribution probabilities are listed in Table 1. They basically satisfy the demands of industrial application guaranteeing safe operation of large rotating machinery.

According to the diagnostic knowledge of these typical faults, the related features are presented in Table 2. In this table, these features are divided into four types. Type A is the features related with the rotating speed of machinery. These features present

Table 1. The typical fault types and their distribution probabilities.

No.	Typical fault types	Abbreviation	Probabilities
1	Unbalance	**BAL**	0.365
2	Misalignment	**ALN**	0.212
3	Rub Failure	**RUB**	0.135
4	Rotor Crack	**CRK**	0.012
5	Oil Whip	**WHP**	0.035
6	Rotating Stall	**STL**	0.035
7	Compressor Surge	**SUR**	0.035
8	Fluid Extraction	**EXC**	0.024
9	Bearing Cap Loosen	**LSN**	0.012
10	Gas Seal Wear	**SEL**	0.024
11	50Hz Interference	**ITF**	0.012
12	Low Freq. Pipe Excitation	**LFP**	0.082
13	High Freq. Pipe Excitation	**HFP**	0.017

Table 2. The interrelationships among features types, fault features, fault types, processing methods and sub-nets.

Feature Type	Fault Feature	Fault Type													Processing Method
		BAL	ALN	RUB	CRK	HFP	LFP	WHP	STL	ITF	SUR	EXC	LSN	SEL	
	1×	●	●		●			●	●						
	2×		●		●										
	3×		●	●											
	4×		●	●											
A	H×			●											H[1]
	0.4~0.52 ×							●							
	0.7~0.86 ×								●						
	FHP					●									
B	FLP						●								
	50Hz									●					
C	P-P										●	●	●	●	S[2]
D	CN										●	●	●	●	D[3]+S
Sub-net Selection				1		2		3	4	5		6		7	

[1] H denotes holospectrum. [2] S denotes statistical indices. [3] D denotes digital filtering.

in the forms of higher or lower harmonics of rotating frequency and possessed fixed ratios with the latter. After the rotating frequency being defined, these features can be picked up automatically based on Holospectrum. Type B consists of the features that possess definite characteristic frequency component, sometime with its own harmonics no matter how the rotating frequency locates. The feature of Type C can be extracted more easily. The features of Type D are processed bye means of digital filtering of input signals and then calculating their statistic indices. These features discriminate the faults manifesting as colored noise in sub-harmonic region [7].

Here, in order to decrease the complexity of architecture and the time-consuming for training and testing processes, we separately give all the effective features for each fault marked by "●" in Table 2.

5 Experiments and Discussion

In order to obtain good fault identification function, the training of RRIDN is an important link and the limited amount of training data available still remains the chief problem for RRIDN construction. So we use counterexamples as the training data to solve the problem [8].

After the training and testing is completed, RRIDN has been put into the diagnostic practice in some Industrial Corp. in China. We selected 10 groups of faulty data from 5 large compressor sets. These faulty data consist of 5 kinds of faults listed in Table 3. In addition to the manually diagnostic results, the diagnostic results by RRIDN are also listed in the table. These results show that the diagnosis error rate of RRIDN is 15.4 percent.

Table 3. The typical fault types and their distribution probabilities.

Group	Judged by Human					RRIDN Reported					Error
No.	BAL	RUB	ALN	SEL	SUR	BAL	RUB	ALN	SEL	SUR	Rate(%)
1	●					●					0
2	●					●					0
3	●	●				●	●	●			34
4		●					●				0
5	●		●			●		●			0
6			●			●		●			50
7	●	●		●		●	●				34
8		●		●			●		●		0
9	●				●	●				●	0
10		●		●		●	●		●		34
Total	6	5	2	3	1	8	5	3	2	1	15.4

Then, to be aimed at some serious faults, we analyzed the diagnostic results of RRIDN. The first sample data was typical unbalance from a fume turbine. The second sample data consists of 4 kinds of faults, i.e.: rub, seal wear, misalignment and unbalance. The third sample data is about surge and unbalance. The diagnostic reports of RRIDN are listed in Table 4, since the RRIDN left the misalignment fault in second data behind. We immediately retrained RRIDN using this faulty data. The retraining time does not exceed one minutes and the correct results were obtained. These good results prove that RRIDN is a successful intelligent and practical diagnostic system.

Table 4. The diagnostic results for 3 Groups of serious faults.

Group	Judged by Human					RRIDN Reported				
No.	BAL	RUB	ALN	SEL	SUR	BAL	RUB	ALN	SEL	SUR
1	●					●				
2	●	●	●	●		●	●	X	●	
3	●				●	●				●

6 Conclusions

This paper discusses the methodology to construct the rapid response intelligent diagnosis networks and describes a diagnostic method for large rotating machinery based on radial basis neural networks and other useful technologies. The model of RRIDN is first constructed by some efficient tools. Then the method of fault feature extraction is also proposed to satisfy the model. In order to decrease the requirement to the amount of training data in RRIDN construction, the counterexamples are used to improve the function of RRIDN.

After trained, RRIDN is applied to diagnostic practice. The results shows its good performance: the system is fast responsible, operator independent and with good adaptability.

References

1. Himmelblau, D.M.: Fault Detection and Diagnosis in Chemaical and Processes. Amsterdam: Elsevier (1987) 108–121
2. Pao, Y.: Adaptive Pattern Recognition and Neural Networks. New York: Addison-Wesley, Inc. (1989) 415-438
3. Frank, P.M.: Fault Diagnosis in Dynamic Systems Using Analytical and Knowledge Based Redundancy-A Survey and Some New Results. Automatica 26 (1990) 459-474
4. Gallant, S.I.: Connectionist Expert System. Communication on ACM 31 (1998) 152-169
5. Specht, D.F.: Probabilistic Neural Networks. Neural Networks 3 (1990) 109-118
6. Jalel, N.A.: The Application of Neural Networks for Fault in Nuclear Reactor. EE91098845, N91-19772 (1991)
7. Qu, L.S.; Discovering the Holospectrum. Mechanical System and Signal Processing (1989) 58-62
8. Xu, G.: A Rapid Response Neural Network for Rotor System Diagnosis. Proceedings of International Conference on Intelligent Manufacturing (1995) 297-301

An Integrated Approach to Fault Diagnosis Based on Variable Precision Rough Set and Neural Networks

Qingmin Zhou[1] and Chenbo Yin[2]

[1] School of Information Science and Engineering, Nanjing University of Technology, Nanjing, Jiangsu 210009, China
mse@njut.edu.cn

[2] School of Mechanical and Power Engineering, Nanjing University of Technology, Nanjing, Jiangsu 210009, China
yinchenbo@njut.edu.cn

Abstract. In the paper, a method of fault diagnosis based on integration of the variable precision rough set model (VPRSM) and neural networks is proposed. By VPRSM attribute reduction, redundant attributes are identified and removed. The reduction results are used as the inputs of neural network. With the proposed method an example for rotary machinery fault diagnosis is given. The diagnosis results show that the proposed approach has better learning efficiency and diagnosis accuracy.

1 Introduction

Neural networks for pattern recognition have good adaptability and self-leaning ability, which makes neural networks to be applied in fault diagnosis [1],[2],[3]. However, neural networks based fault diagnosis systems are often too slow and inefficient. The main reason is that neural networks can't deal with the redundant information from the fault diagnosis vibration signal. The redundant information will easily lead to some problems such as too complex networks structure, long training time and even diagnosis mistake. Because of these disadvantages the further applications of neural networks in the fault diagnosis are limited.

Rough set theory introduced by Z. Pawlak in 1982 has been described as a mathematical tool to deal with vagueness and uncertainty [4], [5]. As an extension of original rough set model, the variable precision rough set model (VPRSM) is defined by W. Ziarko [6], [7]. This is an important extension, which will give us a new way to deal with the noisy data, such as fault diagnosis vibration signal.

In this paper, a method of fault diagnosis based on integration of VPRSM and neural networks is proposed. The method integrates the ability of VPRSM on reduction of diagnosis information system and that of neural networks for fault classification. VPRSM and neural networks show respectively advantages in fault diagnosis. VPRSM are taken as a preprocessing unit of fault diagnosis neural networks. By VPRSM, redundant diagnosis information is reduced and then the reduction results are handled as inputs of neural networks. Therefore, dimensions of input data are decreased and structures of neural networks are improved. Using the method, an example for rotary machinery fault diagnosis is given. The diagnosis results show that the method has better learning efficiency and diagnosis accuracy.

2 The Variable Precision Rough Set Model

2.1 The VPRS Model

For a given information system $S=(U,A,V,f)$, $X\subseteq U$, $B\subseteq A$, lower and upper approximation are defined with precision level β. The value β denotes the proportion of correct classifications [10], in this case, the domain of β is $0.5<\beta\leq 1$.

$$\underline{B}^{\beta}(X)=\bigcup\{[x]_B : P(X/\ [x]_B)\geq\beta\}. \quad (1)$$

$$\overline{B}^{\beta}(X)=\bigcup\{[x]_B : P(X/\ [x]_B)>1-\beta\} . \quad (2)$$

$\underline{B}^{\beta}(X)$ and $\overline{B}^{\beta}(X)$ are respectively called the lower and upper approximation of X with precision level β. Here, $[x]_B$ is the equivalence class, $x\in U$. $P(X/\ [x]_B)$ is referred as conditional probability function [6], [8], [9].

$$P(X/\ [x]_B)=|X\cap[x]_B|/|[x]_B| . \quad (3)$$

Where $|X|$ is the cardinality of the set X.

The β-positive, β-negative and β-boundary regions are respectively defined by

$$POS_B^{\beta}\ (X)=\bigcup\{[x]_B : P(X/\ [x]_B)\geq\beta\}, \quad (4)$$

$$NEG_B^{\beta}\ (X)=\bigcup\{[x]_B : P(X/\ [x]_B)\leq 1-\beta\}, \quad (5)$$

$$BN_B^{\beta}(X)=\overline{B}^{\beta}(X)-\underline{B}^{\beta}(X)=\bigcup\{\ [x]_B : P(X/\ [x]_B)\in(1-\beta,\beta)\} . \quad (6)$$

As β decreases, the boundary region of the VPRSM becomes narrower. That is to say, the size of the uncertain region is reduced. The VPRSM can come back to the original rough set model when $\beta=1$.

2.2 Approximate Reduction

For a given information system $S=(U,A,V,f)$, $A=C\bigcup D$, C is called condition attribute set, while D is called decision attribute set . $X\subseteq U$, $B\subseteq C$. The measure of classification quality is defined by β-dependability of knowledge as follows:

$$\gamma^{\beta}(B,D)=\left|\bigcup\left\{[x]_B : \frac{|X\cap[x]_B|}{|[x]_B|}\geq\beta\right\}\right|\Big/|U| \cdot \quad (7)$$

The β-dependability $\gamma^{\beta}(B,D)$ measures the proportion of objects in the universe, for which classification is possible with the precision level β.

In VPRSM, knowledge reduction is aimed to select the minimum attribute subsets of C which don't change the quality of classification with the precision level β [9]. It is assumed that $red^{\beta}(C,D)$ is β approximate reduction, then

$$\gamma^{\beta}(C,D)=\gamma^{\beta}(red^{\beta}(C,D),D) . \quad (8)$$

No proper subset of $red^{\beta}(C,D)$ at the same β value can also give the same quality of classification. In other words, if any one attribute from $red^{\beta}(C,D)$ is eliminated, the formula (8) will be invalid.

A decision system may have many β-reductions. The intersection of all reductions is called Core. People always pay attention to the reduction that have the least attributes – minimal reduction sets. However, it has been proved that the minimal reductions of the decision system are the NP-hard [11]. The most important thing is to find a reduction algorithm with low computation cost.

3 VPRSM and Neural Networks Based Approach for Fault Diagnosis

3.1 Basic Framework

The basic framework is shown in Fig.1. Vibration signals data from experiment are divided into training samples and testing samples. Continuous attributes of training samples and testing samples must be discretized. By VPRSM reduction, redundant attributes are identified and removed. The reduction set of attributes can be regarded as the input of neural networks. The output of neural networks corresponds to the conclusion of the diagnostic rules. The diagnostic rules prediction accuracy can be tested by using the neural network for fault testing samples. Finally diagnosis results are analyzed.

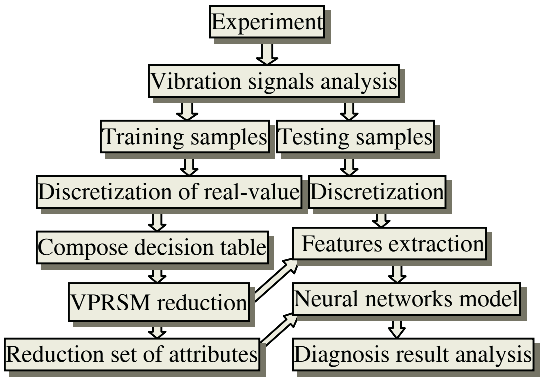


Fig. 1. The basic framework of fault diagnosis based on VPRSM and neural networks

3.2 Experimental Data Analysis and Data Preparation

In fault diagnosis, there are many feature parameters. We can only extract the necessary fault feature attributes and from these features obtain diagnosis rules. For fault diagnosis of rotating machinery, time domain, frequency domain and amplitude value domain can be regarded as fault features. Because the vibration signals of rotating machinery in the frequency domain are obvious, the frequency domain feature of vibration signals is regarded as main fault feature. The damage extent of fault is evaluated by energy distribution of frequency. The power spectrum data are smaller influenced by noise than the time series data. In this paper, power spectrum data are used as fault diagnosis signal. Typical faults of rotating machinery were simulated in our rotor test-bed. Among the rotating equipments, rotor unbalance, bearing oil whip and misalignment are the common faults. In the paper three typical faults power spectrum of rotor collected from our rotor experiments, where the rotor rotation speed is 4860 r/min and sampling frequency is 2560 Hz. Fault feature should be selected by removing unreasonable features with narrow intervals. Suppose that E_V is the average value of vibration energy for each feature frequency spectrum, then

$$E_V = \sqrt{\sum_{j=1}^{n} A_j^{\,2}} \Big/ \sqrt{N_{BF}} \cdot \qquad (9)$$

where A_j is the amplitude of FFT spectrum, N_{BF} is the width of noise band with window (for Hamming window, N_{BF} =1.5). By calculating the average value of vibration energy E_V and choosing a proper threshold μ [12], the attribute value c_i of decision table can be calculated.

If $E_V \geq \mu$ Then c_i={1}

Else c_i={0}

c_i={1} represents abnormal spectrum; c_i={0} represents normal spectrum.

3.3 Fault Diagnosis Based on VPRSM and Neural Networks

In fault diagnosis, there are not one to one definite relations between the fault reasons and the fault symptoms with relatively high information redundancy. The VPRSM can effectively eliminate the redundant information by means of reduction and mining. For fault diagnosis decision system, all equivalence classes for fault symptom sets C and fault type sets D are respectively computed, namely $U/C=\{X_1,X_2,...,X_i\}$ and $U/D=\{Y_1,Y_2,...Y_j\}$. For inconsistent equivalence class X_i, conditional probability function $P(Y_j/\ X_i)$ is calculated. The appropriate precision level β is ascertained. For noisy domains or inconsistent data, β takes smaller value, otherwise β takes bigger value. For a given probability value β, the β-positive region corresponding to a concept is delineated as the set of objects with conditional probabilities of allocation at least equal to β [9],[13]. Lower approximation $\underline{R}_B^{\beta}(X)$ and upper approximation $\overline{R}_B^{\beta}(X)$ are calculated with precision level β. Then we calculate β-dependability $\gamma^{\beta}(B,D)$ and β-accuracy $\alpha_B^{\beta}(x)$. The fault symptom sets (condition attributes) are reduced. For condition attribute reduction, the condition attributes are checked whether all the attributes are indispensable. If the classification ability and the β-dependence of decision table are not changed by eliminating an attribute, then this attribute can be deleted, otherwise it can't be deleted. Each reduction is a representation of the samples which don't change the quality of classification with the precision level β. We select some of the reduction sets to train BP neural network.

Neural network is trained after being reduced. In general, the three-layer BP neural network is suitable for fault diagnosis of rotating machinery. Therefore BP neural networks are used as classification model in our work. A three layers BP neural network is constructed that includes input layer, hidden layer and output layer. The training of BP neural network is performed with the reduction set. Attributes of reduction set are regarded as input layer $X_n = \{X_{n1}, X_{n2}, ..., X_{ni}\}^T$; The output layer corresponds to the conclusion of the diagnostic rules, $Y_n = \{Y_{n1}, Y_{n2}, ..., Y_{ni}\}^T$.In the training process of neural network, if the average quadratic error $\delta \le 0.001$, the learning is successful. Otherwise the training continues. The samples that cannot be learned by BP neural network are deleted from the training samples. Diagnostic rules are acquired.

The outputs from all trained BP neural networks are combined with the diagnosis classification results and are tested by fault testing samples. The diagnostic rules prediction accuracy can be tested by using the BP neural network for the testing samples. Finally diagnosis results are analyzed and synthesized.

4 Application

The fault diagnosis for rotating machinery is used as an example. By experiments 80 groups sample data of rotor are collected and are divided into training samples and testing samples. The training samples include 50 samples shown in table 1. The 30 testing samples are used for testing. $U = \{x_1, x_2, ..., x_{50}\}$ is a finite sets of the fault samples; The condition attribute $C = \{C_1, C_2, ..., C_8\}$ is the fault symptom set, and $C_1, C_2, \ldots, C_8$ indicate respectively frequency range, here f_0 is rotating frequency. Decision attribute D means the fault type. D={1,2,3}shows that the sample has respectively the rotor unbalance fault, oil whip fault and rotor misalignment fault symptom. Because the reference point of power spectrum obtained from the rotor test-bed does not lie in zero point, so it must be converted to zero reference point. The maximum point of absolute value of negative number is as zero point during computing. Then the attributes of decision table are discretized, the attribute value is {1, 0}, which represents respectively abnormal spectrum and normal spectrum. The decision table of fault diagnosis is built, see table 2.

Table 1. Fault diagnosis training samples

U	$0.01\sim0.39 f_0$	$0.40\sim0.49 f_0$	$0.5 f_0$	$0.51\sim0.99 f_0$	f_0	$2 f_0$	$3 f_0$	$5 f_0$	D
	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	
x_1	-32.20	-28.03	-30.15	-23.27	8.09	-20.21	-22.43	-16.84	1
x_2	-29.11	-30.65	-32.76	-11.20	9.68	-21.07	-19.18	-18.50	1
x_3	-20.19	24.05	22.86	19.06	0	-14.43	-28.81	-17.66	2
…	…	…	…	…	…	…	…	…	…
x_{50}	-44.37	-48.10	-49.28	-36.22	-18.79	-8.17	-10.01	-22.94	3

Table 2. The decision table of fault diagnosis

U	$0.01\sim0.39 f_0$	$0.40\sim0.49 f_0$	$0.5 f_0$	$0.51\sim0.99 f_0$	f_0	$2 f_0$	$3 f_0$	$5 f_0$	D
	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	
x_1	0	0	0	0	1	0	0	0	1
x_2	0	0	0	1	1	0	0	0	1
x_3	0	1	1	1	0	0	0	0	2
…	…	…	…	…	…	…	…	…	…
x_{50}	0	0	0	0	1	1	1	0	3

We take $\beta = 0.8$. Lower approximation $\underline{R}_B^\beta(X)$ and upper approximation $\overline{R}_B^\beta(X)$ are calculated with precision level $\beta = 0.8$. Then its β-accuracy $\alpha_B^\beta(x)$ and β-dependability $\gamma^\beta(B, D)$ are calculated. It results that $\alpha_B^\beta(x)$ is equal to 1.0 and $\gamma^\beta(B, D)$ to 1.0. This indicates that the approximate classification ability of decision table with precision level $\beta = 0.8$ is accord with request. The fault symptom sets (condition attributes) are reduced. The Core of the condition attribute reduction is calculated, here the Core is empty. There are four simple reduction subsets of condition attribute C relatively to decision attribute D. Those are $\{C_2, C_5, C_6\}$,$\{C_4, C_5, C_7\}$,$\{C_3, C_6, C_7\}$ and $\{C_3, C_4, C_6, C_7\}$. We select different reduction sets as inputs of BP neural network. Neural networks acquire 'diagnostic rules' from examples through the process of training. Limited by the length of the paper, we only select reduction subsets $\{C_2, C_5, C_6\}$ as inputs of BP neural network. The performance of the BP neural network with reduced 3-input dimension is compared with the original 8 inputs, which the number of the network input nerve cells is effectively decreased. The structures of neural networks are 3-8-3, namely, 3-input nodes, 8- hidden nodes and 3-output nodes. In the training process, if $\delta \le 0.001$, then the learning is successful. In the training process for reduction subsets $\{C_2, C_5, C_6\}$, diagnostic rules are acquired as follows:

$C_2[0]\ C_5[1] \rightarrow D[1]$ $C_5[1]\ C_6[0] \rightarrow D[1]$ $C_2[1]\ C_5[0] \rightarrow D[2]$
$C_2[1]\ C_6[0] \rightarrow D[2]$ $C_5[1]\ C_6[1] \rightarrow D[3]$

The rules above can be also expressed:

1) If the spectrum in f_0 is abnormal, and the spectrum in (0.40~0.49) f_0 and in $2f_0$ are normal, then the rotor unbalance fault exists.
2) If the spectrum in (0.40~0.49) f_0 is abnormal, and the spectrum in f_0 and in $2f_0$ are normal, then the rotor oil whip fault exists.
3) If the spectrum in f_0 and in $2f_0$ are abnormal, then the rotor misalignment fault exists.

These rules are tested by 30 testing samples. The average accuracy is 0.852. Hence, these rules can be regarded as diagnostic rules to diagnose fault of rotating machinery. Similarly, we can use the method to extract diagnostic rules for other reduction subsets. For four reduction subsets of this example, in all we extracted 15 diagnostic decision rules, which the average accuracy is approximately 0.817.

5 Conclusions

In fault diagnosis there is no one to one relation between the fault reasons and the fault symptoms. Therefore the feature extraction and rules acquisition from redundant fault information data are always difficult problem in the traditional fault diagnosis. In the paper, a method of fault diagnosis based on integration of the variable precision rough set model (VPRSM) and neural networks is proposed. The data information of fault diagnosis is obtained directly from vibration signal system in which there are the existence of noisy data and uncertain information. The method can effectively eliminate the redundant fault attributes and acquire the correct diagnosis rules from noisy fault information. This method explores VPRSM based approach to reduce the input-dimension of neural networks for diagnosing of rotating machinery faults. The diagnostic results indicate that the proposed approach is effective in reducing the redundancy of neural networks training samples and extracting the accurate diagnosis rules.

References

1. Liu, S C, Liu, S Y.: An Efficient Expert System for Machine Fault Diagnosis. International Journal Advanced Manufacturing Technology, **21** (2003) 691-698
2. Paya, B. A., East, I. I., Badi, M. N.: Artificial Neural Network Based Fault Diagnostics of Rotating Machinery Using Wavelet Transforms as a Preprocessor. Mechanical Systems and Signal Processing 11 (1997) 751-765
3. Depold, H. R., Gass, F. D.: The Application of Expert Systems and Neural Networks to Gas Turbine Prognostics and Diagnostics. Journal of Engineering for Gas Turbines and Power, Transactions of the ASME , 121 (1999) 607-612
4. Pawlak, Z.: Rough sets.: International Journal of Computer and Information Sciences, **11** (1982) 341-356
5. Pawlak, Z.: Why Rough Sets. Proceedings of the Fifth IEEE International Conference on Fuzzy Systems, **2** (1996) 738-743
6. Ziarko, W.: Analysis of Uncertain Information in the Framework of Variable Precision Rough Sets. Foundations of Computing And Decision Sciences ,**18** (1993)381-396
7. Katzberg, J.D., Ziarko, W.: Variable Precision Extension of Rough Sets. Fundamental Informatics, **27** (1996)155-168
8. Beynon, M.: An Investigation of β-reduct Selection within the Variable Precision Rough Sets Model. In: Ziarko W. and Yao Y. (Eds.) Proceedings of RSCTC 2000, LNAI 2005. Springer-Verlag, Berlin Heidelberg (2001) 114-122
9. Beynon, M.: Reducts within the Variable Precision Rough Set Model: A Further Investigation. European Journal of Operational Research, **134** (2001) 592-605
10. An, A., Shan, N., Chan, C., Cercone, N., Ziarko, W.: Discovering Rules for Water Demand Prediction: An Enhanced Rough-set Approach. Engineering Applications in Artificial Intelligence, **9** (1996) 645-653
11. Skowron A, Rauszer C.: The Discernibility Matrices and Functions in Information Systems. Intelligent Decision Support-Handbook of Applications and Advances of the Rough Sets Theory, Kluwer Academic Publishers (1992) 331-362
12. Hu, T., Lu, B.C., Chen, G.J.: A Gas Turbo Generator Fault Diagnosis New Approach Based on Rough Set Theory. Journal of electronic measurement and instrument 15 (2001) 12-16
13. Zhou, Q.M., Yin, C.B., Li, Y.S.: The Variable Precision Rough Set Model for Data Mining in Inconsistent Information System. In: Chi, C.-H., Lam, K.-Y. (Eds.) Proceedings of AWCC 2004,. Springer-Verlag, Berlin Heidelberg (2004) 285-290

Hybrid PSO Based Wavelet Neural Networks for Intelligent Fault Diagnosis*

Qianjin Guo[1,2], Haibin Yu[1], and Aidong Xu[1]

[1] Shenyang Inst. of Automation, Chinese Academy of Sciences, Liaoning 110016, China
guoqianjin@sia.cn
[2] Graduate School of the Chinese Academy of Sciences, Beijing 100039,China
yhb@sia.cn

Abstract. A model of wavelet neural network (WNN) using a new evolutionary learning algorithm is proposed in this paper. This new evolutionary learning algorithm is based on a hybrid of Particle Swarm Optimization (PSO) and gradient descent algorithm (GD), and is thus called HGDPSO. The Particle Swarm Optimizer has previously been used to train neural networks and generally met with success. The advantage of the PSO over many of the other optimization algorithms is its relative simplicity and quick convergence. But those particles collapse so quickly that it exits a potentially dangerous property: stagnation, which state would make it impossible to arrive at the global optimum, even a local optimum. HGDPSO was proposed for neural network training to avoid premature and eliminate stagnation in PSO. The effectiveness of the HGDPSO based WNN is demonstrated through the classification of the fault signals in rotating machinery. The simulated results show its feasibility and validity.

1 Introduction

With the development of industry, the machine or the production system is becoming more and more complicated, and the fault diagnosis for such large and complicated system is also becoming more and more difficult. Any know-how to help to make the fault diagnosis for the large-scale system easier to handle and implement, and make the diagnosis accuracy is clearly desirable. Recently, neural networks have become a popular tool in fault diagnosis due to their fault tolerance and their capacity for self-organization. The multi-layer perception (MLP) [1], along with the back-propagation (BP) training algorithm, is probably the most frequently used type of neural network in practical applications. Unfortunately, these ANNs have some inherent defects, such as low learning speed, existence of local minima, and difficulty in choosing the proper size of network to suit a given problem. To solve these defects, we combine wavelet theory with it and form a wavelet neural network (WNN) whose activation functions are drawn from a family of wavelets.

The wavelet neural network has been proposed as a novel universal tool for functional approximation [2], which shows surprising effectiveness in solving the conventional problem of poor convergence or even divergence encountered in other kinds of

* Foundation item: Project supported by the National High-Tech. R&D Program for CIMS, China (Grant No. 2003AA414210).

neural networks. It can dramatically increase convergence speed [3],[4]. Advances in its theory, algorithms and application have greatly influenced the development of many disciplines of science and technology, including mathematics, physics, engineering and etc., and will continue to influence other areas as well [5],[6].

How to determine the structure and parameters of the neural networks promptly and efficiently has been a difficult point all the time in the field of neural networks research [7]. In previous work, the standard PSO algorithm has been used in training MFNNs [8],[11],[13]. Other variations of PSO were also to train MFNNs [8],[9],[10], and the performance was acceptable. Particle swarm optimization (PSO) developed by Dr. Kennedy and Dr. Eberhart [12],[13] is one of the modern heuristic algorithms under the EAs and gained lots of attention in various engineering applications [14],[15],[16]. PSO when compared to EP has very fast converging characteristics, however it has a slow fine-tuning ability of the solution. Also PSO has a more global searching ability at the beginning of the run and a local search near the end of the run [13]. Therefore, while solving problems with more local optima, there are more possibilities for the PSO to explore local optima at the end of the run.

To overcome this drawback, a hybrid method that integrates the PSO with a gradient descent algorithm is proposed in this paper. To validate the performance of the proposed approach, fault diagnosis for rotating machine is tested and the results obtained are compared with those obtained using PSO, GD technique in this paper.

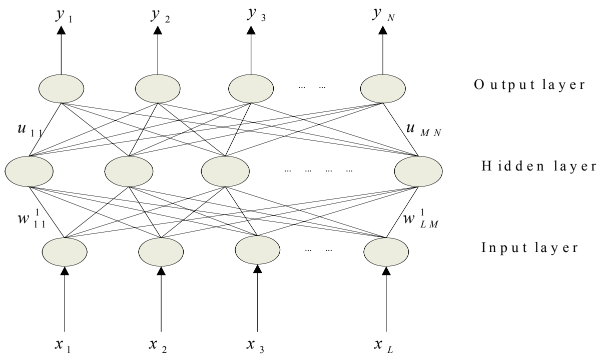


Fig. 1. The WNN topology structure

2 Neural Network for Fault Diagnosis

The WNN employed in this study are designed as a three-layer structure with an input layer, wavelet (hidden) layer and output layer. The topological structure of the WNN is illustrated in Fig.1, where w_{jk} denotes the weights connecting the input layer and the hidden layer, u_{ij} denote the weights connecting the hidden layer and the output layer. In this WNN models, the hidden neurons have wavelet activation functions of different resolutions, the output neurons have sigmoid activation functions.

The activation functions of the wavelet nodes in the wavelet layer are derived from a mother wavelet $\psi(x)$, Suppose $\psi(x) \in L^2(R)$, which represents the collection of all measurable functions in the real space, satisfies the admissibility condition [17]:

$$\int_{-\infty}^{+\infty} \frac{|\hat{\psi}(\omega)|^2}{\omega} d\omega < \infty \quad . \tag{1}$$

where $\hat{\psi}$(x) indicates the Fourier transform of $\psi(x)$. The output of the wavelet neural network *Y* is represented by the following equation:

$$y_i(t) = \sigma(x_n) = \sigma\left(\sum_{j=0}^{M} u_{ij}\psi_{a,b}\left(\sum_{k=0}^{L} w_{jk}x_k(t)\right)\right)(i = 1,2,..., N) \ . \tag{2}$$

where $\sigma(x_n) = 1/(1+e^{-x_n})$. y_j denotes the *j*th component of the output vector; x_k denotes the *k*th component of the input vector; u_{ij} denotes the connection weight between the output unit *i* and the hidden unit *j*; w_{jk} denotes the weight between the hidden unit *j* and input unit *k*; a_j, b_j denote dilation coefficient and translation coefficient of wavelons in hidden layer respectively; *L,M, N* denote the sum of input, hidden and output nodes respectively.

3 Basic Particle Swarm Optimization

Particle swarm optimization (PSO) is an evolutionary computation technique motivated by the simulation of social behavior [12],[13]. In the PSO system, each agent makes his decision according to his own experiences and other agent' experiences. The system initially has a population of random solutions. Each potential solution, called a particle (agent), is given a random velocity and is flown through the problem space. The basic concept of the PSO technique lies in accelerating each agent towards its *pbest* and *gbest* locations, with a random weighted acceleration at each time step and this is illustrated in Fig.2.

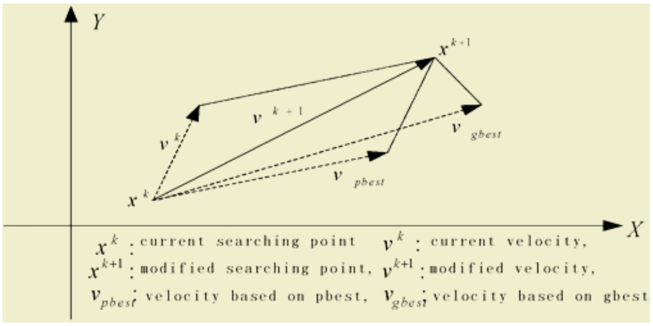

Y
X
x^{k+1}
v^{k+1}
v^k
v_{gbest}
v_{pbest}
x^k
x^k : current searching point v^k : current velocity,
x^{k+1} : modified searching point, v^{k+1} : modified velocity,
v_{pbest}: velocity based on pbest, v_{gbest}: velocity based on gbest

Fig. 2. Concept of modification of a searching point

Searching procedures by PSO can be described as follows: a flock of agents optimizes a certain objective function. Each agent knows its best value so far (*pbest*) and its position. Moreover, each agent knows the best value in the group (*gbest*) among *pbest*, namely the best value so far of the group. The modified velocity and the distance from *pbest* and *gbest* as shown below:

$$v_i^{k+1} = w_i v_i^k + c_1 rand_1 \times (pbest_i - x_i^k) + c_2 rand_2 \times (gbest - x_i^k) \, . \tag{3}$$

where v_i^k denotes current velocity of agent *i* at iteration *k*; v_i^{k+1} denotes modified velocity of agent *i* at iteration *k+1*; *rand₁,rand₂* denote random number between 0 and 1; x_i^k denotes current position of agent *i* at iteration *k*; *pbestᵢ* denotes *pbest* of agent *i; gbest* denotes *gbest* of the group; w_i denotes weight function for velocity of agent *i*; c_1,c_2 , weight coefficients for each term; *k* denotes current iteration number.

Using the above equation, a certain velocity that gradually gets close to *pbest* and *gbest* can be calculated. The current position (searching point in the solution space) can be modified by the following equation:

$$x_i^{k+1} = x_i^k + v_i^{k+1} \ . \tag{4}$$

The inertia weight *w* is introduced in [18] to improve PSO performance. Suitable selection of inertia weight *w* provides a balance between global and local exploration and exploitation. The inertia weight *w* is set according to the following equation.

$$w_i = w_{\max} - \frac{w_{\max} - w_{\min}}{iter_{\max}} \times k \ \cdot \tag{5}$$

where w_{max} ,initial weight, w_{min} , final weight, $iter_{max}$ maximum iteration number.

4 Gradient Descent (GD) Algorithm

The network is trained with gradient descent (GD) algorithm in batch way [19], During the training phase, wavelet node parameters, *a, b* and WNN weights, w_{jk}, u_{ij}, are adjusted to minimize the least-square error, given the d_i^p as the *i*th desired target output of *p*th input pattern, the cost function can be written as:

$$E = \frac{1}{2}\sum_{p=1}^{P}\sum_{i=1}^{N}(d_i^p - y_i^p)^2 \ \cdot \tag{6}$$

For this wavelet neural network, Morlet wavelet has been chosen to serve as an adoption basis function to the network's hidden layer, which has been the preferred choice in most work dealing with ANN, due to its simple explicit expression.

$$\psi_{a,b}(t) = \cos(1.75 t_z) e^{-\frac{t_z^2}{2}} \quad \text{where } t_z = (\frac{t-b}{a}) \ \cdot \tag{7}$$

$$\psi_{a,b}'(t) = -\left(1.75\sin(1.75 t_z)\exp(-\frac{t_z^2}{2}) + \cos(1.75 t_z)\exp(-\frac{t_z^2}{2}) t_z\right) \tag{8}$$

Denote that $net_j = \sum_{k=1}^{L} w_{jk} x_k$

Thus

$$\psi_{a,b}(net_j) = \psi\left(\frac{net_j - b_j}{a_j}\right) \tag{9}$$

$$y_i(t) = \sigma\left(\sum_{j=1}^{M} u_{ij}\psi_{a,b}(net_j)\right)(i = 1,2,\dots N) \tag{10}$$

$$\sigma'(u) = \frac{\partial\sigma(u)}{\partial u} = \sigma(u)[1-\sigma(u)] \tag{11}$$

Such that

$$\delta_{u_{ij}} = \frac{\partial E}{\partial u_{ij}} = -\sum_{p=1}^{P} (d_i^p - y_i^p) y_i^p (1 - y_i^p) \psi_{a,b}(net_j^p) \tag{12}$$

$$\delta_{w_{jk}} = \frac{\partial E}{\partial w_{jk}} = -\sum_{p=1}^{P} \sum_{i=1}^{N} (d_i^p - y_i^p) y_i^p (1 - y_i^p) u_{ij} \psi'_{a,b}(net_j^p) x_k^p / a_j \tag{13}$$

$$\delta_{b_j} = \frac{\partial E}{\partial b_j} = \sum_{p=1}^{P} \sum_{i=1}^{N} (d_i^p - y_i^p) y_i^p (1 - y_i^p) u_{ij} \psi'_{a,b}(net_j^p) / a_j \tag{14}$$

$$\delta_{a_j} = \frac{\partial E}{\partial a_j} = \sum_{p=1}^{P} \sum_{i=1}^{N} (d_i^p - y_i^p) y_i^p (1 - y_i^p) u_{ij} \psi'_{a,b}(net_j^p) (\frac{net_j^p - b_j}{a_j^2}) \tag{15}$$

The learning rate and momentum are set as η and μ in the experiments respectively.

$$\Delta w_{jk}(t+1) = -\eta \frac{\partial E}{\partial w_{jk}} + \mu \Delta w_{jk}(t) \qquad \Delta u_{ij}(t+1) = -\eta \frac{\partial E}{\partial u_{ij}} + \mu \Delta u_{ij}(t)$$
$$\Delta a_j(t+1) = -\eta \frac{\partial E}{\partial a_j} + \mu \Delta a_j(t) \qquad \Delta b_j(t+1) = -\eta \frac{\partial E}{\partial b_j} + \mu \Delta b_j(t) \tag{16}$$

Then the parameters are updated as follows:

$$w_{jk}(t+1) = w_{jk}(t) + \Delta w_{jk}(t+1) \qquad u_{ij}(t+1) = u_{ij}(t) + \Delta u_{ij}(t+1)$$
$$a_j(t+1) = a_j(t) + \Delta a_j(t+1) \qquad b_j(t+1) = b_j(t) + \Delta b_j(t+1) \tag{17}$$

5 Hybrid Particle Swarm Optimization

There is not widespread satisfaction with the effectiveness of the gradient descent algorithm since there is a danger of getting stuck or oscillating around a local minimum. In addition, the convergence of the gradient descent algorithm during the training is sensitive to the initial values of weights. If the initial values of the weights are not properly selected, the training process is more accurate in terms of number of iteration required to reach a pre-specified error criterion. Tests have shown that particle swarm optimization can be superior to gradient descent algorithm in generalizing to previously unseen test data. The tendency of particles to "overfly" the global and personal best solutions leads to the algorithm exploring new space, and avoid the problem of being stuck on local minima.

Rather than relying solely on gradient descent algorithm or PSO, a better method might be to combine the two. If we used PSO to find the area of the best solution, we would avoid many (if not all) problems associated with a lack of PSO diversity. This would combine the best of both worlds, and avoid the pitfalls of both. In the beginning of the run, PSO has more possibilities to explore a large space and therefore the agents are freer to move and sit on various valleys. The accuracy of the PSO methods increases very slowly after many generations. The solution may fall and may stagnate in the local minima itself. In such situation, the gradient algorithm is employed to escape from the local minima. The search will continue until a termination criterion is satisfied. In this study, we used a hybrid algorithm integrating PSO with gradient descent algorithm for WNN training. We will call this algorithm HGDPSO in the following sections.

Without loss of generality, it is denoted that W being the connection weight matrix between the input layer and the hidden layer, U, the one between the hidden layers and the output layer, A, the dilation coefficient of wavelons in hidden layer, and B, the translation coefficient of wavelons in hidden layer.

It is further denoted as $X=\{W,U,B,A\}$. When a PSO is used to train he WNN, the *i*th particle is represented as $X_i=\{W_i,U_i,B_i,A_i\}$. The best previous position (i.e., the position giving the best fitness values) of any particle is recorded and represented as $P_i=\{P_i^w,P_i^u,P_i^b,P_i^a\}$, The index of the best particle among all particles in the population is represented by the symbol g, hence $P_g=\{P_g^w,P_g^u,P_g^b,P_g^a\}$ is the best matrix found in the population. The rate of position change (velocity) for particle i is represented as $V_i=\{V_i^w,V_i^u,V_i^b,V_i^a\}$. Let m and n denote the index of matrix row and column, respectively. The particles are manipulated according to the previous equations (3), (4).

The pseudo code of the proposed solution methodology that integrates the PSO with GD algorithm for WNN training can be summarized as follows:

(1) Initialize the dilation parameter a_j, translation parameter b_j and node connection weights w_{jk}, u_{ij} to some random values. The dilation parameter a_j are limited in the interval (1,3), translation parameter b_j are limited in the interval (1,2). Randomly initialize particle position $X_i \in D$ in R. Randomly initialize particle velocities $0 \le v_i \le v^{\max}$. Let k indicates iteration times, Set constants k_{max}, c_1, c_2, V^{max}, w , d. Set k=0,i=1.

(2) Input learning samples $X_n(t)$ and the corresponding output values O_n to WNN, where t varies from 1 to L, representing the number of the input nodes, n represents the *n*th data samples of training set.

(3) Initialize a population (array) of particles with random positions and velocities in the d dimensional problem space.

(4) For each particle, evaluate the desired optimization fitness function (see eq.(6)) in d variables.

(5) Compare particle's fitness evaluation with particle's *pbest*. If current value is better than *pbest*, then set *pbest* value equal to the current value and the *pbest* location equal to the current location in d-dimensional space.

(6) Compare fitness evaluation with the population's overall previous best. If the current value is better than *gbest*, then reset *gbest* to the current particle's array index and value.

(7) Change the velocity and position of the particle according to equations (3) and (4) respectively.

(8) Check whether *gbest* is trapped into the stagnate, if not go to step (10) .

(9) Use gradient descent algorithm to optimize some particles (here randomly select 3 particles) whose fitness evaluation value is not equal to the *gbest.*

(10) (Check the stop criterion). Repeat step (4) until a criterion is met, usually a sufficiently good fitness or a maximum number of iterations/epochs.

(11) Save the training parameters and the whole training of the WNN is completed.

6 Experiments for Fault Diagnosis of Rotating System

Experiments were performed on a machinery fault simulator, which can simulate the most common faults, such as misalignment, unbalance, resonance, radial rubbing, oil whirling and so on. The schematic of the test apparatus is mainly consists of a motor, a coupling, bearings, discs and a shaft etc. The fault samples are obtained by simulating corresponding fault on experiment system. The measurements with acceleration, velocity, or displacement data from rotating equipment are acquired by the NI digital signal acquisition module, and then are collected into an embedded controller. Each condition was measured with a given times continuously. The frequency of used signal is 5000Hz and the number of sampled data is 1024.

The features of vibration signals are extracted with wavelet packet analysis and FFT, the time-frequency spectrum of data is computed and fed into the training stage, in which 6 faults and 7 frequency bounds are selected to form a feature vector. These feature vectors are used as input and output of BP or WNN.

6.1 Selection of Parameters for Training Algorithms

The selection of these optimal parameters plays an important role in various training algorithms. A single parameter choice has a tremendous effect on the rate of convergence. For this paper, the optimal parameters are determined by trail and error experimentations.

In HGDPSO, a particle X is a set of parameters of WNN denoted as $\{W,U,B,A\}$. The domain of the weights w_j, the dilation a_j and translation b_j is different, so this work divides X into X_1, $X2$ and X_3, where X_1={W, U}, X_2={B}, X_3={A}, and the rate of position change (velocity) V for particle X is accordingly divided into $V1$, $V2$, $V3$ respectively, where $V_1=\{V_i^w,V_i^u\}$, $V_2=\{V_i^b\}$, $V_3=\{V_i^a\}$. The search space range available for X_1 is defined as (-100,100), X_2 is defined as (0,100) and X_3 is defined as (1,100), that is, the particle cannot move out of this range in each dimension. The other settings of the HGDPSO algorithm are as follows: V_{max} was set to 5, swarm size was set to 50, c_1 and c_2 were both set to 2, and the inertia weight was linearly decreased from 0.7 to 0.4. In PSO, the population size and initial individuals are the same as those used in HGDPSO. For fair comparison, the parameters c_1, c_2, and w are also the same as those used in HGDPSO. The GD algorithm has two parameters to be set: the learning rate and the momentum.

6.2 Network Training and Determination

In this experiment, two types of Neural Networks, namely wavelet neural networks and BP networks, are trained on fault classification using various training algorithms. In the first series of experiments, we want to test the performance of the WNN and BP network. In the second series of experiments, we want to test the performance of the various training algorithms for WNN. Each experiment was run 50 times for given iterations, and the results were averaged to account for stochastic difference.

6.2.1 Comparison of WNN and BP Method

Comparing to WNN, the same 150 groups of data are processed with a 7-10-6 BP network, expecting output error threshold is 0.001. The MSE function is same as Eq.(6). The other parameters are same as WNN.

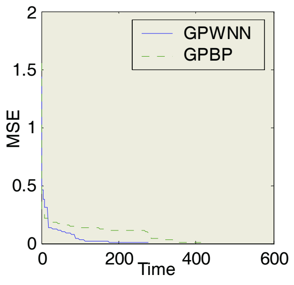


Fig. 3. Convergence curves using HGDPSO for WNN (GPWNN), BP (GPBP)

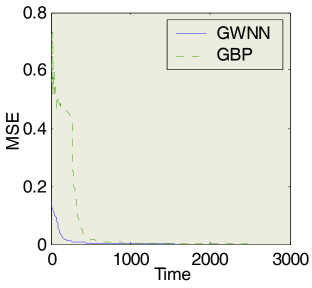


Fig. 4. Convergence curves using GD algorithm for WNN(GWNN), BP(GBP)

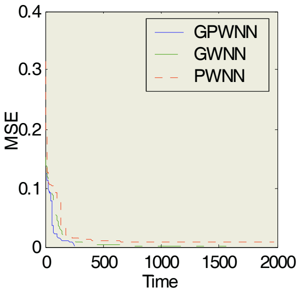


Fig.5. Convergence curves using PSO(PWNN), GD (GWNNN), HGDPSO (GPWNN) for training WNN

Fig.3 and Fig.4 demonstrate the training history and the performance of the WNN and BP networks by using HGDPSO and GD respectively. By looking at the shapes of the curves in Fig.4, it is easy to see the WNN trained with GD algorithm converges more quickly than the BP trained with GD algorithm. As seen in Fig.3, it is clear that the simulation time obtained by the WNN trained with HGDPSO algorithm is comparatively less compared to the BP networks trained with HGDPSO algorithm.

Table 1 shows the final values reached by each algorithm after 2500 iterations were performed. The second column in this table lists the diagnosis accuracy on the 150 actual sample data. The WNN achieve higher diagnosis accuracy (95.33%) of the tested set than that of that of BP (90.67%). The test results confirm that, in both cases, the proposed WNN have a better capability for generalization than the BP methods.

Table 1. Comparison of WNN and BP method

Method	Diagnosis accuracy (%)	Sum error	Epochs
GBP	90.67	0.001	2490
GWNN	95.33	0.001	1550
GPBP	93.33	0.001	410
GPWNN	96.67	0.001	280
PWNN	89.33	0.008	2000

From the comparison of training histories in WNN and BP networks trained with various methods, it can be seen that, for the given errors, the WNN presents better convergence performance than the BP networks. On considering the classify accuracy, the results produced by the WNN model are more close to the original data than those by BP networks. In general, the WNN generates more accurate classify and presents better performance than the BP ones does.

6.2.2 Comparison of HPSO, GD and PSO Algorithm

To show the effectiveness and efficiency of HGDPSO integrating the GD algorithm with the PSO, WNN designed by GD algorithm and PSO are also applied to the same fault diagnosis problem. As shown in figure 5, the MSE error of 0.001 has been reached by using HGDPSO algorithm after an average of 280 iterations in the training phase, and after 1550 iterations, the mean squared error reached the same level by using GD algorithm. Meanwhile, after 2000 iterations, the mean squared error reached by using PSO technique was not less than 0.008.

By looking at the shapes of the curves in Figure 5, it is easy to see the PSO converges quickly under training phase but will slow its convergence speed down when reaching the optima. The PSO lacks global search ability at the end of run that a method is required to jump out of the local minimum in some cases.

In the testing phase, the HGDPSO trained WNN was able to discriminate the fault examples with an accuracy higher than 96% (see Table 1), compared to GD' 95.33%, while the PSO trained WNN with an accuracy less than 90%. From Table 1, we can see that the classification error for HGDPSO was the smallest.

The results on performance clearly show that the HGDPSO is a much stronger optimizer than the basic PSO and the implemented GD algorithm on the WNN training. The HGDPSO represent a clear and substantial improvement over the basic PSO and GD algorithm, not only in the final solutions, but also in the speed with which they are found.

7 Conclusions

A HGDPSO-based wavelet neural network approach is developed for fault diagnosis. The approach takes a novel kind of optimization algorithm, i.e., HGDPSO optimization algorithm, to train wavelet neural work. The feasibility and effectiveness of this new approach is validated and illustrated by a study case of fault diagnosis on rotating machine. The data measured at a machinery fault simulator by the data acquisition module are chosen as the original data to testify the performance of the proposed method. The results show that the HGDPSO-based WNN has a better training performance, faster convergence rate, as well as a better diagnosis ability than the other methods to be selected cases.

References

1. Chen, D. S., Jain, R.C.: A Robust Back Propagation Learning Algorithm for Function Approximation. IEEE Trans. Neural Networks, **5** (1994) 67-479
2. Zhang, Q., Benveniste, A.: Wavelet Networks. IEEE Trans. Neural Networks, **3** (1992)889-898
3. Zhang, J., Walter, G., Miao, Y., Lee, W.N.: Wavelet Neural Networks for Function Learning. IEEE Trans. Signal Processing, **43** (1995)1485-1497
4. Delyon, B., Juditsky, A., Benveniste, A.: Accuracy Analysis for Wavelet Approximations. IEEE Trans. Neural Networks. 6(1995)332-348.
5. Shashidhara, H.L., Lohani, S., Gadre, V.M.: Function Learning using Wavelet Neural Networks. Proceedings of IEEE International Conference on Industrial Technology, **2** (2000) 335-340

6. Zhang, Q.: Using Wavelet Network in Nonparametric Estimation. IEEE Trans. Neural Networks, **8** (1997)227-236
7. Johansen, M.M.: Evolving Neural Networks for Classification. Topics of Evolutionary Computation 2002, Collection of Student Reports (2002) 57-63
8. van den Bergh, F., Engelbrecht, AP.: Cooperative Learning in Neural Networks using Particle Swarm Optimizer. South African Computer Journal , **26** (2000) 84-90.
9. van den Bergh, F., Engelbrecht, A.P.: Training Product Unit Networks Using Cooperative Particle Swarm Optimizers. Proc. IJCNN 2001, Washington DC, USA (2001) 126-132
10. van den Berg, F.: An Analysis of Particle Swarm Optimizers. Ph.D. Dissertation, University of Pretoria, Pretoria (2001)
11. He, Z., Wei, C., Yang,L., Gao, X., Yao, S., Eberhart, R., Shi, Y.: Extracting Rules from Fuzzy Neural Network by Particle Swarm Optimization. Proc. IEEE International Conference on Evolutionary Computation, Anchorage, Alaska, USA, (1998) 74-77
12. Kennedy, J., Eberhart, R.C.: Particle Swarm Optimization. Proc. IEEE International Conference on Neural Networks.(1995)1942-1948.
13. Kennedy, J., Eberhart, R.C., Shi ,Y.H.: Swarm Intelligence: Morgan Kaufmannn (2001)
14. Yoshida, H., Kawata, K., Fukuyama, Y., Takayama, S., Nakanishi, Y.: A particle swarm Optimization for Reactive Power and Voltage Control Considering Voltage Security Assessment. IEEE Trans. on Power Syst , **15** (2000) 1232–1239
15. Abido ,M.A.: Optimal Design of Power System Stabilizers Using Particle Swarm Optimization. IEEE Trans. Energy Convers, **17** (2002) 406–413
16. Eberhart, R.C., Hu, X.: Human Tremor Analysis Using Particle Swarm Optimization. Proc. Congress on Evolutionary Computation 1999.Washington, DC. Piscataway, NJ: IEEE Service Center (1999) 1927-1930
17. Daubechies, I.: The wavelet Transform , Time-Frequency Localization, and Signal Analysis. IEEE Trans. Inform Theory , **36** (1990) 961-1005
18. Shi, Y.H., Eberhart, R.: A Modified Particle Swarm Optimizer. Proc. IEEE International Conference on Evolutionary Computation, Anchorage, Alaska (1998) 69-73
19. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning Internal Representations by Error Propagation. Nature , 323 (1986) 533-536

Global-Based Structure Damage Detection Using LVQ Neural Network and Bispectrum Analysis

Guangming Dong[1], Jin Chen[1], Xuanyang Lei[1], Zuogui Ning[2], Dongsheng Wang[2], and Xiongxiang Wang[2]

[1]State Key Laboratory of Vibration, Shock& Noise, Shanghai Jiaotong University, Shanghai, 200240, China
{gmdong,JinChen}@mail.sjtu.edu.cn
[2]Institute of Structural Mechanics, Chinese Academy of Engineering Physics, Mianyang, Sichuan 621900, China

Abstract. In this paper, a new approach for the global-based structure damage detection is proposed, which is based on the combination of bispectrum feature extraction technique and LVQ neural network identification method. A finite element model based on a steel frame structure with various joint damage patterns is analyzed. Results of analysis demonstrate higher damage identification capability in comparison with modal assurance criterion (MAC) method in a noise-contaminated environment.

1 Introduction

The need for global-based damage detection techniques that can be applied to complex structures has led to the development of methods that examine changes in vibration characteristics of the structure. The basic idea behind this technology is that any local damage somehow changes the dynamics of the structure and that the extracted features from the structural response can reflect this change. Traditionally, most of these features have been derived from a modal analysis of the structure, e.g. resonant frequencies, mode shapes, flexibility, etc [1]. Other non-modal based methods have included ARX model [2], time-frequency analysis [3], etc. More recently, Higher-order statistics (HOS) and corresponding spectra have been used to extract information from a measured signal [4], and there has been increasing interest in using neural networks to evaluate damage status in complex structures [5]. The combination of HOS based feature extraction and neural network based feature classification is a new trend in the study of global-based structure damage detection.

In comparison with previous methods, since bispectrum analysis can provide information of the third-order properties of a signal and can restrain Gaussian noise from the environment and measurement, it is suitable for extracting the features from a signal measured in a noise-contaminated environment. On the other hand, the learning vector quantization (LVQ) neural network may be used as a good classification tool [4]. In this paper, a finite element model based on a four story steel frame structure with various joint damage patterns is analyzed and the location problem of the damaged bolted connections in the structure is carried out based on the combination of the bispectrum feature extraction technique and LVQ neural network identification method. Results of analysis demonstrate high identification capability of the location of damaged joint, especially in a low signal-to-noise environment.

2 Basic Principle of LVQ Neural Network

Learning vector quantization (LVQ) is a nearest neighbor pattern classifier based on competitive learning. The architecture of an LVQ neural network is shown in Fig. 1. The LVQ neural network is basically composed of three layers, that is, an input layer, a competitive layer and an output layer. In general, the input layer is an N-dimensional vector X with each component $x_i\ (i=1,2,\cdots,N)$ corresponding to one input feature. The competitive layer contains S nodes that usually depend on the number of classes, and the output layer possesses T units with each output unit representing a particular classification target.

The LVQ neural network is trained to find the node i for which the weight vector W_i is the closest one to the input training vector X. During the training process of the LVQ neural network, the inner potential of the competitive layer is defined by the formula: $p_i = \dfrac{1}{d_i(X,W_i)}$, where $d_i(X,W_i)$ is the Euclidean distance from a training vector X to each mode's weight W_i of the competitive layer.

The mode with maximum potential is called "winner". The value of winner is set as 1, and all other nodes in the competitive layer are set as 0, that is:

$$z_i = \begin{cases} 1, & \mathit{Winner},\ p_i \Rightarrow \text{maximum} \\ 0, & \mathit{else} \end{cases} . \tag{1}$$

The weight vector of the competitive layer is adjusted as follows:

$$W_i^{\text{new}} = W_i^{\text{old}} + \alpha(X - W_i^{\text{old}})z_i . \tag{2}$$

Where W_i^{new} is the adjusted weight vector after adjustment, and the α represents the learning coefficient.

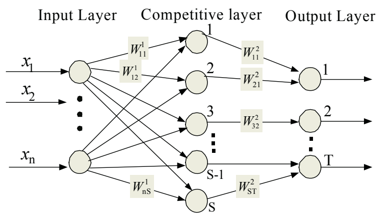


Fig. 1. Architecture of an LVQ neural network

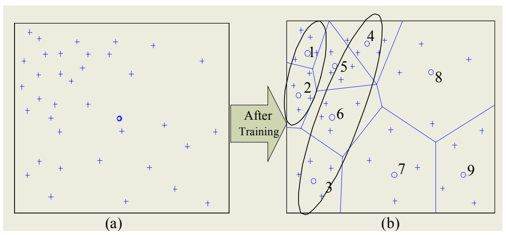


Fig. 2. Sub-areas of competitive layer of LVQ after training

After training of the LVQ neural network, the input training vector space is classified into many sub-areas in the competitive layer with each node controlling one sub-area and several sub-areas correspond to one classification target in the output layer(Fig. 2). For example, by referring to Fig. 1 if the input is a two dimensional variable and the competitive layer is supposed to include nine elements, then the training samples can be expressed in figure plotted in "+". The initial locations of each element of competitive layer are plotted in "o" illustrated in Fig. 2(a), which is the geometrical center of training samples. The whole sampling area can be classified into nine sub-areas, which is expressed in Fig. 2(b) after no-teaching-assistant training. Furthermore, it is supposed that output layer indicates three targets, sub-area 1 and 2 correspond to target1, sub-area 3, 4, 5 and 6 correspond to target 2, etc, and can be classified into three groups after teaching-assistant training.

3 Bispectrum Feature Extraction

Bispectrum analysis forms a subset of higher order statistics (HOS) analysis. First, we give the definition of the bispectrum. Suppose $\{x(n)\}$ is a kth (higher) order stationary random process and its kth order cumulant $c_{kx}(\tau_1,\cdots,\tau_{k-1})$ satisfies: $\sum_{\tau_1,\cdots,\tau_{k-1}=-\infty}^{\infty} |c_{kx}(\tau_1,\cdots,\tau_{k-1})| < \infty$; then, the kth order spectrum is defined as the $(k-i)$th dimensional discrete Fourier transformation of the kth order cumulant, that is:

$$S_{kx}(\omega_1,\cdots,\omega_{k-1}) = \sum_{\tau_1=-\infty}^{\infty}\cdots\sum_{\tau_{k-1}=-\infty}^{\infty} c_{kx}(\tau_1,\cdots,\tau_{k-1})\exp[-j\sum_{i=1}^{k-1}\omega_i\tau_i]\,. \tag{3}$$

In particular, the third order spectrum $S_{3x}(\omega_1,\omega_2)$ is called bispectrum. We usually denote it as $B_x(\omega_1,\omega_2)$. Bispectrum has two frequency indices ω_1, ω_2 and is complex-valued. Usually, only the amplitude value of a bispectrum is considered.

The bispectrum possesses some attractive properties. For a Gaussian random process or variable, since the high order cumulant $c_k(k>2)$ is equal to zero, its bispectrum is also zero. Due to the overlap capacity of cumulant, the bispectrum also has the overlap capacity; that is, the bispectrum of a combined signal is equal to the summation of each original signal's bispectrum. By making a synthetic consideration, we easily find that the bispectrum analysis is very suitable for restraining Gaussian noise in signal processing even in low signal-to-noise ratio case, while in most of the applications, noise can be considered to obey a Gaussian distribution.

In practice, the direct calculation method is used to estimate the bispectrum. The algorithm of the direct calculation method can be seen in Ref. [6].

4 Application Example

4.1 Analysis Object

In resent years, there have been various global, vibration-based damage detection methods studied by researchers. These techniques have been applied to many structures in various environments, making it difficult to compare the advantages and disadvantages of each technique. To compare and contrast the pros and cons of these techniques, a task group of the dynamics committee of the American Society of Civil Engineering (ASCE) joined with the International Association for Structural Control (IASC) to develop a benchmark structure in structural health monitoring <http:// wusceel.cive.wustl.edu/asce.shm>.

(1) Structure overview: The structure (Fig. 3) used in the benchmark study is a four-story, two-bay by two-bay steel frame scale model structure with a total height of 3.6m and a plane of 2.5m×2.5 m, which is a diagram of analytical model of the structure. Each story is composed by 9 columns of 0.9 m each in height, 12 beams of 1.25m each in length and 4 floor slabs (one per bay).

(2) FE Model: 120DOF finite element model (Fig. 4) on this structure is developed, which requires floor nodes to have the same horizontal translation and in-plane rotation. The columns and floor beams are modeled as Euler-Bernoulli beams.

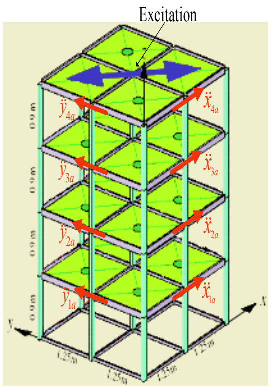


Fig. 3. Diagram of analytical model

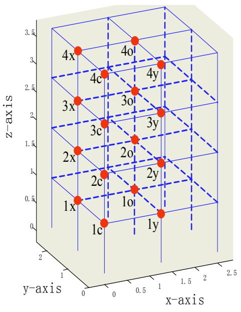


Fig. 4. FE model and joint damage positions

(3) Excitation: It is applied on the fourth floor in x-y direction (Fig. 3), and is modeled as filtered Gaussian white noise (Gaussian white noise passed through a low-pass Butterworth filter with a 100 Hz cutoff) with root mean square value 150N.

(4) Measurement: There are 16 acceleration measurement points in the structure, and $\ddot{x}_{ia}, \ddot{y}_{ia}\,(i=1,2,3,4)$ are shown in Fig. 3 for x- and y-direction measurement. $\ddot{x}_{ib}, \ddot{y}_{ib}$ $(i=1,2,3,4)$ are in the symmetry positions about $\ddot{x}_{ia}, \ddot{y}_{ia}$ and not shown in Fig. 3 for clarity. Sampling frequency is 1k Hz.

(5) Damage simulation: Bolted connections are popular in steel structures due to economy and simplicity in fabrication. However, these beam-to-column connections are more susceptible to damage than other parts of the structure. When these bolted connections are damaged such as bolt preload loss, the joint can't translate the rotational deformation, that is, the damaged joints change from rigid joints to hinged joints, which is one of the critical factors in many structural collapses. In this study, one or two damaged joints are assumed to be at one of the positions marked in solid circles in Fig. 4.

4.2 Feature Extraction

In this research work, traditional MAC (modal assurance criterion) method [1] was also used as a comparison with the bispectrum feature extraction method. Here we employ $\text{MAC}_i\ (i=1,2,3,4)$, which comes from the first four modes, and the first ten natural frequencies as the input training vector for the LVQ neural network.

As mentioned in section 3, Fig. 5 gives the 3-D plot of an amplitude bispectrum from the measurement point $\ddot{x}_{1a}$. Because the bispectrum is two-dimensional function and has some symmetric properties in the $\omega_1 \sim \omega_2$ plane, many types of 1-D slices or plane projections are used for analysis, including radial, vertical, horizontal or diagonal slices. The $\omega_1 = \omega_2 \geq 0$ diagonal slice of the amplitude bispectrum is determined as the basis for feature extraction. As shown in Fig. 6, the maximum value of the diagonal slice obtained from each measuring point is employed to construct the input training vector for the LVQ neural network.

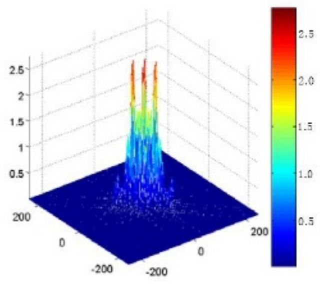


Fig. 5. 3-D plot of a bispectrum example

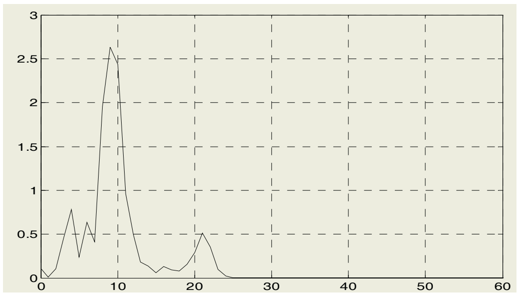


Fig. 6. Diagonal slice of amplitude bispectrum

4.3 Result of Analysis

According to the above discussion, the two cases below are studied:

(1) One damaged joint occurs at positions 1x,2x,3x,4x in the floor beam along x-direction respectively and at positions 1y,2y,3y,4y in the floor beam along y-direction respectively

The eight damage simulation patterns described above are used to evaluate the early damage identification capability of bispectrum-network combination method and compare with the MAC-network method. We set up the LVQ neural network using the following parameters:

Bispectrum method: $I_{16} - C_{32} - O_8$, MAC method: $I_{14} - C_{32} - O_8$, where *I*, *C* and *O* represent input, competitive and output layers respectively, and the subscripts of them mean the number of nodes (neurons).

Table 1. Test successful rate of bispectrum-network combination method under different SNRs

SNR	1x	1y	2x	2y	3x	3y	4x	4y
30dB	100%	100%	100%	100%	100%	100%	100%	100%
20dB	100%	100%	100%	100%	100%	100%	100%	100%
10dB	88%	96%	66%	92%	60%	92%	72%	100%

Table 1, 2 give the analysis results by bispectrum-network combination method and MAC-network combination method under different SNRs (Signal-to-Noise Ratio). 50 measured data sets under SNR = 30 dB for each damage pattern are used to train the LVQ neural network and another 50 data sets under SNR = 30, 20, 10 dB for each pattern are applied to test the network.

We can easily find that bispectrum method shows a better performance than MAC. In particular, when SNR is higher than 20 dB, the identification based on the bispectrum-network combination method shows a perfect score. Another result can be obtained that the joint damage in the floor beam along y-direction is more easily detected than the joint damage in beam along x-direction by using the bispectrum-network combination method, which is caused by the asymmetric moment of inertia of column and floor beam, and the joint damage has more influence on the weak direction of the moment of inertia.

Table 2. Test successful rate of MAC-network combination method under different SNRs

SNR	1x	1y	2x	2y	3x	3y	4x	4y
30dB	100%	96%	90%	96%	92%	98%	96%	100%
20dB	78%	76%	66%	62%	56%	70%	68%	90%
10dB	62%	40%	12%	2%	8%	8%	2%	60%

(2) Two damaged joints occur at positions 1c,2c,3c,4c,1o,2o,3o,4o in the floor beams along x- and y-direction.

Two damaged joints case is also studied as the more damage severity status of the structure compared with the one-damaged-joint case. The eight damage simulation patterns described above are used to evaluate the effect of the bispectrum-network combination method and the MAC-network combination method is also calculated for comparison. The LVQ neural network setup is the same with the one-damaged-joint case. The results of bispectrum and MAC method show better performance than the one joint damage case, which agrees the common-sense reasoning, and better identification results are also obtained by using the bispectrum-network combination method in comparison with the MAC method.

5 Conclusions

In this paper, a new approach for the global-based structure damage detection is proposed, which is based on the combination of bispectrum feature extraction technique and learning vector quantization (LVQ) neural network identification method. The purpose of introducing the bispectrum analysis and neural network combination method is to enhance the performance of damage location in a noise-contaminated environment. In the one-damaged-joint case, we can easily find that the bispectrum-network combination method always shows a better performance than the MAC-network combination method in various Signal-to-Noise Ratios. In particular, when SNR is higher than 20 dB, identification result based on the bispectrum-network combination method shows a perfect score.

Acknowledgement

Support from National Natural Science Foundation of China (No.10176014 and No. 50335030) is gratefully acknowledged.

References

1. Doebling, S.W. , Farrar C.R., Prime M.B.: A Summary Review of Vibration-based Damage Identification Methods. Shock and Vibration Digest, **30** (1998) 91-105
2. Sohn, H., Farrar, C.R.: Damage Diagnosis Using Time Series Analysis of Vibration Signals. Smart Materials and Structures, **10** (2001) 1-6
3. Hera, A., Hou, Z.K.: Application of Wavelet Approach for ASCE Structure Health Monitoring Benchmark Studies. J. of Eng. Mech., 1**30** (2004) 96-104
4. Chen, J., et. al.: A Bispectrum Feature Extraction Enhanced Structure Damage Detection Approach. JSME Intl. J., Series C, **45** (2002) 121-126
5. Sohn, H., et. al.: A Review of Structural Health Monitoring Literature: 1996-2001, Los Alamos National Laboratory Report, LA-13976-MS (2003)
6. Zhang, X.D.: The Modern Signal Processing, (in Chinese), Tsinghua University Press, Beijing (1995)

Fault Detection for Plasma Etching Processes Using RBF Neural Networks

Yaw-Jen Chang

Department of Mechanical Engineering, Chung Yuan Christian University
200, Chung Pei Rd., Chung Li, Taiwan 32023, China
justin@cycu.edu.tw

Abstract. As the semiconductor industry moves toward ULSI era, stringent and robust fault detection technique becomes an essential requirement. Most of the semiconductor processes have nonlinear dynamics and exhibit inevitable steady drift in nature, traditional statistical process control (SPC) must be used together with the time series model in order to detect any nonrandom departure from the desired target. However, it is difficult to create the time series model and sometimes found not tolerant to non-stationary variation. To overcome this difficulty, a fault detection technique using radial basis function (RBF) neural network was developed to learn the characteristics of process variations. It is adaptive and robust for processes subject to tolerable drift and for varied process setpoints. Therefore, equipment malfunctions and/or faults can be detected and the false alarms can be avoided.

1 Introduction

As the semiconductor industry strides toward the ULSI era, not only the device geometry continues to shrink but also the fabrication processes become more complex. Most of the semiconductor processes have nonlinear dynamics due to the physical and/or chemical reactions. Moreover, many of them exhibit inevitable steady drift in nature because of the gradual wearing-out phenomena or build-up of material on the components of the tools. In order to increase the throughput and to reduce the scraps, stringent and robust fault detection technique is obviously an essential requirement for promoting better quality control and higher overall equipment efficiency (OEE).

Currently, statistical process control (SPC) is the most commonly used method for the monitoring of manufacturing processes in the semiconductor industry. The implementation of SPC is based on the assumption that the data must be identically, independently and normally distributed (IIND). Any nonrandom variation away from the desired target can be then detected using control charts.

However, the process drift explains that the data collected from tools are generally non-stationary, in addition, auto-correlated or cross-correlated. SPC method cannot be applied directly on these data. The process dependencies must be modeled and filtered out with appropriate time series models so that the residual is IIND for SPC implementation [1],[2].

The major motivation of this research is to develop a simple fault detection system that captures the dependencies in the process by learning the characteristics of process variation using radial basis function (RBF) neural networks. Equipment malfunction and/or the faults can thus be detected and the false alarms can be avoided.

This paper is organized as follows: Section II presents a brief introduction to the plasma etching process. Section III addresses the design of RBF fault detection system. Section IV discusses the simulation and experimental results with practical implementation in a plasma etcher.

2 Plasma Etch

Etch is one of the most critical steps in the IC fabrication processes for replicating the designed patterns on the wafer surface. This process selectively removes unwanted material from the wafer to form the circuit patterns that were defined by photolithography. Since the last two decades, plasma etch has become the primary etching method for obtaining anisotropic profile due to the shrink of feature size. There are also many advantages such as good CD control and better uniformity within wafer, from wafer-to-wafer, and from lot-to-lot.

The mechanism of plasma etch is that the gaseous etchants is ionized in the plasma to generate the reactive free radicals which diffuse to the surface of the material and react chemically with the material to be etched. Once material is removed by etch, it is difficult to rework and most likely becomes scraps.

Many parameters, e.g. gas flow, power, and pressure, are involved in the control of plasma etch. Some of them are mutually implicative each other. Therefore, a robust fault detection system is absolutely necessary.

3 RBF Fault Detection System

There are several kinds of techniques available for fault detection. For instance, it is not surprising that the model-based method is popularly used in the industry [3],[4]. Once the model is precisely built-up, any system fault would be difficult to flee from the detector. However, the performance of the model-based approach depends strongly on the accuracy of the models. In general, the mathematical models are simplified under a satisfactory assumption, especially for nonlinear systems, linearization around an operating point is usually necessary. Model accuracy is then sacrificed. In addition, models are mostly difficult to change afterward.

Since the semiconductor manufacturing usually suffers from a high level of nonlinearity, in recent years artificial neural networks have evoked great interest in the areas of process modeling and fault detection because of their ability to learn complex nonlinear functions [5],[6]. By means of continuously learning from the training samples, a neural network renews its weights and, therefore, memorizes the correct responses. It is able to overcome the shortcomings of model-based techniques because an ANN does not require the detailed knowledge about the system for establishing its model.

In this study, radial basis function (RBF) neural network was adopted as the fault detection engine. RBF network was proposed by Broomhead and Lowe. This neural network possesses good generalization capability with minimal network structure so that it offers a variety of applications in the curve fitting (approximation) problems and suit the requirements of fault detection for the complicated systems.

The RBF network consists of three layers including input, hidden, and output layer. Differing from the neural networks with sigmoidal activation functions, it utilizes Gaussian functions in the hidden layer. Thus, the training procedure of RBF neural network requires a learning algorithm to determine both the center and the width of each Gaussian function.

3.1 Design of RBF Fault Detection System

Let the time series $S = \{v[n] : 1 \le n \le N\}$ represent a collection of N observations sampled in discrete time fashion $n = 1, 2, \cdots, N$ with sampling interval T. In order to learn the correct characteristics of process variation, it is very important that the time series must include enough data for extracting out complete pattern, for instance, the observations have to cover at least one period of cyclical variation. An analysis window [7], illustrated in figure 1, moving forward at each time increment T was adopted for tracing dynamic data. The right side of each window corresponds to the current time n. Assume that each window has length d and contains d+1 consecutive observations from S, that is, $v[n]$, …, $v[n - d]$. Define

$$x_n^{n-k} = \frac{v[n] - v[n-k]}{kT} . \tag{1}$$

where $k \le d$ and is a positive integer. Thus, for each window a training set $(x_{n-d+k}^{n-d}, \cdots, x_{n-1}^{n-k-1}, x_n^{n-k}, x_{n+1}^{n-k+1})$ of (d-k+2)-tuples is extracted from S, of which the first d-k+1 components represent network inputs and the last component represents the desired output.

The new training data derived from time series S render RBF network useful for tracing drift rate of every time instant and forecasting the next drift behavior. There is a tradeoff between precision of pattern learning and computation burden.

3.2 Control Chart

Since the control chart is a simple and effective graphical representation of the process status, it serves as a fundamental vehicle in SPC for monitoring variations. In this study, the concept of control chart was also employed for RBF fault detection system. When implementing this RBF network for fault detection, the comparison of the estimated output $\hat{y}_n$ to the desired output y_n yields a mean square error σ calculated as

$$\sigma = \sqrt{\frac{1}{(N-1)} \sum_{1}^{N} (y_i - \hat{y}_i)^2} . \tag{2}$$

where N is the number of experiments. Therefore, the upper and lower control limits was constructed to be $\pm 3\sigma$, respectively. Any variation beyond the control limits is considered to be a fault.

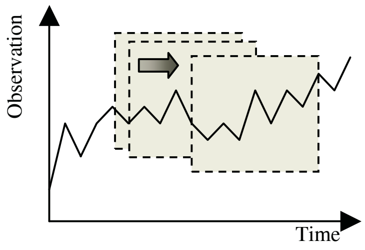


Fig. 1. Analysis window for RBF fault detection system

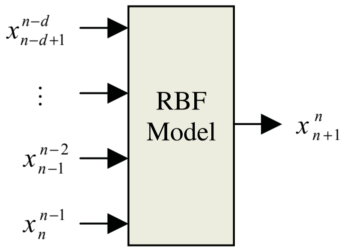


Fig. 2. Inputs/outputs of RBF neural network

4 Simulation and Experimental Results

This proposed fault detection system is designed to learn the characteristics of the process variation and thus has three features: (1) It endures inevitable steady drift existing in many semiconductor manufacturing processes, the false alarms can be avoided when the process is still under the tolerance limit. (2) It is adaptive to different process setpoints. (3) It is able to identify abnormal process shifts.

Before proceeding to the practical implementation of RBF fault detection system for the plasma etch, three simulations for the above mentioned cases were carried out. The first case was to examine the capability in detecting the process shifts. The solid line in figure 3 describes the normal process subject to the white noise, while the dotted line explains the process subject to an abnormal disturbance. The RBF fault detector captured the fault quickly. Figure 4 depicts the simulation result of the second case when the process has inevitable drift, provided that the drift is still under the tolerance limit. No false alarm was issued.

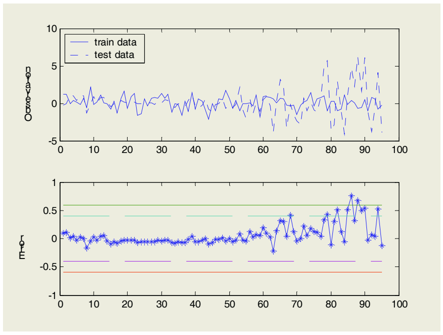


Fig. 3. RBF fault detector for process subject to abnormal disturbance

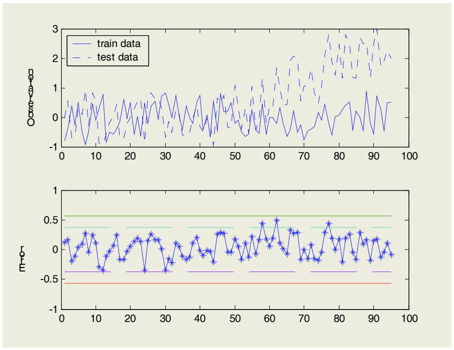


Fig. 4. RBF fault detection system for process subject to tolerable drift

Furthermore, each process tool in the IC foundry fab may execute several process recipes per day. Different recipes have variant parameter setpoints and are regarded as the sudden jumps when changing the process. The false alarm would be then involved. However, a robust fault detector must have the capability to realize the above two situations and to identify any irregular shifts. Thus, the third case was designed to test the performance of RBF detector under a varied manufacturing environment. Figure 5 shows that the RBF detector is an adaptive and robust solution.

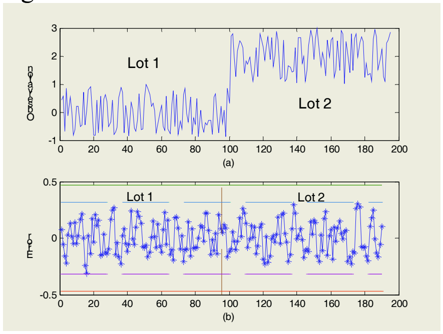


Fig. 5. RBF fault detector for varied process sepoints

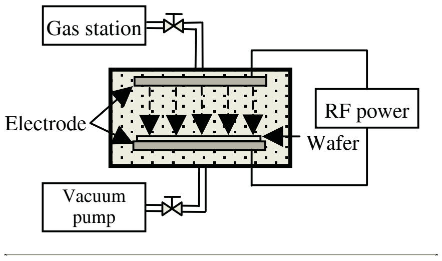


Fig. 6. Schematic of a typical plasma etching system

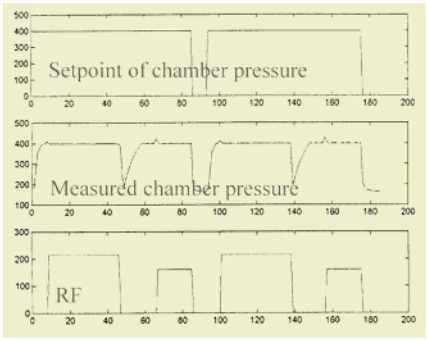


Fig. 7. Measurements of Lam 490 plasma etcher

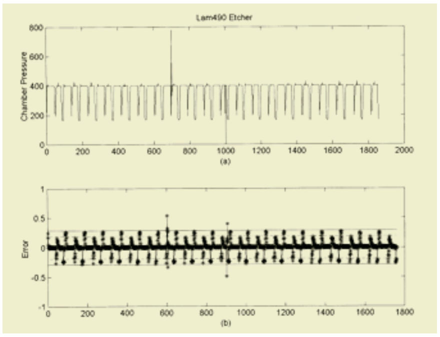


Fig. 8. RBF fault detection system for plasma etching process

In this study, the fault detection system was implemented to a Lam 490 plasma etcher. The basic components of a typical plasma etching system, depicted in Figure 6, include: (1) the reaction chamber where the etching process takes place; (2) an RF power supply to create the plasma; (3) the pumping system for establishing and maintaining the reduced pressure; and (4) the gas supply system to meter and control the flow of reactant gases. A wide range of reactant chemistries is used, for example, silicon nitride (Si_3N_4) is commonly etched by carbon tetrafluoride (CF_4) that is mixed with O_2 and N_2. The RF power is utilized to ionize the gases and, thus, produce chemically reactive species which can react with the material to be etched. During the plasma etching process, the chamber pressure must be precisely controlled.

Therefore, the pressure in the etching chamber of Lam 490 was investigated. Tool data collected through SECS were found that the pressure measurements were influenced by the switching of RF power. These two parameters revealed a level of highly cross-correlated relationship which is shown schematically in Figure 7. In this case it is obviously difficult to establish the model of pressure. Furthermore, it is also perplexed to implement SPC to monitor the pressure.

As illustrated in figure 8, without the effort for establishing the model, however, RBF fault detection system was proven to have good performance in capturing equipment malfunctions.

5 Conclusions

Most of the semiconductor processes have nonlinear dynamics and exhibit inevitable steady drift in nature, RBF fault detection system with moving analysis window is able to learn the characteristics of the process variation. It is adaptive and robust for processes subject to tolerable drift and for varied process setpoints. Therefore, equipment malfunctions and/or faults can be detected and the false alarms can be avoided.

References

1. Spanos, C.J.: Statistical Process Control in Semiconductor Manufacturing. Proc. of IEEE, **80** (1992) 819-830
2. Spanos, C.J., Guo, H.F., Miller, A., Levine-Parrill, J.: Real-Time Statistical Process Control Using Tool Data. IEEE Trans. on Semiconductor Manufacturing, **5** (1992)
3. Chen, J., Patton, R.J.: Robust Model-based Fault Diagnosis for Dynamic Systems. Kluwer Academic Publishers, (1999)
4. Chen, M.H., Lee, H.S., Lin, S.Y., Liu, C.H., Lee, W.Y., Tsai, C.H.: Fault Detection and Isolation for Plasma Etching Using Model-Based Approach. IEEE/SEMI Advanced Manufacturing Conf., (2003) 208-214
5. Baker, M.D., Himmel, C.D., May, G.S.: Time Series Modeling of Reactive Ion Etching Using Neural Networks. IEEE Trans. Semiconductor Manufacturing, **8** (1995) 62-71
6. Zhang, B., May, G.S.: Towards Real-Time Fault Identification in Plasma Etching Using Neural Networks. ASMC/IEEE, (1998) 61-65
7. Maki, Y., Loparo, K.A.: A Neural-Network Approach to Fault Detection and Diagnosis in Industrial Processes. IEEE Trans. Control Systems Technology, **5** (1997) 529-541

Detecting Sensor Faults for a Chemical Reactor Rig via Adaptive Neural Network Model

Ding-Li Yu[1] and Dingwen Yu[2]

[1] Control Systems Research Group, School of Engineering
Liverpool John Moores University, Byrom Street, Liverpool L3 3AF, UK
d.yu@livjm.ac.uk
[2] Dept. of Automation, Northeast University at Qinhuangdao, China

Abstract. An adaptive neural network model based approach to sensor fault detection is proposed for multivariable chemical processes. The neural model is used to predict process output for multi-step ahead with the prediction error used as the residual, while the model is on-line updated to capture dynamics change. The recursive orthogonal least squires algorithm (ROLS) is used to adapt a radial basis function (RBF) model to reduce the effects of data ill conditioning. Two error indices are developed to stop the on-line updating of the neural model and its corresponding threshold is used to distinguish the fault effect from model uncertainty. The proposed approach is evaluated in a three-input three-output chemical reactor rig with three simulated sensor faults. The effectiveness of the method and the applicability of the method to real industrial processes are demonstrated.

1 Introduction

The importance of fault detection and isolation (FDI) for dynamic systems, especially the safe-critical systems and cost-critical processes, is more and more realized in recent years [1]. The effects of faults caused by the physical change of plant components or actuators can be detected by observer-based methods. However, these methods are insensitive to the sensor faults, because when the residual is estimated, the sensor faults have been included in the input-output data. When a theoretical model can be derived for a plant and the model parameters are constant and can be accurately estimated, the error between model output and the sensor output can be used as the residual to indicate the fault occurrence when the model is used on-line to monitor the system. When the systems are time varying or without an accurate model, sensor faults will be difficult to detect by analytical redundancy and redundant sensors have to be used.

In this paper, we propose a new idea to detect sensor faults for slow time varying processes without mathematical models. The idea is to model the process using a neural network. The network model is on-line updating to capture the slow dynamics change and therefore its accuracy can be maintained. In every sample period, the model is used to do a multi-step ahead prediction of the process output, while the predicted output for the current sample instant is compared with the real process output and the error is used as the residual. As the predicted output for the current sample instant is based on the data multi-step before the current sample, it is not affected by the fault occurred currently. Therefore, the sensor faults can be detected. Two conditions are developed for decision-making of fault occurrence. As soon as a sensor fault is detected, on-line updating of the neural model will stop. In this way, the input-output data that is contaminated by the sensor fault will not be used to train the model. The idea is implemented by a RBF network and the on-line updating of the network weights are achieved using the ROLS algorithm [3]. The ROLS algorithm is used to reduce the effects of data ill-conditioning on the modelling error.

This novel method is applied to a three-input three-output chemical reactor rig to detect sensor faults. Three sensor faults are superimposed on the process real data and are on-line detected. The detection results demonstrated in the paper indicate the feasibility of the proposed method and the applicability of the method to real multivariable processes.

2 The Chemical Reactor Rig

The reactor used in this research is a pilot system established in the laboratory to generally represent the dynamic behaviour of real chemical processes in industry. The schematic of the chemical reactor is shown in Fig.1. It consists of a continuously stirred tank (15 litres) to which the chemical solutions, NH_4OH , CH_3COOH and Na_2SO_3 , and air are added. The liquid level in the tank is maintained at a pre-specified constant level by an outflow pump system. The concentrations of all different kinds of solution are constant. The flow rates of solutions CH_3COOH and Na_2SO_3 are constant, while the flow rate of NH_4OH and the air are adjustable by a servo-pump and a mass flow meter to regulate the pH and dissolved oxygen (pO_2) in the liquid, respectively. The tank is also equipped with an electric heating system to adjust the liquid temperature. The liquid in the tank is stirred continuously to make sure the pH, the dissolved oxygen and the temperature are consistent throughout the tank. All the three variables are measured and controlled by a personal computer.

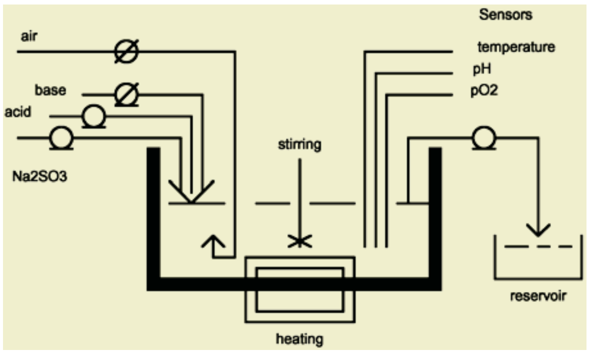


Fig. 1. The chemical reactor process

It has been shown in the experiments that the coupling between variables is very significant. The rate of absorption of oxygen into the liquid and the reaction of the sodium sulphite, for example, significantly depend on the liquid temperature. So, the process is non-linear, time varying, multivariable and without a known mathematical model. For further details of the rig see [4]. Process input and output are chosen as

$$u = [Q \quad f_b \quad f_a]^T, \quad y = [T \quad pH \quad pO_2]^T \tag{1}$$

where Q, f_b and f_a denote the heating power, the flow rate of alkaline and the flow rate of air respectively.

3 Adaptive RBF Model

The process can be represented by the multivariable NARX model of the following form,

$$y(t) = f(y(t-1), \cdots, y(t-n_y), u(t-1-d), \cdots, u(t-n_u-d)) + e(t)\,. \tag{2}$$

are the process output, input and noise respectively; p and m are the number of outputs and inputs respectively; n_y and n_u are the maximum lags in the outputs and inputs respectively; d is the maximum time delay in the inputs; and $f(*)$ is a vector-valued, non-linear function. When neural networks are used to model the process, the measurements of the process at different sample times can be taken as the input of the network, while the network implements the non-linear transformation $f(*)$ in (2).

A RBF network is adopted to model the chemical reactor as it has the advantages over the multi-layer perceptron (MLP) network with short training time and converging to the global minimum when the least squares (LS) type algorithms are used. The RBF network performs a non-linear mapping $x \in \Re^n \to \hat{y} \in \Re^p$ via the transformation,

$$\hat{y} = W^T \phi\,. \tag{3}$$

$$\phi_i = e^{-\frac{\|x-c_i\|^2}{\sigma_i^2}}, \quad i = 1, \cdots, n_h\,. \tag{4}$$

where $\hat{y}$ and x are the network output and input vectors respectively, $\phi \in \Re^{n_h}$ denotes the output vector of the non-linear basis functions in the hidden layer with ϕ_i the ith element of ϕ, $W \in \Re^{n_h \times m}$ is the weight matrix with element wij denoting the weight connecting the ith hidden node output to the jth network output, $c_i \in \Re^n$ denotes the ith centre vector, σ is the width of the Gaussian function and n_h is the number of nodes in the hidden layer. In developing a neural model for a non-linear system, we need to determine the model structure in terms of choosing the input vector x, determining the number of hidden layer nodes. Then, choose centres and widths, and train the weight matrix. All of these depend on the process dynamics and based on the input-output data, and will be addressed in follows.

In this application, the network input vector is composed of the past values of the model output and system input. The model orders and time-delays are chosen using the method [4], resulting in a network input vector,

$$x(t) = [T_r(t-1), pH(t-1), pO_2(t-1), T_r(t-2), pH(t-2), pO_2(t-2), Q(t-22), f_b(t-1), f_a(t-1)]^T\,. \tag{5}$$

Before the training, the RBF network centres were chosen using the standard *k*-means clustering method and the width is chosen an appropriate constant. The weighting matrix of a RBF network can be updated using a LS algorithm since the weights are linearly related to the output. For the training data from a real system, ill-conditioning can appear in the data set and causes a reduction in the modelling accuracy. In this paper a ROLS algorithm for MIMO systems is used as a numerically robust method for determining the weights. This algorithm is used on-line and is briefly described here. The least squares problem is formed as follows. Considering (3) for a set of *N* input-output training data, we have

$$Y = \hat{Y} + E = \Phi W + E \,. \tag{6}$$

where $Y \in \Re^{N\times p}$ is the desired output matrix, $\hat{Y} \in \Re^{N\times p}$ is the neural network output matrix, $\Phi \in \Re^{N\times n_h}$ is the hidden layer output matrix, $E \in \Re^{N\times p}$ is the error matrix. Equation (6) can be solved for $W(t)$ using the recursive MIMO least squares algorithm to minimise the following time-varying cost function,

$$J(t) = \left\| \begin{bmatrix} \lambda Y(t-1) \\ \cdots \\ y^T(t) \end{bmatrix} - \begin{bmatrix} \lambda \Phi(t-1) \\ \cdots \\ \phi^T(t) \end{bmatrix} W(t) \right\|_F \,. \tag{7}$$

By orthogonal decomposition of (7) an optimal solution of $W(t)$ can be solved from

$$R(t)W(t) = \widehat{Y}(t) \,. \tag{8}$$

and leaves the residual at stage t as $\left\| \eta^T(t) \right\|_F$. The decomposition can be achieved efficiently by applying Givens rotations to the augmented matrix to obtain the following transformation [2],

$$\begin{bmatrix} \lambda R(t-1) & \lambda \widehat{Y}(t-1) \\ \phi^T(t) & y^T(t) \end{bmatrix} \rightarrow \begin{bmatrix} R(t) & \widehat{Y}(t) \\ 0 & \eta^T(t) \end{bmatrix} . \tag{9}$$

The procedure of the ROLS algorithm is therefore the following: at stage *t*, calculate $R(t)$ and $\widehat{Y}(t)$ according to (9), then solve $W(t)$ in (8). Initial values for $R(t)$ and $\widehat{Y}(t)$ can be assigned as $R(0) = \alpha I$ and $\widehat{Y}(0) = 0$, where α is a small positive number.

4 Fault Detection

The developed RBF model is used on-line to predict the process output for K-step ahead, i.e. use the model repeatedly for K times and with the predicted output to replace the process output in the network input vector (4) after the first time. The residual is designed as the difference between the real system output and the neural model output:

$$\varepsilon(t) = y(t) - \hat{y}_K(t) \,. \tag{10}$$

The RBFN model is trained on-line to follow any process dynamics change. When a fault is detected the on-line training will stop so that the fault will not be learned. Occurrence of a fault is determined by checking two conditions. The first condition is the mean squared prediction error for M previous samples,

$$\frac{1}{M}\sum_{i=t-M+i}^{M}\varepsilon^{T}(i)\varepsilon(i) > \xi_1 . \tag{11}$$

The second condition is the squared norm of network input change,

$$[x(t)-x(t-1)]^{T}[x(t)-x(t-1)] > \xi_2 . \tag{12}$$

Here ξ_1, ξ_2 are the pre-specified thresholds that are determined in the experiment. When the error indices are found greater than the tolerances, the on-line training stops and the sensor fault is detected. Fault isolation for different sensors is achieved by checking residual for individual output.

5 Fault Detection Results

The developed method is applied to the chemical reactor rig when it is under a closed-loop control to collect training data. The process sample time for all three variables is selected to be 10 sec. The three sensor faults were simulated to occur during sample instants t=1400-1410 with a 20% change superimposed on the sensor output. The RBF network used is as described in (3) and (4), with 9 inputs as in (5), 18 neurons in the hidden layer and 3 outputs. The centres were chosen before on-line use by the *K*-means clustering algorithm. The forgetting factor in the ROLS algorithm was chosen as $\lambda = 0.99$ and the summed sample number in equation (11) was chosen as *M*=5. The tolerance is calculated by $\xi_1 = 1.2^2 p$ and $\xi_2 = 1.5^2 n$ for the normalised data. 5-step ahead predictions have been made. The model prediction and process output are shown in Fig.2, where the past model output values are reset by the process measurement if no fault is detected. This is to increase the model accuracy and this is why the model predictions are equal to process output for no fault part. The residuals are displayed in Fig.3. It can be clearly seen that the residual response to the fault has considerable amplitude and therefore the faults can be detected, although the process variables, especially the dissolved oxygen, have time-varying nature. Isolation of the sensor faults is naturally achieved as each element of the residual vector can be evaluated separately and indicates the corresponding sensor fault.

6 Conclusions

A sensor fault detection method is proposed for multivariable non-linear time-varying processes. The process dynamics change is modelled by the RBFN with on-line training using the ROLS algorithm. The fault is detected by multi-step ahead prediction error. Two conditions are proposed to determine the occurrence of the fault. The method is evaluated with a reactor rig and effectiveness is shown.

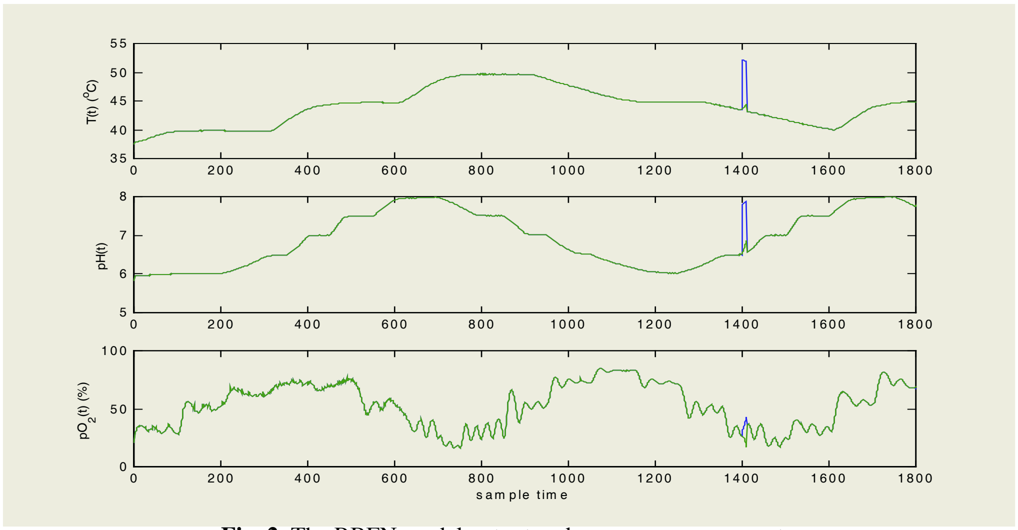


Fig. 2. The RBFN model output and process measurement

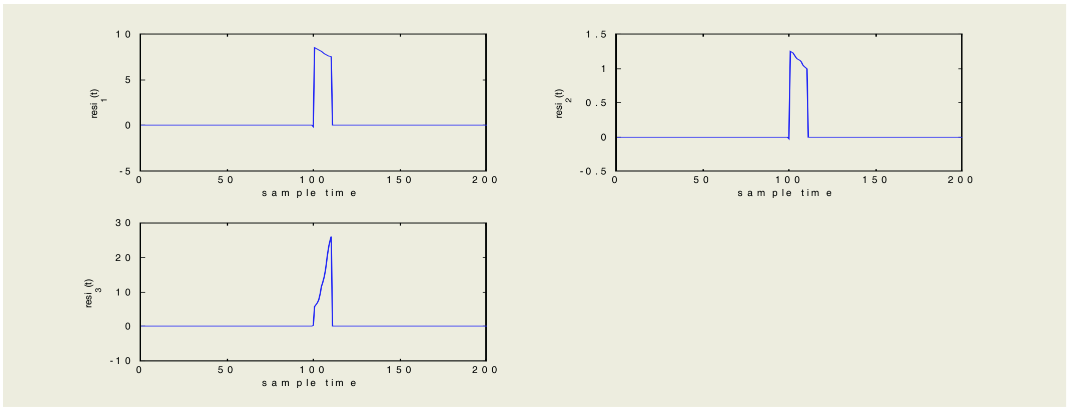


Fig. 3. Three residuals for three output variables

References

1. Patton, R.J., Chen, J. and Siew, T.M.: Fault Diagnosis in Non-Linear Dynamic Systems Via Neural Networks. Proc. IEE Int. Conf. Control'94, 21-24 March, Coventry, U.K., **2** (1994) 1346-1351
2. Bobrow, J.E. and Murray, W.: An Algorithm for RLS Identification of Parameters that Vary Quickly with Time. IEEE Trans. Automatic Control, **38** (1993) 351-354
3. Yu, D.L., Gomm, J.B. and Williams, D.: A Recursive Orthogonal Least Squares Algorithm for Training RBF Networks. Neural Processing Letters, **5** (1997) 167-176
4. Yu, D.L., Gomm, J.B. and Williams, D.: An Input Structure Selection Method for Neural Modelling and Its Application to a Chemical Process. Engineering Application of Artificial Intelligence, **13** (2000) 15-23

Optimal Actuator Fault Detection via MLP Neural Network for PDFs

Lei Guo[1,2], Yumin Zhang[1], Chengliang Liu[3], Hong Wang[2], and Chunbo Feng[1]

[1] Research Institute of Automation, Southeast University, Nanjing, Jiangshu 210096, China
[2] Control Systems Centre, Manchester University, Manchester M60 1QD, UK
l.guo@seu.edu.cn, hong.wang@manchester.ac.uk
[3] Shanghai Jiaotong University, Shanghai 210020, China

Abstract. In this paper a new type of fault detection (FD) problem is considered where the measured information is the stochastic distribution of the system output rather than its value. A multi-layer perceptron (MLP) neural network is adopted to approximate the probability density function (PDF) of the system outputs and nonlinear principal component analysis (NLPCA) is applied to reduce the model order for a lower-order model. For such a dynamic model in discrete-time context, where nonlinearities, uncertainties and time delays are included, the concerned FD problem is investigated. The measure of estimation errors, which is represented by the distances between two output PDFs, will be optimized to find the detection filter gain. Guaranteed cost detection filter are designed based on LMI formulations.

1 Introduction

The study on the design of filter-based fault detection (FD) schemes for stochastic systems, as one key research subject in quality control, has received much attention during the last two decades [1], [2]. Filters are widely used to generate residual signals to detect and estimate the fault, where Gaussian variables are concerned and the mean and/or variance of estimation errors are optimized. However, non-Gaussian variables with the asymmetric distributions exist in many cases. On the other hand, along with the development of advanced instruments and data processing techniques, in practice the measurement for filtering or control may be stochastic distributions of the system output, rather than the value of some variables. Typical examples include the retention of paper making, particle distribution, molecular weight distribution, and flame grey-level distribution processes [4], [5]. This is also different from the traditional filtering or filter-based FD results. New filtering and control methods are required to be developed by using the stochastic property.

One of the main obstacles is the formulation of the probability distributions for the stochastic output, which generally requires the solutions of partial differential equations. B-spline expansions have been used to model the dynamic output probability density functions (PDFs), which lead to a challenge on selecting the basis functions [4], [5]. Recently, the multi-layer perceptron (MLP) neural network models have been applied to the shape control problems for the output PDFs [6]. In this paper, following the development of the neural network modeling procedures for PDF control problems, MLP models are used to investigate the above new type of FD problems. It is shown that the concerned FD problem can be transformed to a deterministic nonlinear FD problem, for which we present a new detection filter scheme. Guaranteed cost optimization performance is applied and the solvability condition is provided in terms of linear matrix inequality (LMI) formulations.

2 Main Results

2.1 MPL Neural Network Model for PDF Approximations

Consider a discrete-time dynamic stochastic system where $u(k) \in R^m$ is the input, $y(k) \in [a,b]$ is the output of the concerned. F is supposed to be an actuator fault to be diagnosed. The output PDF is denoted by $\gamma(z,u(k),F)$, which is supposed to be continuous or piecewise continuous. The following result can be seen from [6].

Lemma 1: $\gamma(z,u(k),F)$ can be approximated by the MPL Neural Network Model as

$$\gamma(z,u,F) = P(W(u,F))\big(L(\Theta(u,F))z + b(u,F)\big) + d(u,F) + \omega_0 . \tag{1}$$

where $P(x) = \begin{cases} x, x \geq 0 \\ 0, x < 0 \end{cases}$, and $L(x) = \dfrac{1}{1+e^{-x}}$. $W(u,F)$ and $L(\Theta(u,F))$ are the weights of the neural network, $b(u,F)$ and $d(u,F)$ are the biases of the neural network, and ω_0 is the approximation error. Define $\eta(z,u,F) = W(u,F)\big(L(\Theta(u,F))z + b(u,F)\big)$, then

$$d(u,F) = \begin{cases} \dfrac{1}{(b-a)^n}\left(1 - \int_{z\in\Omega} \eta(z,u,F)dz\right) & d(u,F) + \eta(z,u,F) \geq 0 \\ -\eta(z,u,F) & otherwise \end{cases} . \tag{2}$$

With such an MLP neural network model, the dynamic nonlinear relationship between the input and the output PDF can be established via the weightings described in (1) subject to constraint (2). Due to the constraint for PDFs, (2) should be satisfied which means that $d(u,F)$ depend on $\eta(z,u,F)$. Denoting

$$\overline{V}(k) := \begin{bmatrix} W(u,F) & \Theta(L(u,F)) & b(u,F) \end{bmatrix}^T . \tag{3}$$

the next step is to formulate the relationship between (u,F) and V, which corresponds to a further identification procedure [4-6]. For the concerned filter-based FD problem, $u(k)$ is irrelevant and will be neglected in the following arguments, and $\gamma(z,u(k),F)$ will be abbreviated by $\gamma(z,u,F)$.

Furthermore, NLPCA model can be used to reduce the model, where nonlinear transfer functions can map data from the high dimension input space to a low dimension data compression space (see *e.g.* [6]). The concrete procedures are omitted in this paper to save space. It is supposed that by NLPCA, $\overline{V}(k)$ can be reduced to $V(k)$ with a lower dimension through multiple layers by selecting proper transfer functions (see *e g.* [6]).

2.2 Nonlinear Dynamic Model for the Weightings

ω_0 represents the model uncertainty or the approximation of PDFs, which is supposed to satisfy $|\omega_0| \le \delta_0$ for all $\{z, F, k\}$, where δ_0 is assumed to be a known positive constant. Different from the linear model adopted in [4-6], in this paper we consider the following dynamic model

$$\begin{cases} x(k+1) = Ax(k) + A_d x(k-d) + Gg(x(k)) + DF(k) \\ V(k) = Ex(k) \end{cases}. \tag{4}$$

where $x(k) \in R^m$ is the unmeasured state, $V(k)$ is the output as well as the weight corresponding to (1). d is the time delay. A, A_d, G, E and D represent the known parametric matrices. $g(x(k))$ is a given nonlinear function which satisfies the following norm condition

$$\left\| g(x_1(k)) - g(x_2(k)) \right\| \le \left\| U_2(x_1(k) - x_2(k)) \right\|. \tag{5}$$

for any $x_2(k)$, $x_1(k)$ where U_2 is a known matrix.

Different from the most previous results, the information for feedback is $\gamma(z, F, k)$ at every sample time. It is noted that $\gamma(z, F, k)$ trained by MLP neural networks is a nonlinear function with a modeling error. To provide a feasible detection filter design method, Taylor expansions are further used to approximate as follows

$$\gamma(z, F, k) = B_0^T V_0 + B^T(z) Ex(k) + h(Ex(k)) b_n(z) + \omega(z, F, k). \tag{6}$$

where B_0 is a known constant vector, V_0 is a known vector as the static equilibrium point and the independent term. $B(z)$ is a known vector function of sample value z and $h(V(k))$ is an unknown nonlinear function satisfying

$$\left\| h(V_1(k)) - h(V_2(k)) \right\| \le \left\| U_1(V_1(k) - V_2(k)) \right\|. \tag{7}$$

where U_1 is also a known constant matrix. Actually, $h(V(k))$ corresponds to the first differential term related to the Taylor expansions and $b_n(z)$ is denoted as the corresponding sample value function. ω is a new modeling error which includes both ω_0 and the error resulting from Taylor expansions. It is supposed that $|\omega(z,F,k)| \le \delta$, where δ is a known constant.

2.3 New Criteria for Fault Detection by Using PDFs

In order to detect the fault based on the changes of output distributions, we construct the following vector as the measurement output and the residual signal

$$\varepsilon(k) = \int_a^b \sigma(z)\left(\gamma(z,F,k) - \hat{\gamma}(z,k)\right)dz . \tag{8}$$

where $\hat{\gamma}(z,k) = B_0V_0 + B(z)E\hat{x}(k) + h\left(E\hat{x}(k)\right)b_n(z)$ and $\sigma(z) \ge 0 (\ne 1)$ is a weight function to tune the difference described as (8). The estimation error system for the estimation error defined by $e(k) = x(k) - \hat{x}(k)$ can be described by

$$\begin{aligned} e(k+1) &= (A - L\Gamma_1)e(k) + A_d e(k-d) + G\left[g(x(k)) - g(\hat{x}(k))\right] \\ &\quad - L\Gamma_2\left[h(Ex(k)) - h\left(E\hat{x}(k)\right)\right] + DF(k) - L\rho(k) \\ &= \tilde{A}X_k + DF(k) - L\rho(k) . \end{aligned} \tag{9}$$

where $\rho(k) = \int_a^b \sigma(z)\omega(z,F,k)dz$ can be regarded as a bounded disturbance and the auxiliary matrices are denoted as

$$\begin{cases} \Gamma_1 = \int_a^b \sigma(z)B(z)Edz, \Gamma_2 = \int_a^b \sigma(z)b_n(z)dz, \tilde{A} = \begin{bmatrix} A - L\Gamma_1 & A_d & -L\Gamma_2 & G \end{bmatrix} \\ \Pi_0 := diag\left\{-P + Q + \frac{1}{\lambda_1^2}E^TU_1^TU_1E + \frac{1}{\lambda_2^2}U_2^TU_2, -Q, -\frac{1}{\lambda_1^2}I, -\frac{1}{\lambda_2^2}I\right\} \\ \tilde{\Pi}_1 = \Pi_1 + diag\{I,0,0,0\}, \Pi_1 = \Pi_0 + diag\{\eta I, \eta I, 0, 0\} \end{cases} . \tag{10}$$

X_k is an auxiliary augmented vector denoted as

$$X_k = \begin{bmatrix} e^T(k) & e^T(k-d) & h^T(Ex(k)) - h^T\left(E\hat{x}(k)\right) & g^T(x(k)) - g^T\left(\hat{x}(k)\right) \end{bmatrix}^T$$

From $|\omega(z,F,k)| \le \delta$, it can be verified that

$$\|\rho(k)\| = \left\|\int_a^b \sigma(z)\omega(z,F,k)dz\right\| \le \tilde{\delta} := \delta\left\|\int_a^b \sigma(z)dz\right\| . \tag{11}$$

It can be seen that

$$\varepsilon(k) = \Gamma_1 e(k) + \Gamma_2\left(h(Ex(k)) - h\left(E\hat{x}(k)\right)\right) + \rho(k) . \tag{12}$$

For the sample time $\{-d, -d+1, \cdots -1, 0\}$, we denote $x(k) = 0, \hat{x}(k) = 0$.

In order to detect F more sensitively, we construct the auxiliary vector $z_0(k) = e(k)$ and consider the sub-optimal guaranteed cost problem for $J_0 = \|z_0(k)\|^2$ subject to error system (9). The following result is an extensive result of Lemma 1 in [2]. For simplicity, modeling error $\omega(z, F, k)$ denoted in (6) is neglected which implies that $\rho(k)$ is assumed to 0. In the following, the proofs are omitted for brevity.

Theorem 1: If for the parameters $\lambda_i > 0\,(i = 1,2)$, there exist matrices $P > 0$, $Q > 0$, R and constant $\eta > 0$ satisfying

$$\tilde{\Pi} = \begin{bmatrix} \tilde{\Pi}_1 & \tilde{A}^T P \\ P\tilde{A} & -P \end{bmatrix} < 0 \ . \tag{13}$$

then in the absence of both F and $\Delta(k)$, error system (9) with gain $L = P^{-1}R$ is asymptotically stable and the error satisfies

$$\|e(k)\|^2 \le \tilde{\alpha}^2 := e^T(0)Pe(0) + \sum_{l=1}^{d} e^T(-l)Qe(-l) \ . \tag{14}$$

Theorem 2: For the weighting system without modeling error $\rho(k)$, if for the parameters $\lambda_i > 0\,(i = 1,2)$, there exist matrices $P > 0$, $Q > 0$, R and constant $\eta > 0$ satisfying (13), then F can be detected by the following criterion

$$\|\varepsilon(k)\| > \tilde{\beta} = \tilde{\alpha}\left(\|\Gamma_1\| + \|\Gamma_2\|\|U_1\|\|E\|\right) \ . \tag{15}$$

where $\tilde{\alpha}$ is determined by (14).

Remark 1: Different from most classical observer design methods [1,2], residual $\varepsilon(k)$ is the difference of the measured PDFs and estimated PDFs, which can be formulated as a nonlinear output of $e(k)$. If modeling error is neglected, it can be claimed that asymptotical stability of the error system can be guaranteed.

Remark 2: In Theorem 2, the threshold can be tuned by $\tilde{\alpha}$, which is further dependent on η as well as P, Q and R. This problem can be dealt with as an optimization procedure for $\tilde{\alpha}$ subjected to LMIs and H_∞ optimization technique can be applied (see [5]). This problem will be discussed in the future research.

3 Conclusions

In this paper, a new type of fault detection problem is studied. Different from the conventional FD problems, the measurement information for fault detection is the statistic property of some variables, where the concerned stochastic variables are not confined to be Gaussian. In order to provide a feasible FD framework, MLP neural networks with modeling errors are applied to approximate the output PDFs. Furthermore the NLPCA technique can be used to reduce the model order and then a nonlinear dynamic system for the weighting and bias can be established. LMI-based solvable conditions are provided to ensure the estimation error of the fault detection filter by using the guaranteed cost filtering approach.

Acknowledgements

This work is supported partially by the Leverhulm Trust, UK, the NSF of China, the China Post-Doctoral Foundation and the Jiangsu Post-Doctoral Foundation of China.

References

1. Frank, P.M., Ding, S.X.: Survey of Robust Residual Generation and Evaluation Methods in Observer-based Fault Detection Systems. Journal of Process Control, **7** (1997) 403-424
2. Gertler, J.: Fault Detection and Diagnosis in Engineering Systems. Marcel Dekker, New York (1998)
3. Guo, L.: Guaranteed Cost Control of Uncertain Discrete-time Delay Systems Using Dynamic Output Feedback. Transactions of the Institute of Measurement and Control, **24** (2002) 417-430Guo, L., Wang, H.: Fault Detection and Diagnosis for General Stochastic Systems Using B-Spline Expansions and Nonlinear Observers. Proc. of IEEE 43rd CDC (2004), USA, and to appear in IEEE Trans on CAS-I (2005)
5. Wang, H., W. Lin: Applying Observer Based FDI Techniques to Detect Faults in Dynamic and Bounded Stochastic Distributions. Int. J. Control, **73** (2000) 1424–1436
6. Wang, H.: Multivariable Output Probability Density Function Control for Non-Gaussian Stochastic Systems Using Simple MLP Neural Networks. Proc. IFAC Int. Conf. on Intelligent Control Systems and Signal Processing, Algarve, Portugal (2003) 84-89

Feature Selection and Classification of Gear Faults Using SOM

Guanglan Liao[1], Tielin Shi[1], Weihua Li[2], and Tao Huang[1]

[1] School of Mechanical Science and Engineering,
Huazhong University of Science and Technology, Wuhan, Hubei 430074, China
{g.l.liao,tlshi,taoh}@mail.hust.edu.cn
[2] College of Automotive Engineering,
South China University of Technology, Guangzhou, Guangdong 510640, China
whlee@scut.edu.cn

Abstract. Feature selection is a key issue to machine fault diagnosis. This paper presents a study that uses self-organizing maps to realize feature selection and reduce dimensionality of the raw feature space for machine faults classification. By means of evaluating the responses of every dimensional feature in SOM networks neurons weights to the input data, the feature sets having the main responses and being sensitive to pattern recognition are selected. Industrial gearbox vibration signals measured under different operating conditions are analyzed using the method. The experimental results indicate that the method selects sensitive feature sets effectively for gear faults classification and recognition, and has a good potential for application in practice.

1 Introduction

A gearbox transmission system is one of the fundamental and most important parts of machinery and employed in industrial sectors worldwide. If failures occur to any gears during operating conditions, the faulty gearbox system would result in serious damage. Therefore, the fault diagnosis of the gearbox system is crucial so as to prevent the system from malfunction that could cause damage or entire system halt. Up to now fault diagnosis of industrial gearboxes has received intensive study for several decades. There have been many investigations carried out to diagnose the performance of industrial gearboxes [1]. Among those fault diagnosis methods, pattern recognition method provides a systematic approach to acquiring knowledge from fault samples. In fact, mechanical fault diagnosis is essentially a problem of pattern recognition, in which feature selection plays an important role.

Features are any parameters extracted from the measurements through signal processing in order to enhance damage detection. Feature selection is helpful to reduce dimensionality, discard deceptive features and extract an optimal subspace from the raw feature space, and therefore is critical to the success of fault diagnosis. There are many methods for feature selection, such as genetic analysis, back-propagation network, etc. Since neural networks, which are widely applied in the field of pattern recognition now, are able to approximate almost any nonlinear functions, they are possible to model the nonlinear dynamics of gearboxes when failures occurring. Examples of successful applications of neural networks in gear faults diagnosis are abundant [2]. In this paper a study is presented that uses self-organizing maps (SOM)

[3] to realize feature selection for gear faults diagnosis. Industrial gearbox vibration signals measured under different operating conditions are analyzed using the method. The results indicate that the method selects sensitive feature sets effectively for gear faults classification and has a good potential for application in practice.

The paper is organized as follows. The approach for feature selection based on SOM networks is presented in section 2 illustrated by an example, and then verified for gear faults analysis on experimental data measured from industrial gearbox system in section 3. The investigation is concluded in section 4.

2 Feature Selection Based on SOM

The SOM network defines a mapping from a high dimensional input space onto a low dimensional (usually 2-D) array of output neurons. Locations of the output neurons in the output layer are indicative of intrinsic statistical features contained in the input vectors. Therefore, the distribution of every dimensional feature in the neurons weights will approximate that in the input data, which will do help for feature selection. In the following feature selection based on SOM networks is introduced.

Suppose the input vectors, say $\boldsymbol{p} = (p_1, p_2,\cdots, p_n)^T$, are n dimensional, and there are m output neurons in total in the SOM network arranged in a 2-D sheet with hexagonal shape. The coordinates of output neurons j fixed in the sheet are denoted by $\boldsymbol{x}_j$, j=1,2,m, and the corresponding n-dimensional synaptic weights $\boldsymbol{w}_j = (w_{j1},w_{j2},\cdots, w_{jn})^T$.

Similar to a novel SOM visualization technique [4], the Euclidean distances between the ith dimensional feature in $\boldsymbol{p}$ and neurons weights $\boldsymbol{w}_j$ are computed

$$q_{ji} = \left\| \boldsymbol{p}_i - \boldsymbol{w}_{ji} \right\| \quad j = 1, 2, \cdots, m . \tag{1}$$

Then the responses of the ith dimensional feature in neurons weights to $\boldsymbol{p}$ are obtained

$$R_{ji}(\boldsymbol{p}) = \frac{N_c(j) f(q_{ji})}{\sum_{k=1}^{m} N_c(k) f(q_{ki})} \qquad j = 1, 2, \cdots, m . \tag{2}$$

Here the responses of the ith dimensional feature to $\boldsymbol{p}$ in the whole output neurons add up to 1, c is the BMU of $\boldsymbol{p}$, and $N_c(j)$ a neighboring function decreasing with the distances between c and j. Since the responses, $R_{ji}(\boldsymbol{p})$, are related closely with the Euclidean distances between the neurons weights and $\boldsymbol{p}$, where the further the weights are away from $\boldsymbol{p}$, the less responses the neurons have, it is appropriate to choose $f(q_j)$ to be a bounded function monotonically decreasing with q_j. Usually, the decreasing exponential, sigmoid, or Lorentz functions are all suitable choices. Now the coefficients given by $R_{ji}(\boldsymbol{p})$ can be used to provide a visualization of the trained SOM results. The image points for $\boldsymbol{p}$ in the 2-D space according to the responses of the ith dimensional feature can be

$$\boldsymbol{x}_{\boldsymbol{p}}^{i} = \sum_{j=1}^{m} R_{ji}(\boldsymbol{p}) \boldsymbol{x}_j \quad i = 1, \cdots, n . \tag{3}$$

So, the sensibilities of features for data clustering can be evaluated according to the distribution of the image points, $\boldsymbol{x}_{\boldsymbol{p}}^{i}$. To analyze the results quantitatively, two parameters, scattering value in clusters and between clusters are used [5], where the former is smaller and the latter is larger, the corresponding feature is more sensitive.

Next the well-known Fisher's iris data set [6] consisting of four measurements taken from 150 iris plants were demonstrated to verify the method. Each plant was from one of three species of iris, *setosa*, *versicolor* and *virginica*. The four measurements including *sepal length*, *sepal width*, *petal length* and *petal width* from each plant were treated as points in 4-D space. A 50×50 hexagonal SOM had been applied to the data set, and the feature selection method was used. The scattering values of 4 features between clusters were 0.9669, 0.1433, 2.0184, 1.1287, and in clusters 0.3383, 0.1071, 0.2176, 0.1268 accordingly.

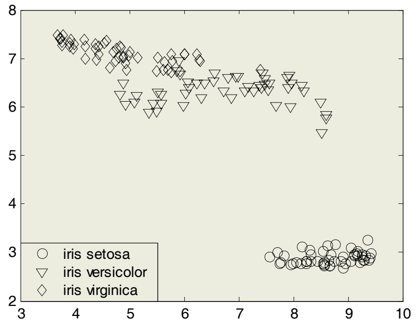


Fig. 1. The classification of iris data including the sensitive features, *sepal length*, *petal length* and *petal width*, trained with SOM and visualized with a novel technique

From the figures, it can be found that *sepal width*, although with smallest scattering value in clusters, has a scattering value between clusters too much smaller than others. Hence, it must be insensitive for data clustering of all features. Then the sensitive features including *sepal length*, *petal length* and *petal width* can be selected for further analysis. Fig. 1 shows the results of data clustering based on these 3 features trained with SOM and visualized with the novel technique [4]. It can be observed from Fig. 1 that only 3 features selected for training and classification a fairly good separation of the 3 species on the map is obtained. Therefore, the feature selection based on SOM is effective for sensitive feature selection and data clustering.

3 Gear Faults Classification

Gear fatigue experiments were conducted to verify the method. The gearbox acceleration vibration signals were anti-aliased and then sampled at 12.5 kHz. During the testing process, the gearbox underwent three stages, normal condition, a crack in one tooth root and the cracked tooth broken. It is known that the time domain of the vibration signals provides a useful feature set for gear faults diagnosis. Here only 11 time domain features, including *maximum value* (Max), *minimum value* (Min), *standard deviation* (Std), *absolute mean value* (Pro), *crest factor* (C), *impulse factor* (I), *clearance factor* (L), *root mean square value* (Rms), *kurtosis* (Kur), *skewness* (Skew) and *variance* (Var), were used as raw feature sets for gear faults analysis. There selected 75 raw vibration signals, each having 1024 samples, measured under normal, tooth cracked and tooth broken conditions of the gearbox (25 per condition) for the

Table 1. Scattering value comparison

	Scattering value between clusters	Scattering value in clusters
Max	0.498	0.069
Min	0.494	0.065
Std	0.510	0.040
Pro	0.509	0.040
C	0.270	0.154
I	0.246	0.163
L	0.279	0.151
Rms	0.510	0.040
Kur	0.342	0.103
Skew	0.206	0.123
Var	0.507	0.039

investigation, which were divided into 2 data sets: data set 1 consisted of 60 raw vibration signals (20 per condition) for training, and data set 2 consisted of the others (5 per condition) for testing.

First, 11 features mentioned above of each raw signal in data set 1 were computed and normalized. In total 60 11-D raw feature data were obtained and then fed to the SOM network with a 50×50 hexagonal structure, and the scattering values were displayed in Table 1.

From the table 6 features having large scattering values between clusters and small in clusters were selected, including Max, Min, Std, Pro, Rms and Var. 60 6-D feature data were obtained and fed to the SOM network again. The results are shown in Fig. 2a. Here the image points of gear feature data under normal, tooth cracked and tooth broken conditions are denoted by "*circle*", "*triangle*" and "*diamond*" signs, respectively. It can be seen clearly from the figure that three clusters are formed, where cluster 1 indicates the normal condition, cluster 2 indicates the gearbox condition with a cracked tooth, and cluster 3 the condition with a broken tooth. Although there may be difficult to identify the gearbox condition with fatigue cracks against the normal state from time domain and frequency domain, it can be observed from Fig.2a that three conditions of gearbox are distinguished clearly from each other according to the distribution of the image points.

Then 6 selected features of each raw signal in data set 2 were computed and normalized. 15 6-D feature data were obtained and fed to the SOM network for testing. The results are shown in Fig. 2b, where solid "*circle*", "*triangle*" and "*diamond*" represent the image points of feature data under the gearbox conditions of normal, tooth cracked and tooth broken correspondingly. As can be seen, solid "*circle*" are distributed in the region represented with "*circle*" indicative of the normal condition, solid "*triangle*" are distributed in the "*triangle*" region indicative of the gearbox condition with a cracked tooth, and solid "*diamond*" in the "*diamond*" region indicative of the tooth broken condition. Hence it is very easy to identify different gearbox conditions.

4 Conclusions

One of the most complex problems for machine faults diagnosis is to select the signal features and describe the relationship between the condition and the signal features as accurately as possible. In this paper a method for feature selection based on SOM is presented, in which the responses of every dimensional feature in neurons weights to the input data are obtained, and then the feature sets sensitive to data clustering and pattern recognition are selected accordingly. The analysis results on industrial gearbox vibration signals prove that sensitive feature sets are selected efficiently from raw feature sets, and gear faults classification and recognition are realized. Therefore, the SOM network has great potential for gear faults analysis in practice.

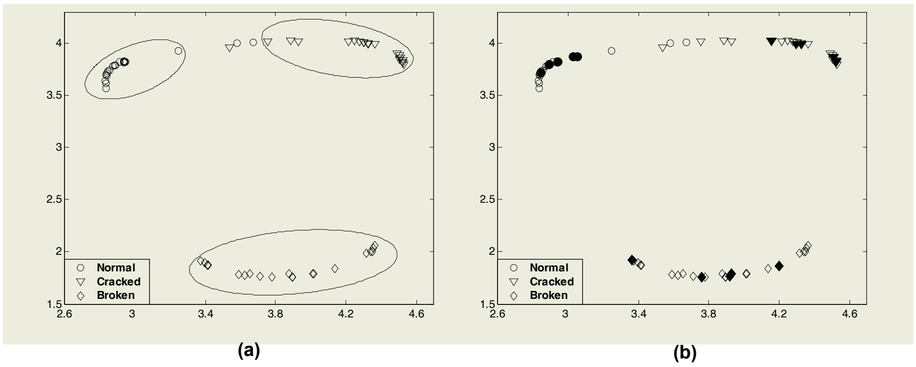


Fig. 2. Feature data of gearbox under normal, tooth cracked, and tooth broken conditions, clustered (a) and recognized (b) by SOM

Acknowledgements

The authors would like to acknowledge the financial support provided by the National Key Basic Research Special Found of China (Grant No. 2003CB716207) and the National Natural Science Foundation of China (Grant No. 50205009 and 50405033).

References

1. Ypma, A.: Learning Methods for Machine Vibration Analysis and Health Monitoring, Ph. D. Dissertation. Pattern Recognition Group, Department of Applied Physics, Delft University of Technology (2001)
2. Samanta, B., Al-Balushi, K.R., Al-Araimi, S.A.: Use of Genetic Algorithm and Artificial Neural Networks for Gear Condition Diagnostics. In: Starr, A., Rao, B.K.N (eds.): The 14th International Congress on Condition Monitoring and Diagnostic Engineering Management, Manchester (2001) 449–456
3. Kohonen, T.: Self-Organized Formation of Topologically Correct Feature Maps. Biological Cybernetics, **43** (1982) 59–69
4. Liao, G., Liu, S., Shi, T., Zhang, G.: Gearbox Condition Monitoring Using Self-Organizing Feature Maps. IMechE: Journal of Mechanical Engineering Science, **218** (2004) 119–129
5. Bian, Z., Zhang, X: Pattern Recognition. Tsinghua University Express, Beijing (2000)
6. Fisher, R.A.: The Use of Multiple Measurements in Taxonomic Problems. Annual Eugenics, **7** (1936) 178–188

Application of Fuzzy SOFM Neural Network and Rough Set Theory on Fault Diagnosis for Rotating Machinery

Dongxiang Jiang, Kai Li, Gang Zhao, and Jinhui Diao

Department of Thermal Engineering, Tsinghua University, Beijing 100084, China
jiangdx@tsinghua.edu.cn, likai03@mails.tsinghua.edu.cn

Abstract. This paper presents a new method that applies fuzzy logic, rough set theory and SOFM neural network to rotating machinery fault diagnosis. In this method, firstly, relationships between the fault causations and fault symptoms are established by fuzzy logics. Then the Rough Set Theory (RST) is applied to obtain a minimal sufficient subset of features, which is helpful to simplify the structure of neural network. Next, the 2-dimension output mapping of the standard fault samples (training samples) is obtained by a self-organizing neural network. Finally, we input some simulation samples (testing samples) and gain the reasonable conclusions by comparison between the two output mappings. Experimental results have demonstrated the effectiveness of this method and its nice prospect of applying to rotating machinery fault diagnosis.

1 Introduction

Artificial neural networks (ANN) have received wide research efforts in fault diagnostics because of its key strengths such as self-learning, nonlinear recognition, association, strong anti-noise capability and tolerance [1]. The Self-organizing Feature Map (SOFM) neural network is one group of ANN. The most distinct feature of SOFM neural network is that training is an unsupervised process. It also has some advantages such as simple structural algorithm, fast learning and lateral association, etc. However, we must address some problems during diagnosing rotating machinery by SOFM neural network:

- It is difficult to define a standard value to distinguish normal from abnormal strictly.
- When the dimension of input vector is large, the training time may be long.
- The collected data samples are normally incomplete, or the meaningful information is quite shortage due to lack of measurable parameters.
- The association ability of SOFM is limited, that is, when beyond boundary, it will associate by wrong manners and result in false diagnosis or missed diagnosis [2].

In order to address problems described above, this paper proposes a new method – Fuzzy Logic, Rough Set and SOFM neural network. First we introduce Fuzzy Logics to establish the membership matrix between fault causations and fault symptoms.

Second we eliminate redundant features through Rough Set Theory (RST) without decreasing the equality of classification and gain the reduction decision table. Third we construct a SOFM to learn for the input standard fault samples data. At last, fault diagnosis is completed by SOFM. In one word, we take fuzzy logics and rough set theory as pre-process system and SOFM neural network as information process system.

2 Data Pre-process

2.1 Fuzzy Logic Process

The rotating machinery fault diagnosis process normally includes two steps: abnormality condition identification and fault diagnosis (or fault classification). When we judge the operation condition is abnormal, it is necessary to perform a further diagnosing and find out the fault causations [3]. However, experienced engineers are often required to interpret measurement data that are frequently inconclusive. Fuzzy logic allows items to be described as having a certain membership degree in a set. Thus, they may fall into some interior range [4].

Fuzzy subsets are the range of feature parameters x_i and membership functions $\mu_{\underset{\sim}{A}}(x)$ describe a certain membership degree by $\mu_{\underset{\sim}{A}}(x_i)$ ($0 \le \mu \le 1$). The membership function may be selected as increasing Cauchy-Function as

$$\mu_{\underset{\sim}{A}}(x) = \begin{cases} 0 & x \le \alpha \\ \dfrac{k(x-\alpha)^\beta}{1+k(x-\alpha)^\beta} & x > \alpha > 0, \beta > 0 \end{cases} . \tag{1}$$

Where the factors: α, β and k is specified by experience.

Or you can select it as decreasing Cauchy-Function (or combination of them) according to different alarm manners [3]. Once the membership function is determined, the fuzzy membership matrix between fault causations and fault symptoms can be derived.

2.2 Rough Set Theory Process

Rough set theory (RST), introduced by Zdzislaw Pawlak in the early 1980s, is a new mathematical tool to deal with vagueness and uncertainty. This approach seems to be of fundamental importance to artificial intelligence and cognitive sciences, especially in the areas of machine learning, knowledge acquisition, decision analysis, knowledge discovery from databases, and pattern recognition [5]. One of the main advantages of rough set theory is that it does not need any preliminary or additional information about data. Now we present two important concepts – Reduction and Decision table.

Reduction: A reduction is a minimal sufficient subset of features RED $\subseteq$ A such that [6]:

a) R (RED) = R (A), i.e., RED produces the same classification of objects as the collection A of all features;
b) for any feature $f \in$ RED, R (RED – $\{f\}$) $\neq$ R (A), i.e., a reduction is a minimal subset with respect to the property a).

Decision table: A decision table is a two-dimension table, which describes each object by two kinds of attributes: condition attribute and decision attribute.

3 Self-organizing Feature Map [7]

The network architectures and signal processes used to construct nervous systems can roughly be divided into three categories, each based on a different philosophy. The Self-Organizing Feature Map is one of them. Compared with the most popularly used model BPNN, it has a structure more similar to humanity biology [1]. The most distinct feature is that the training is an unsupervised process. The structure of the model is shown in Fig. 3.

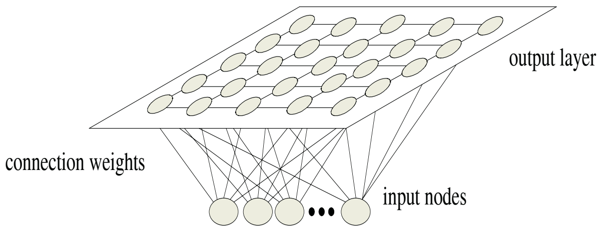


Fig. 1. Self-Organizing Neural Networks

Such model is made up of two layers – input and output: Every input neuron connects with the output ones by connection weighting vectors. The number of the input nodes is determined according to the dimensions of the input vectors and the input nodes receive input values. The output layer is a plane, which is made up of neurons arrayed in a certain way (square or hexagon, etc). The learning of the Self-organizing Feature Map neural network is different from that of BPNN, because of self-organizing. The training procedures (or algorithms) are available in reference [7].

4 Fault Diagnostic Processes of Fuzzy Logic, Rough Set and SOFM Neural Network

With the knowledge described in section 2 and 3, we can apply the new approach that combines fuzzy logic and RST with SOFM to diagnose rotating machinery as following processes:

1) Data collecting. Collect data that reflect the operating conditions of rotating machinery;
2) Fuzzy logic pre-process. Extract feature parameters. Establish membership matrix by fuzzy logic;
3) Discretization. Numerical attributes and attributes that have an ordering on them may have to be discretized. This amounts to defining a coarser view of the world, and to making quantitative data more qualitative.
4) Reducing. Compute reducts and gain the input data of SOFM neural network. Thus, we have reduced the number of NN's inputs.
5) Training. Make the SOFM to learn for input data.
6) Testing. Input test samples to be diagnosed, denote each sample's location.
7) Diagnosis results. Compare the training and testing output mappings to gain the diagnosis conclusions. Analyze the accuracy and availability of this method.

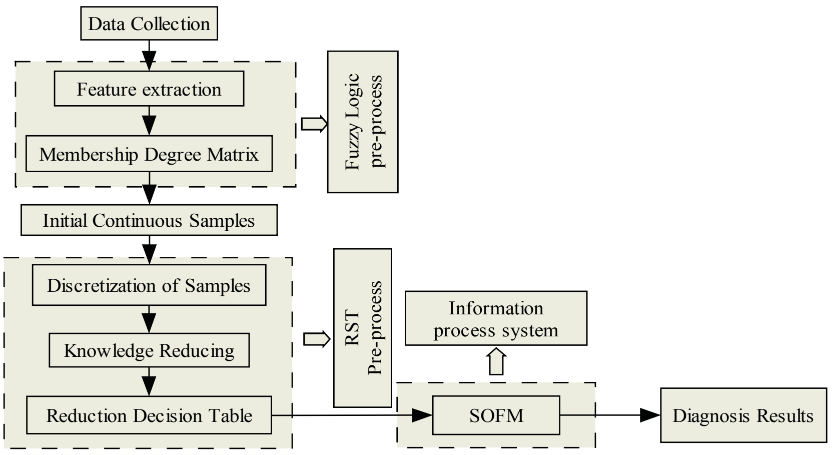


Fig. 2. Fault Diagnostic Process of Fuzzy Logic, Rough Set and SOFM Neural Network

5 Application of Faults Diagnosis on Rotating Machinery

According to field expert's experience and theoretical research outcomes, Table 1 shows faults classification of rotating machinery.

Table 1. Faults classification of rotating machinery

Faults No.	Faults Name	Faults No.	Faults Name
F0	normal	F10	pedestal looseness
F1	imbalance	F11	foundation looseness
F2	components missing	F12	worn coupling
F3	bent shaft	F13	vibration caused by clearance (electricity magnet excited)
F4	shaft-seal rubbing	F14	Sub-harmonic vibration
F5	axial rubbing	F15	oil whirl
F6	axial misalignment	F16	oil whip
F7	eccentricity faults	F17	steam excited vibration
F8	rotor crack	F18	valve vibration
F9	shrunk-on-disc failure	F19	power disturbance

Like any faults diagnosis, feature extraction is an important step in detecting faults of rotating machinery. For a typical vibration analysis of rotating machinery, features can be extracted from spectrum domain. Nowadays Spectrum analysis is well proven as a practical and powerful tool for fault diagnosis of rotating machinery because it results from a great deal of engineering experience. However, it is a difficult work to find the best relationship of faults and spectrum data because state of rotating machinery is complex, and influenced by numerous of process parameters. Table 2 shows the spectrum and process features description.

Now we get twenty samples – one normal sample (F0) and nineteen standard fault samples (F1-F19). In order to minimize the number of the SOFM's input, we process them by Rough Set Theory. Discretize them by Boolen Reasoning algorithm and

Table 2. Spectrum and process description

Spectrum	Description	Process	Description
S1	0.01~0.39X	P1	Amplitude jump during the operation
S2	0.4~0.49X	P2	vibration at varying power load
S3	0.50X	P3	axial vibration
S4	0.51~0.99X	P4	shaft average centerline
S5	1×X	P5	critical speed spectrum
S6	2×X	P6	stable at varying running speed
S7	3~5×X	P7	vibration level increase with running up
S8	odd of X	P8	level jump during running up
S9	High X	P9	3x at 1/3 critical speed
S10	Power line	P10	half-speed whirl

compute the reduction by Johnson's algorithm [8]. The calculation result is RED= {S1, S2, S3, S5, S9, P1, P2, P3, P5}. According to Table 2 and RED, we can obtain the final input, which contains only nine features. The number of output nodes will affect the diagnosis result directly [1]. We select the number of output nodes as 100 (10×10). The training result is shown in Fig. 3. As is shown in Fig.3, the training samples have been fully distinguished (classified).

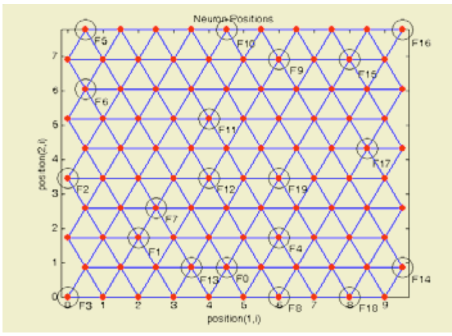


Fig. 3. Training result of standard samples

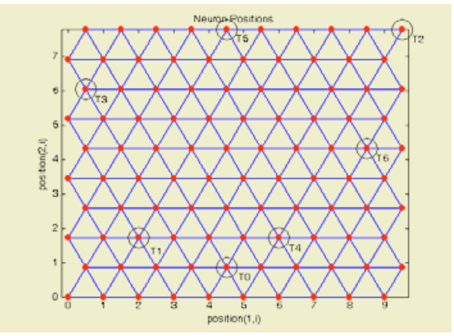


Fig. 4. Diagnosis result

In order to prove the practicability of this method, we provide seven testing samples (simulation fault samples) T0 – T6, which are simulated from F0, F1, F16, F6, F4, F10 and F17. The output input seven simulation samples mapping is shown in Fig. 4.

Comparing Fig.3 and Fig.4, we can gain the following diagnosis results: T0 is diagnosed as normal (F0); T1 – imbalance (F1); T2 – oil whip (F16); T3 – axial misalignment (F6); T4 – shaft-seal rubbing (F4); T5 – pedestal looseness (F10); T6 – steam excited vibration (F17). This accurate classification has verified the practicability of the RST based fuzzy SOFM diagnosis method.

6 Conclusions

In this paper, a new method combined different techniques – Fuzzy Logics, Rough Set Theory and Self-organizing Feature Map Neural Network – has been proposed to diagnose rotating machinery. First we applied fuzzy subsets and a certain membership function to describe several fault conditions. In this way, we got a membership matrix indicates the relationships between fault causations and their symptoms. Second we applied RST to reduce the number of input of our SOFM neural network. As a result we found that more than half attributes (inputs) have been eliminated while the quality is not decreased. After data pre-process, we construct the SOFM neural network to learn for standard fault samples. Finally, we provide several simulation fault samples (testing samples) to perform diagnosing and gain the reasonable conclusions by comparison between the two output mappings. The accurate diagnosis results have fully verified the practicability of the new method for rotating machinery.

References

1. Jiang, D., Wang, F., Zhou, M., Ni, W.: Application of Fuzzy Self-organizing Neural Network to Aeroengine Fault Diagnosis. Journal of Aerospace Power, **16** (2001) 80-82
2. Ling, W., Jia, M., Xu, F., Hu, J., Zhong, B.: Optimizing Strategy on Rough Set Neural Network Fault Diagnosis System. Proceedings of the CSEE, **23** (2003) 98-102
3. Jiang, D.: The Study on Technique and Application of Performance Monitoring and Diagnosis for Large-scale Power Plants. Tsinghua University, Dept. of Thermal Engineering (1996)15-29
4. Benbouzid, M.E.H. and Nejjari, H.: A Simple Fuzzy Logic Approach for Induction Motors Stator Condition Monitoring. Electric Machines and Drives Conference (2001) 634-639
5. Yon, J., Yang, S., Jeon, H.T.: Structure Optimization of Fuzzy-Neural Network Using Rough Set Theory. International Fuzzy Systems Conference Proceedings (2001) 1666-1670
6. Kusiak, A.: Rough Set Theory: A Data Mining Tool for Semiconductor Manufacturing. IEEE Transactions on Electronics Packaging Manufacturing, **24** (2001) 44-50
7. Kohonen, T.: The Self-organizing Map. Proceedings of IEEE, **78** (1990) 1464-1480
8. Aleksander, ▯hrn: ROSETTA Technical Reference Manual. Norwegian University of Science and Technology, Dept. of Computer and Information Science (2001) 16-28

Identification of the Acoustic Fault Sources of Underwater Vehicles Based on Modular Structure Variable RBF Network

Linke Zhang[1], Lin He[1], Kerong Ben[2], Na Wei[2], Yunfu Pang[2], and Shijian Zhu[1]

[1] Institute of Noise and Vibration, Navy University of Engineering, Wuhan, Hubei 430033, China
[2] Department of Computer Science, Navy University of Engineering, Wuhan, Hubei 430033, China

Abstract. In this paper, neural network approaches are for the first time employed to identify acoustic fault sources of underwater vehicles. According to the characteristics of acoustic fault sources, a novel sources identification model based on modular structure variable radial basis function (SVRBF) network is proposed. Unsupervised algorithms for clustering and supervised learning are combined to train the network, especially, the number of hidden and output layer neurons can be modified on-line so that the network has the capability of incremental learning. The results of experiment show that the proposed network has better generalization performance than traditional BP network and RBF network, and is effective in learning new fault patterns.

1 Introduction

For the noise control of underwater vehicles, it is most important to identify the acoustic fault sources in the complex underwater system. A number of different approaches such as time series analysis, correlation analysis, spectrum analysis, wavelet methods, etc. have been applied to identify acoustic fault sources of the underwater vehicles. Unfortunately, excessive noise sources and interaction between them result in complex hull vibration and nonstationary noise signal, which makes the traditional model-based techniques be ineffective and fault sources are often hard to detect [1].

However, the fault sources identification of underwater vehicles can, in nature, be regarded as a pattern recognition problem. As we all known, neural network approaches require no detailed analysis of the different kinds of faults or modeling of the system, and they have been applied to a wide range of pattern recognition problems [2],[3]. Thus, this paper proposes a novel modular structure variable radial basis function (SVRBF) network to identify acoustic fault sources of underwater vehicles.

The paper is organized as follows. Section 2 provides a brief analysis of the characteristics of acoustic fault sources of underwater vehicles. The architecture and algorithm of modular SVRBF network is discussed in detail in Section 3. Section 4 summarizes the steps of source identification approach based on modular SVRBF network. A validate experiment is introduced in Section 5. Finally, a conclusion is given in Section 6.

2 Analysis of Acoustic Fault Sources Characteristics

Analyzing the characteristics of fault sources is of great benefit to design reasonable network architecture and algorithm.

(1) The space distribution of fault sources
There exist large numbers of potential acoustic fault sources in underwater vehicles, and different acoustic fault may exhibit similar feature. So it is not the best choice to apply one large-scale neural network to identify all fault sources. According to the physical and feature space distribution of fault sources, we may divide the whole vehicle into several subsystem, and then design different sub-networks to identify fault sources of different subsystems.

(2) The nonstationary feature of fault sources signals
The hull vibration and underwater radiated noise signals are usually nonstationary. And the fault feature excursion happens with the change of operation condition, velocity of flow, water temperature and so on, which increases the difficulty of identifying fault sources. So a good generalization ability of neural network is required.

(3) The abrupt fault sources
How to collect complete fault sample is the key to fault sources identification of underwater vehicles. Because of the hardship of underwater acoustic experimentation, the appearance of some new abrupt faults is unavoidable during the process of fault sources identification. Thus it is important of the neural network to have the capability of incremental learning new fault patterns.

3 Modular SVRBF Network

From the analysis of acoustic fault sources characteristics, it is required that neural network have good convergence and generalization performance and incremental learning capability to identify fault sources. However, the traditional BP network exhibits serious drawbacks and limitations and can not meet the requirement [4],[5]: (1) BP network is slow to train and may converge to a local minimum; (2) BP network provides high output responses to input data that fall into regions of the measurement space where there are no training samples, and is sensitive to outliers which may affect every free parameter in the network. The RBF network model overcomes some of these problems. Compared with BP network, RBF is faster to train and provides low output responses to inputs that fall into regions of the data space where there are no training samples. Furthermore, the RBF network can learn and memorize new fault pattern if only the number of hidden and output neurons can be adaptively adjusted on-line. Thus, a sources identification model based on modular SVRBF network is proposed.

3.1 Architecture of Modular Network

There are three levels included in the modular network. The first pre-allotting level is used to choose the correct sub-network based on condition monitoring information from other sensors. The second identification level composed of several RBF sub-networks is responsible to identify fault sources. Considering that different sub-networks may be chosen simultaneously, the third level is used to fuse different identification results for a more credible conclusion. The architecture of network is shown in Fig.1.

The modular network exhibits the space distribution characteristics of fault sources, which can solve convergence problem on a certain extent. Otherwise, each sub-network is only responsible for fault identification within one subsystem, especially, the hidden and output neurons of each sub-network is variable on-line for better generalization performance and incremental learning capability.

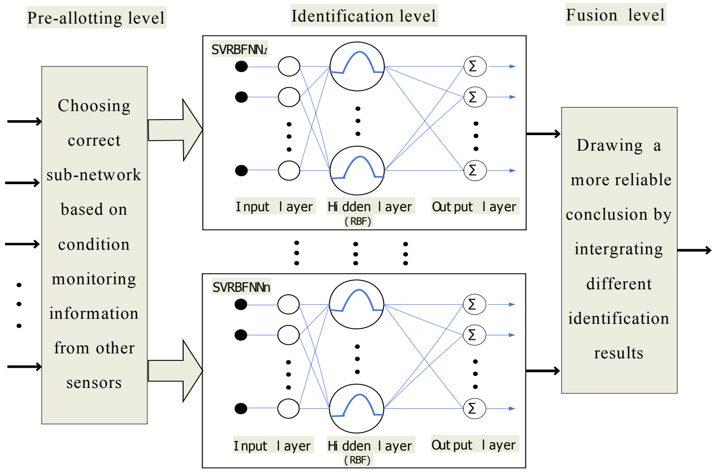

Pre-allotting level
Identification level
Fusion level
Choosing correct sub-network based on condition monitoring information from other sensors
SVRBFNN1
SVRBFNNn
Input layer
Hidden layer (RBF)
Output layer
Drawing a more reliable conclusion by intergrating different identification results

Fig. 1. Architecture of modular SVRBF network

3.2 Algorithm of SVRBF Network

It is clear that to fast and effectively determine the number of hidden neurons, the centers of RBFs and output layer weights are the key tasks in RBF network design [6]. Learning in the proposed RBF network may be done in three stages: 1) determining the parameters of the RBFs, including the centers and the scaling parameters, 2) calculating of the output layer weights, and 3) adjusting of the hidden and output neurons on-line.

(1) Determining the Parameters of RBFs

The number of hidden layer neuron is adaptively adjusted. A new neuron is added to the hidden layer when the fault pattern cannot be identified. The weights of hidden lay-er are initialized with the value of new fault sample. The training is unsupervised.

1) A suitable initial radius of the core function r is determined according to the prior knowledge of feature distribution of fault samples;
2) Start from one hidden neuron and initialize the weights $W_{1,1}$ (the first clustering center) with a random sample X_1. The number of element in the clustering and the number of hidden neuron are respectively: $m_1=1$, $N_1=1$;
3) Input a new sample X_2, and calculate the distance d between the sample X_2 and the clustering center $W_{1,1}$, that is, $d=\|X_2-W_{1,1}\|$. If $d\le r$, the new sample is subject to the first clustering, then increase the number of the element of the clustering by 1:

$m_1= m_1+1$, and then $W_{1,1}=(W_{1,1}+ X_2)/m_1$; if d>r, X_2 is chosen to be a new clustering center, that is, $m_2=1$. And a corresponding new hidden neuron is added, whose initial weight $W_{1,2}$ is set X_2. Now the number of hidden neuron N_1 is 2.

4) Input X_i (i=3,4,…,N) one by one, and calculate the distance $d_{ij}=\|X_i -W_{1,j}\|$, i=3,4,..N, j=1,2,…,N_1. Supposing that the minimum distance d_{min} is defined as: $d_{min}=\min\|X_i- W_{1,k}\|=\|X_d-W_{1,k}\|$. If $d_{min}\le r$, the sample X_d is subject to the kth clustering, and $m_k=m_k+1$, then $W_{1,k}=(W_{1,k}+ X_d)/m_k$. If $d_{min} > r$, X_d is chosen to be a new clustering center, $N_1= N_1+1$, $W_{1,N1}=X_d$;
5) The step 4 continue until all training samples are used;
6) Calcu1ating the error:

$$J=\sum_{j=1}^{N_1}\sum_{x_i\in j}\sum_{n=1}^{N}\left|x_i(n)-W_{1,j}(n)\right|^2 . \tag{1}$$

The training will be terminated when the error is small enough.

7) Calculate the radius of the core function:

$$\sigma^2{}_j=\frac{1}{m_j}\sum_{x_i\in j}\sum_{n=1}^{N}\left|x_i(n)-W_{1,j}(n)\right|^2 , \quad j=1,2,\ldots, N_1. \tag{2}$$

(2) Calculating of the Output Layer Weights

The classical LMS algorithm was used to train the output layer weights [7]. And it is subject to supervised learning.

1) Initialize the connecting weights between hidden and output layer $W_{2,k}(j)$, randomly;
2) Given the train set(X, t), calculate the output of the hidden layer:

$$u(j)=\exp(-\left\|x-W_{1,j}\right\|^2/(2\sigma^2{}_j)) , \quad j=1,2,\ldots, N_1. \tag{3}$$

3) Calculate the output of the output layer:

$$y(k)=\sum_{j=1}^{N_1}W_{2,k}(j)u(j) , \quad k=1,2,\ldots, N_2. \tag{4}$$

4) Calculate the output error:

$$e(k)=t(k)-y(k) , \quad k=1,2,\ldots, N_2. \tag{5}$$

5) Modify the weights

$$W^{s+1}{}_{2,k}(j)=W^{s}{}_{2,k}(j)+\lambda(s)e(k)u(j) , \quad j=1,2,\ldots, N_1,\ k=1,2,\ldots, N_2. \tag{6}$$

where λ(s) is the step size, which is adaptively adjusted with the increase of the iterations.

6) Than go to step 2 until the iteration process is end.

(3) Adjusting the Hidden and Output Neurons On-Line

A RBF network provides low output responses to inputs that fall into regions of the data space where there are no training samples, and in nature, can detect a novel fault.

Furthermore, the network will exhibit incremental learning capability if the output neurons can be adaptively adjusted on-line. The process can be explained as follows: when a RBF network with M output neurons is utilized to identify fault sources, a new output neuron together with a new hidden one will be added to the network on-line if the input sample cannot be identified. Then the new input sample should be used to train the network so that the network can memorize the new fault pattern while not forget old patterns. The algorithm of optimally determining the total number of output and hidden layer neurons is similar to the above one of determining the centers of RBFs, and here is omitted.

4 Source Identification Based on Modular SVRBF Network

The approach can be summarized in the following steps:

Step 1
According to some prior knowledge, the sources identification model based on modular SVRBF network is set up, and the threshold value of each fault pattern is set as:$\lambda = (\lambda_1,\lambda_2,\ldots,\lambda_M)$.

Step 2
Given the input x, select a suitable sub-network based on condition monitoring information from sensors and calculate the output of each pattern y_i (i=1,2,…,M), then compare it with the corresponding threshold λ_i (i=1,2,..,M).

If $y_i \geq \lambda_i$, then $x \in$the i^{th} pattern, the degree of membership is y_i; otherwise, $x \notin$ the i^{th} pattern.

If for all i, we have $y_i < \lambda_i$, hence, x is subject to a new fault pattern. And then go to step 3; otherwise, it must exit one i, we obtain $y_i \geq \lambda_i$,, then go to step 4.

Step 3
Set M=M+1, add a new output neuron and a new hidden neuron. Initialize the weight of the new hidden neuron $W_{1,M}$ with the sample value and the connecting weights $W_{2,M}$ between the hidden layer and output layer with a random value. The LMS algorithm is employed to modify $W_{2,M}$.

Step 4
According to the rule of maximum degree of membership, if $y_j > y_i$, i=1, 2, …, M, $i \neq j$, we have $x \in$the j^{th} fault pattern.

5 Experiment and Result

An experiment is carried out in the model ship towing tank in Navy University of Engineering (see Fig. 2) to demonstrate the effectiveness of the proposed SVRBF network. In order to simulate different acoustic fault of underwater vehicle, a composite cylindrical shell whose reduced scale size is 1:9 to one cabin of some type of underwater vehicle and two actuators with different supply frequency ranged from 20 to 320 Hz are used.

The vibration signals collected from accelerometers mounted on the shell are fed into a charge amplifier. The amplified signal is sampled and inputted into Pimento signal analysis system. The sampling rate is 1kHz, and 1024 samples for each data

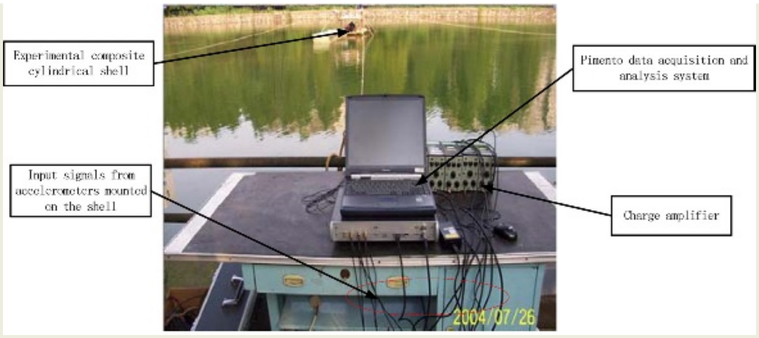

Experimental composite cylindrical shell
Pimento data acquisition and analysis system
Input signals from accelerometers mounted on the shell
Charge amplifier
2004/07/26

Fig. 2. The scene of sources identification experiment

frame, which are transformed into frequency-domain signals and used for feature vectors as the inputs of the neural networks.

Five types of fault patterns, whose feature frequency is respectively 20Hz, 110Hz, 220Hz, 280Hz, and 320Hz, are simulated. The former four fault patterns are utilized to test the generalization performance of the SVRBF network, and the latter to test the incremental learning capability.

In order to assess the effectiveness of proposed network, a traditional BP and RBF network have been developed to solve the same problem for comparison. The results are shown in Table 1.

Table 1. Results of different networks in fault sources identification

Network		The former four fault patterns		The latter new fault pattern	
		Training sample (89)	Testing sample (100)	Output of network	Testing sample(189)
BP (traingdx)		1.00	0.91	(0.00 0.00 0.00 0.00 0.00)	--
RBF		1.00	0.89	(0.00 0.00 0.00 0.00 0.00)	--
SVRBF	Before	1.00	0.95	(0.00 0.00 0.00 0.00 0.00)	--
	After	--	0.94	(0.00 0.00 0.00 0.00 0.00 1.03)	0.87

As can be seen from Table 1, the results show that: 1) the SVRBF network exhibits better generalization performance, and the identification ratio is 95%, improved by 4% and 5% comparing with the BP and RBF network respectively; 2) the traditional BP and RBF network can not identify the new fault, but the SVRBF network can do this with 87% identification ratio, while not forget the old fault pattern with the identification ratio 94% after neurons adjustment approximating to 95% before.

6 Conclusions

This paper presents a novel modular SVRBF network to identify acoustic fault sources of underwater vehicles. Unsupervised algorithms for clustering and supervised learning are combined to train the network whose hidden and output layer neurons can be modified on-line. The steps of source identification approach based on modular SVRBF network are also given. The results of experiment show that the proposed SVRBF network has better generalization performance than traditional BP and RBF network, and is effective in learning the new fault pattern.

References

1. Yang, D.: Noise Sources Identification and Analysis for Underwater Vehicles. PhD thesis, Harbin university of engineering, Harbin (1998)
2. Lampariello, F. Sciandrone, M.: Efficient Training of RBF Neural Networks for Pattern Recognition. IEEE Transactions on Neural Networks, **12** (2000) 1235-1242
3. Mak, M.W., Kung, S.Y.: Estimation of Elliptical Basis Function Parameters by the EM Algorithm with Application to Speaker Verification. IEEE Transactions on Neural Networks, **11** (2000) 961-969
4. Bishop, C.M.: Neural Networks for Pattern Recognition, Clarendon Press: Oxford (1995)
5. Gao, X., Lu, J.: Target Recognition Algorithm for Passive Sonar System with High Generalization Ability. ACTA ACUSTICA, **23** (1998) 213-220
6. Schwenker, F.: Radial-Basis-Function Networks: Learning and Applications. Fourth International Conference on Knowledge-Based Intelligent Engineering Systems & Allied Technologies, Brighton (2000) 33-43
7. Fang, S.: A Hidden Layer-structure-adaptive Neural Network Used in Underwater Acoustic Target Recognition. Journal of Southeast University, **29** (1999) 89-94

A Dynamic Recurrent Neural Network Fault Diagnosis and Isolation Architecture for Satellite's Actuator/Thruster Failures

Li Li, Liying Ma, and Khashayar Khorasani

Department of Electrical and Computer Engineering
Concordia University
Montreal, Quebec, Canada H3G 1M8
kash@ece.concordia.ca

Abstract. In this paper, a fault diagnosis and isolation (FDI) strategy based on a Dynamically Driven Recurrent Neural Network (DDRNN) architecture is proposed for use in situations when there are thruster/actuator failures in the satellite's attitude control system. The proposed architecture is motivated from the following facts: (1) the satellite's attitude dynamics is highly complex and nonlinear, (2) the large volume of data that is generated in the attitude control system has to be monitored in real-time by the ground station operators which is a highly labor-intensive and time-consuming task, and (3) dynamically driven recurrent neural networks (DDRNN) have been shown to have the ability to learn, recognize and generate complex temporal patterns. To improve the FDI performance accuracy, the proposed architecture is designed to consist of two DDRNNs. The first DDRNN determines and diagnoses the presence of a faulty thruster. The second DDRNN then identifies which thruster is faulty, i.e. it isolates the location of the fault. Extensive simulation results are shown that demonstrate and verify that the proposed two DDRNNs scheme is more efficient and robust as compared to a scheme that is based on a single feed-forward back-propagation neural network or a single DDRNN scheme, especially in the presence of external disturbances and noise.

1 Introduction

Monitoring and diagnosis of spacecrafts require higher control autonomy, which would then enable them to achieve performance with higher precision, faster slewing, and larger maneuvers in spite of actuator, sensor or component failures and disturbances, since: (1) For missions in outer space, the capacities of ground control to respond to emergency situations are limited, mainly because of long round trip communication delays, (2) In hostile environments the ground control could be interrupted for a long time, (3) There is an increased need to minimize the cost of ground-based support during the long duration of space missions, and (4) There is an increased requirement to improve reliability and mission lifetime.

To achieve the goals of autonomous operation, the spacecraft should be designed to have self-contained fault diagnosis, isolation and recovery (FDIR) capabilities. This is to be realized by embedding the FDIR function on the spacecraft. An integral component of this requirement is fault diagnosis since a fault can be correctly isolated only when it is detected. During the last two decades, a large number of work have been performed using analytical approaches based on quantitative models. The idea is to generate signals that reflect inconsistencies between nominal and faulty system operation. Such signals, called residuals, are usually generated using analytical approaches, such as observers, parameter estimation or parity equations based on analytical (or functional) redundancy.

The main assumption made when using the above methods is that a precise mathematical model of the system being diagnosed is required. This makes quantitative model-based approaches very difficult to use in real systems, since any unmodeled dynamics can affect the performance of the fault diagnosis and isolation (FDI) scheme. There are also certain FDI techniques developed that are based on "if-then" rules, which are process-history based qualitative methods. These kinds of approaches also require lengthy and costly development and testing stages, and it is generally difficult to obtain precise and formal satisfactory stability and robustness guarantees.

For fault diagnosis of spacecraft attitude control dynamics, it should be noted that the attitude dynamics of a spacecraft is highly nonlinear, complex and time-varying. On the other hand, dynamically driven recurrent neural networks (DDRNN) have the learning ability to recognize and generate temporal patterns. Therefore, dynamically driven recurrent neural networks (DDRNN) are promising for the development of FDI schemes. In this paper, a FDI scheme based on DDRNN is designed and implemented for achieving FDI and system recovery in the presence of thruster fault in the attitude control system of a spacecraft. To improve the performance and accuracy of the proposed scheme, the architecture is designed to consist of two DDRNNs. In the first step, a DDRNN diagnoses whether there is a faulty thruster or not. In the next step, the second DDRNN identifies which thruster is faulty, i.e. it identifies the location of the fault. Extensive numerical simulations are conducted to demonstrate that the two-DDRNNs scheme is more efficient than a scheme that is based on a feed-forward back-propagation neural network, especially in the presence of disturbances and noises. Moreover, our proposed approach is much more efficient than a single-neural network DDRNN scheme with improved stability and robustness properties.

2 Thruster and Actuator Failures

This paper focuses on spacecraft attitude control system thruster and actuator diagnosis and isolation. It is assumed that the spacecraft is a rigid body with thrusters that provide torques about three mutually perpendicular axes, where the equations of motion for the attitude dynamics may be governed by:

$$I\dot{\omega} = -\omega^X I\omega + BT$$

where $\omega \in IR^3$ denotes the inertial angular velocity of the spacecraft with respect to an inertial reference frame, ω^X denotes the skew matrix of ω, and I denotes the second-moment of inertial matrix (a diagonal matrix) which is taken about the mass center of the spacecraft.

Let us assume that there are six thrusters that provide the desired torques in the three mutually perpendicular directions, and every two thrusters are arranged in one pair, where one is the redundant or backup while the other one is the real operational actuator. For instance, the thruster 1 and thruster 2 are arranged in the X-axis direction, thrusters 3 and 4 are arranged in the Y-axis direction, and thrusters 5 and 6 in the Z-axis direction. Accordingly, $T = [t_1, \cdots, t_6]^T$ denotes the torques generated by the corresponding individual thrusters (actuators), $B \in IR^{3\times 6}$ denotes the allocator which assigns and switches on and off the torques from each thruster(actuator). Under the corresponding situation with no thruster(actuator) failure, we have $B = \begin{bmatrix} 1\,0\,0\,0\,0\,0 \\ 0\,0\,1\,0\,0\,0 \\ 0\,0\,0\,0\,1\,0 \end{bmatrix}$.

The actual torque $U_a \in IR^3$ which is applied to the spacecraft is given by $U_a = BT$. Let $U_c = [U_{c1}, U_{c2}, U_{c3}]$ denotes the vector of control torques commanded by the controller to the three pairs of thrusters (each pair has one operating thruster and one redundant one). In the ideal case, with no fault, each thruster generates the commanded torque. This implies that switching on the torques from thrusters 1,3, and 5, and switching off torques from thrusters 2,4, and 6, which are the redundant ones. Therefore, the allocator B is used as a security measure to keep-off the undesired torques due to the delays in response from the faulty thrusters when switching the two thrusters of one thruster pair. This scheme is to ensure that $T = U_c$.

In practice, usually there are two possible types of thruster faults: (1) no torques are generated due to a complete loss of thruster, and (2) torques are not equal to the desired values corresponding to the control torques commanded by the controller. The latter case is possible because of the saturation property of a thruster, i.e. when the control torques commanded by the controller are exceeding the limited torques of a thruster, where the saturated torques will be generated instead of the desired values. The relationship between the reference input U_r (control objective), the commanded torques U_c, the torques generated by the thrusters T, and the actual torques U_a that are applied to the spacecraft is shown in Figure 1. In the presence of a thruster failure when the actual torques are not the desired ones corresponding to the commanded torques, it is clear that the control objectives cannot be achieved.

3 Dynamically Driven Recurrent Networks

It is well-known that recurrent neural networks can respond and represent temporally to an externally applied input signal. These kinds of recurrent networks are known as dynamically driven recurrent neural networks (DDRNN). Due to the application and presence of feedback, a DDRNN can be used to capture a state representations. This makes DDRNNs suitable for diverse applications. Moreover, the use of global feedback has the potential of reducing the memory requirement significantly.

3.1 State-Space Model and Elman's Neural Network

State space representation of a dynamical nonlinear and complex system is one of the properties of DDRNNs. The output of the hidden layer is fed back to the input layer

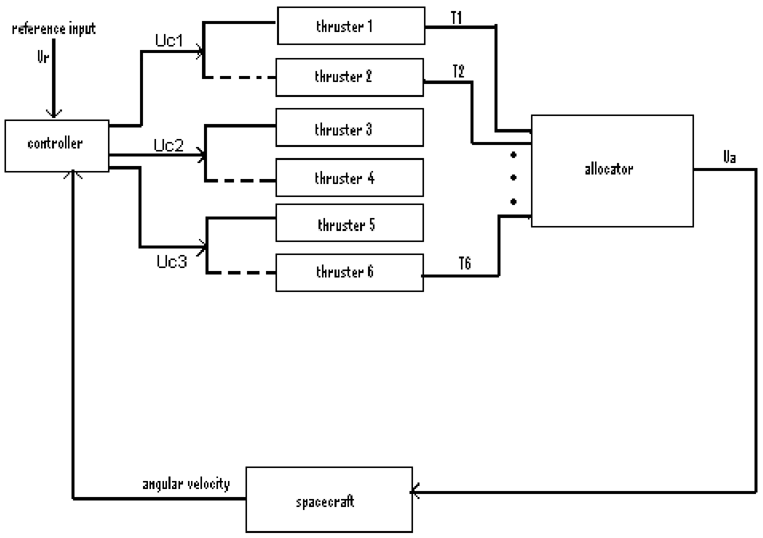


Fig. 1. Relationship of U_r, U_c, and U_a

via a bank of unit delays. The input layer consists of a concatenation of feedback nodes and source nodes. The neural network is connected to the external environment via the source nodes. The number of unit delays used to feed the output of the hidden layer back to the input layer determines the order of the model. Elman's network has a similar architecture to that of ordinary state-space model except for the fact that the output layer could be nonlinear and the bank of unit delays is omitted. Elman's network contains recurrent connections from the hidden neurons to a layer of context units consisting of unit delays. These context units store the outputs of the hidden neurons for one time step, and then feed them back to the input layer. Thus the hidden neurons have some record of their prior activations, which enables the network to perform learning tasks that extend over time. The hidden neurons also feed the output neurons. Due to the nature of the feedback around the hidden neurons, these neurons may continue to recycle information through the network over multiple time steps, and thereby discover abstract representations of dynamical characteristics. Figure 2 shows the structure of the DDRNNs that is utilized in this paper for detecting and isolating the thruster failure in the attitude control system of a satellite.

4 Two DDRNNs Fault Diagnosis and Identification (FDI) and Recovery Scheme

The stages for the development of the proposed FDI scheme are outlined below:

STEP 1: Application of the DDRNN to determine whether there is a fault
The thruster torques commanded by the controller and the satellite outputs (body angular velocities) sampled from the attitude control system are fed to the DDRNN for

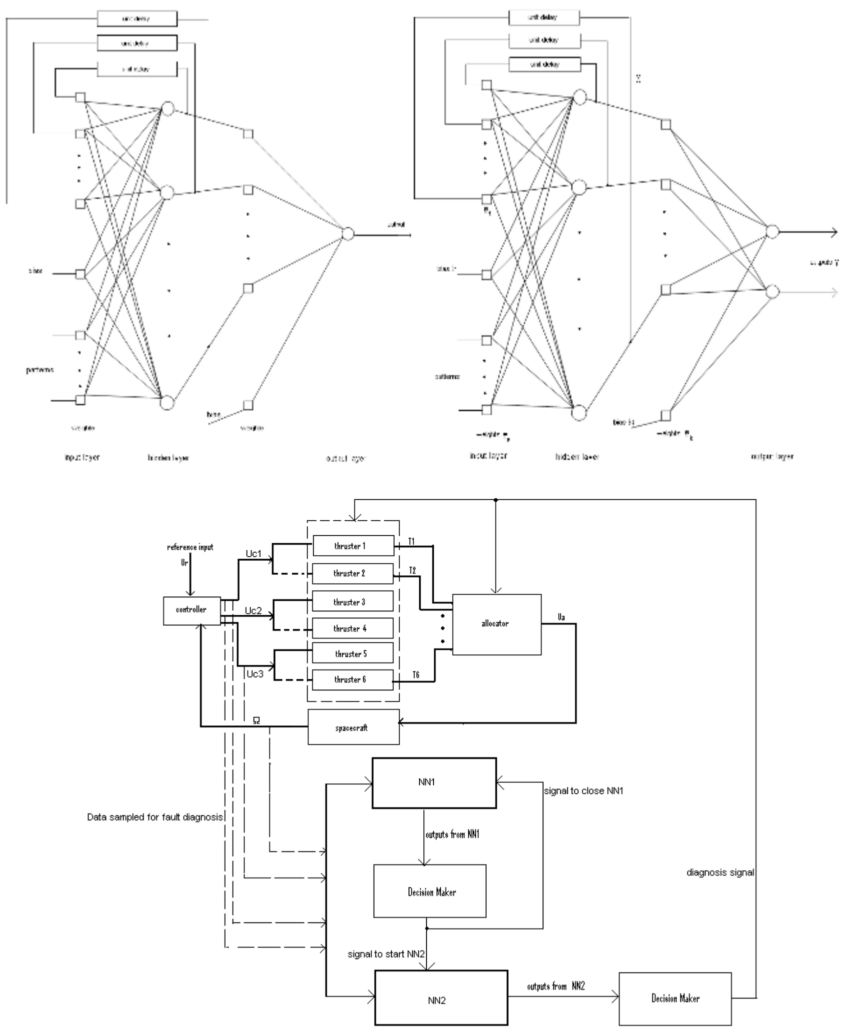


Fig. 2. Structure of the two-DDRNN FDI and system recovery schemes

determining if there is a fault in the thruster. The DDRNN is designed to have one hidden layer and one output neuron. Figure 2 shows the structure of the DDRNN. A tansig transfer function is used for the hidden layer, and a purelin transfer function is selected for the output layer. Specifically, we have: $x_k = \tan sig(w_p p + w_f x_{k-1} + b_1), y_k = purelin(w_h x_k + b_2)$, where $\tan sig(n) = 2/(1 + \exp(-2n)) - 1, purelin(n) = n$. When there is no fault, the target output is chosen as " 0" and when there is a thruster (actuator) fault, the target output is chosen as "1". The output of this neuron is then sent to the Decision Maker (refer to Figure 2). If the output is greater than 0.5, it is decided that a fault exists and the Decision Maker sends a signal to the next DDRNN. Otherwise, it is decided that there is no fault and the Decision Maker does not send a signal.

Once the Decision Maker sends a signal to the next DDRNN, the first DDRNN will become "inactive" for a certain duration of time (in our simulations we have selected 0.8 seconds) until the redundant thruster operates normally. Beyond this time interval the first DDRNN will be activated in order to maintain monitoring the data for the presence of other possible fault(s).

STEP 2: Initiation of the second DDRNN to isolate the faulty thruster
Once the first DDRNN has diagnosed the presence of a fault, a signal will be sent to the second DDRNN. Upon receiving this signal, the second DDRNN using the real-time data from the satellite will attempt to identify the faulty thruster. The second DDRNN is designed to have two output neurons. Figure 2 shows the structure of the second DDRNN. The relationship between each type of thruster fault and the corresponding desired target labels is shown in Table 1.

Table 1.

Fault type	1 (The thruster (actuator) in use from the first pair is faulty)	2 (The thruster (actuator) in use from the second pair is faulty)	3 (The thruster (actuator) in use from the third pair is faulty)
Target output	[0 1]	[1 0]	[1 1]

The two outputs from the second DDRNN are then sent to the Decision Maker. Based on this information the decision maker diagnoses the type of fault and sends a corresponding diagnosis signal to initiate the next process. If the first output is less than 0.5, and the second output is greater than 0.5, the diagnosis made is that the fault is of type 1. On the other hand if the first output is greater than 0.5, and the second output is less than 0.5, the diagnosis made is that the fault is of type 2, and finally if both outputs are greater than 0.5, the diagnosis made is that the fault is of type 3. Upon receiving the diagnosis signal, the following tasks will commence automatically by our proposed strategy: (1) The faulty thruster will be disengaged from the attitude control system. The corresponding element in the allocator matrix will also be set to zero, and (2) Subsequently the corresponding redundant/backup thruster will be engaged, and the corresponding element in the allocator matrix will be set to 1. Therefore the main goal of the above scheme is to complete the FDI process and clear the fault in order to ensure that the satellites attitude control system remains stable and can achieve its desired control objectives.

4.1 Training the First DDRNN for Diagnosing Thruster Failure

As stated above the first DDRNN has one hidden layer and one output layer, and the output layer has one neuron. When training the DDRNN, the following operations take place: (1) The entire input/output data is presented to the network and the network's output is calculated and compared with the actual data for generating the error signals, (2) At each time step, the error is backpropagated to obtain the gradients for each weight and bias parameters. This gradient is an approximation since the contributions of weights and biases via the delayed recurrent connection are ignored, and (3) The gradient is used to update the weights and biases according to $\varDelta X = \alpha \varDelta X(n-1) + \alpha l_r \frac{d_{Perf}}{dX}$, where $\varDelta X$ is the change to the weight or the bias parameters, $\varDelta X(n-1)$ is the change to the weight or bias at the previous iteration, $\frac{d_{Perf}}{dX}$ is the derivative of the performance index with respect to the weight and bias, l_r is the learning rate, and α is the momentum term. Our goal was to get the highest accuracy in the fault diagnosis while using the least number of data from the satellite by utilizing the least number of hidden neurons. The solution was found through numerical simulations. Back-Propagation-Through-Time algorithm was used as the training algorithm. Once the network is trained, it was tested by both clean data as well as data that represented the presence of noise and disturbances in the satellite attitude control system.

4.2 Training the Second DDRNN for Isolating the Faulty Thruster

The second DDRNN has one hidden layer and one output layer, and the output layer has two neurons. The torques commanded by the controller and the outputs of the no-fault model and the three faulty models were sampled and fed into the DDRNN as inputs for training. The desired target outputs are described in the above table. As in the first DDRNN the Back-Propagation-Through-Time algorithm was used as the training algorithm. Once the network is trained, it was test by both clean data as well as data that represented the presence of noise and disturbances in the satellite attitude control system.

5 Simulation Results and Comparison with Other Architectures

Once the two DDRNNs are properly trained they were applied for FDI of thruster failures in the attitude control system of the satellite. The results in Figures 3 to 6 depict the performance of our proposed architecture for the fault type 3 case only due to space limitations. Similar results are also obtained for the other two faulty cases. The fault is simulated to occur at time 2 seconds. In order to illustrate the performance of our proposed strategy with other alternative solutions, the simulation results for the two-DDRNNs, two-FFBPNN (feedforward back-propagation neural network), and one-DDRNN FDI and system recovery scheme are conducted and the results are summarized in Table 2.

The comparison of the data presented in the table below reveals that the fault diagnosis performance accuracy of the two-DDRNNs scheme is higher than that of the two-FFBPNN and one-FFBPNN schemes especially in the presence of disturbances and noise.

6 Conclusion

In this paper, a new fault diagnosis, isolation and recovery method designated as two-DDRNNs FDI and system recovery scheme is presented. The two-DDRNNs FDI and

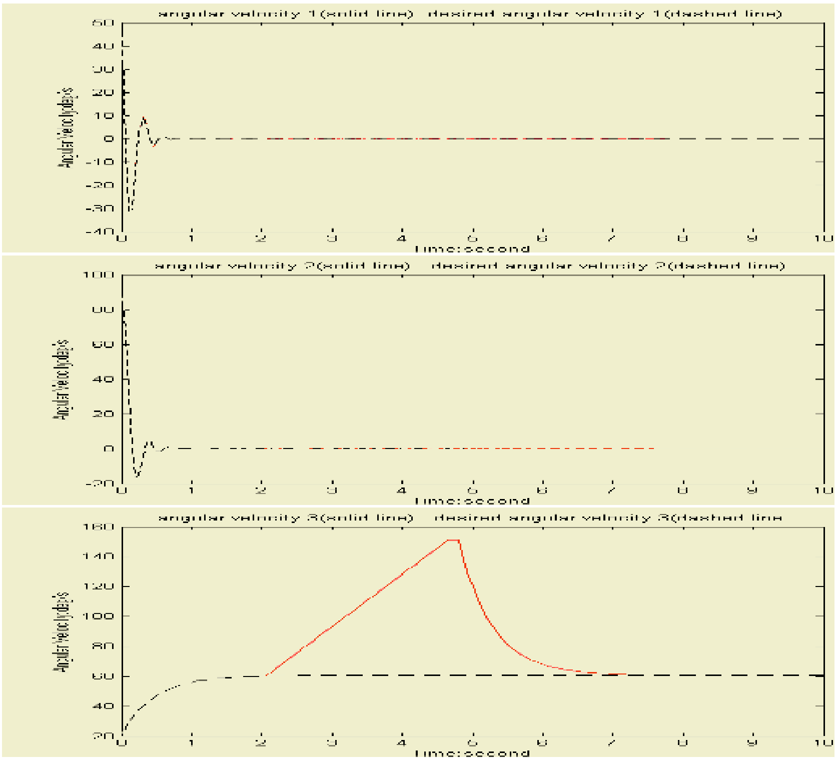


Fig. 3. Response (angular velocity) using the two-DDRNN FDI and system recovery scheme with the third thruster pair being faulty and the desired response (without a fault)

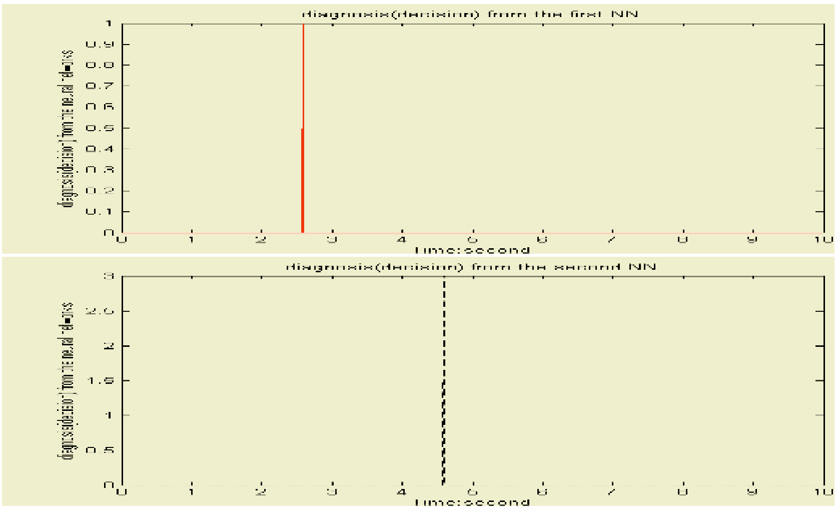


Fig. 4. Diagnosis (decision) from the first and the second DDRNNs FDI and system recovery scheme when the third thruster is faulty

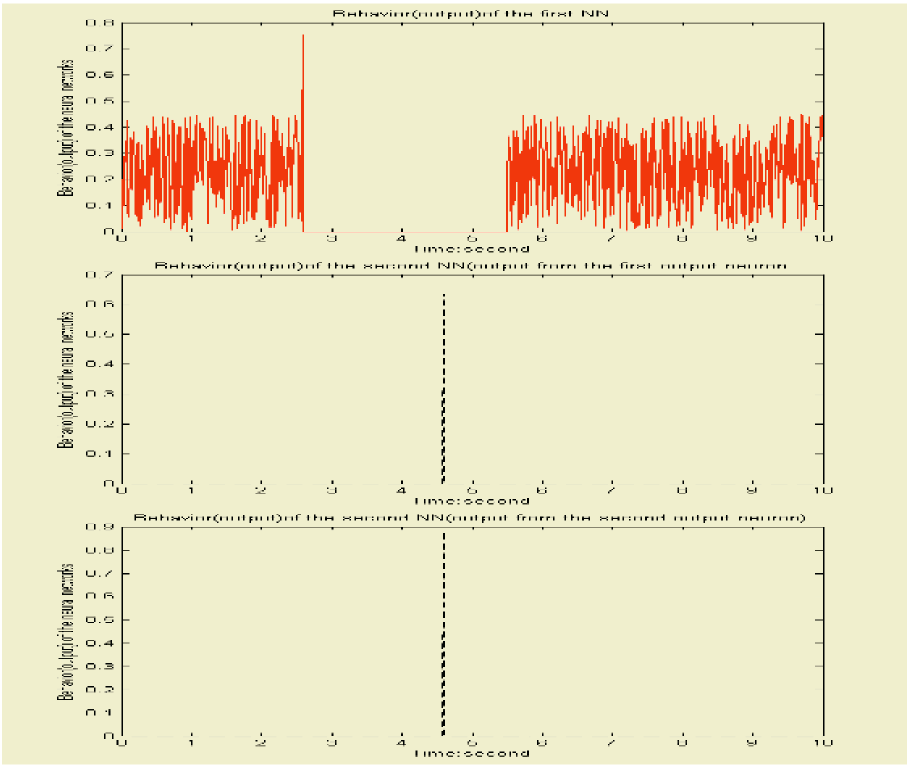


Fig. 5. Behavior (output) of the first and second DDRNNs FDI and system recovery scheme when the third thruster is faulty

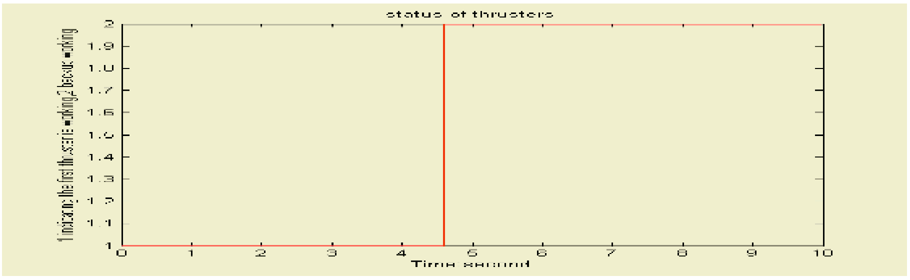


Fig. 6. Status of the thrusters when the third thruster is faulty with the two-DDRNNs FDI and system recovery scheme: "1" indicates the thruster normally in use is operating, "2" indicates the redundant thruster is operating

system recovery scheme makes use of dynamically driven recurrent neural networks. Extensive simulation results demonstrated that the two-DDRNN FDI and system recovery scheme is more efficient than the FDI method based on the common feed forward back propagation neural networks and is also more efficient than a single one-DDRN

Table 2.

Scheme	Structure of Neural Network						Perform. (Clean Data)	Perform. (Noisy Data)
2 DDRNNs	1^{st} DDRNN			2^{nd} DDRNN			97.4%	96.1%
	Input nodes	Hidden nodes	Output nodes	Input nodes	Hidden nodes	Output nodes		
	121	25	1	265	25	2		
2 FFBPNN	Structure of NNs						94.2%	90.3%
	1^{st} DDRNN			2^{nd} DDRNN				
	Input nodes	Hidden nodes	Output nodes	Input nodes	Hidden nodes	Output nodes		
1 DDRNN	121	25	1	265	25	2	95.1%	78.8%
	Input nodes		Hidden nodes		Output nodes			
	121		25		2			

FDI method. It was demonstrated that the two-DDRNNs scheme has high robustness in the presence of external disturbances and noise. The future work would focus on the FDI of simultaneously multiple thruster failures utilizing our proposed neural networks architectures.

References

1. BoŠkovi, J.D., Li, S.M., Mehra, R.K.: Intelligent Control of Spacecraft in the Presence of Actuator Failure. Proceedings of the 38th IEEE Conference on Decision and Control, **5** (1999) 4472-4477
2. Hughes, P. C.: Spacecraft Attitude Dynamics (1986)
3. Haykin, S.: Neural Networks: A Comprehensive Foundation (1999)
4. Brown, G. M., Bernard, D. E., Rasmussen, R. D.: Attitude and Articulation Control for the Cassini Spacecraft: A Fault Tolerance Overview. Digital Avionics Systems Conference, (1995)
5. Elman, J. L., Finding Structure in Time. Cogn. Sci., **14** (1990) 179-211
6. Wilson, E., Rock, S. M.: Reconfigurable Control of a Free-flying Space Robot Using Neural Networks. Proceedings of 1995 American Control Conference, **2** (1995) 1355-1359

Fault Detection in Reaction Wheel of a Satellite Using Observer-Based Dynamic Neural Networks

Zhongqi Li, Liying Ma, and Khashayar Khorasani

Department of Electrical and Computer Engineering
Concordia University, Montreal, Quebec H3G 1M8 Canada

Abstract. This paper presents a methodology for the actuator fault detection in the satellite's attitude control system (ACS) by using a dynamic neural network based observer. In this methodology, a neural network is used to model a nonlinear dynamical system. After training, the neural network, it can give very accurate estimation of the attitude positions of the satellite. The difference between the actual and the estimated outputs is used as a residual error for fault detection. The simulation results show advantages of this method as compared to the method based on a generalized Luenberger linear observer.

1 Introduction

Rigid attitude control of spacecrafts has been widely investigated in the past several years [1-3]. There are a variety of control techniques that can be used in attitude control system (ACS). In this paper, we apply the classical PID controller in the system. Through the PID controller, the satellite will meet the ACS accuracy requirements (in our case, the satellite should be maintained within 0.25^{o} of a Earth-pointing attitude in all the three axes for any attitude set point change in a specified range) in several minutes.

For attitude change of satellite, we use a three-axis stabilized system with on-axis reaction wheels for control actuation. Normally, the actuator subsystem contains 4 reaction wheels (3 active and 1 redundant) and three magnetorquers used for momentum dumping. In this sense, reaction wheel is an essential element of attitude control system. Reaction wheels are momentum exchange devices which provide reaction torque to a spacecraft and store angular momentum. An accurate mathematical model is presented in [4]. Reaction wheels' working condition and performance heavily affect the attitude control of satellite. For better attitude control, any fault in the reaction wheels should be detected as early as possible. Mainly, there are three types of faults that can happen in reaction wheels that are of interest to us. They are bus voltage fault, current loss (power loss) and temperature change fault in the wheel.

2 Analysis

Three separate control loops will be used to command the three reaction wheels control of each satellite axis. The single-axis control block diagram is shown below.

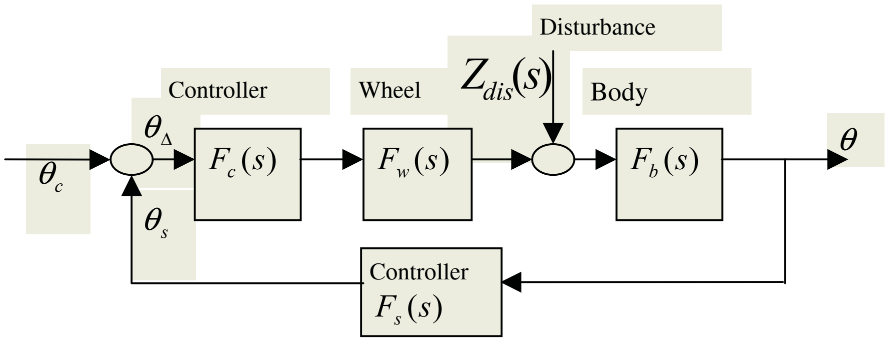


Fig. 1. Single–axis control block diagram.

Sensor
For the transfer function of sensor F_s, a unity feedback is used, that is:

$$F_s = 1 \tag{1}$$

Actuator
For nearly ideal reaction wheel model, which includes consideration of the back-EMF and viscous friction [4], we have

$$F_w = \frac{-G_d K_t Js}{Js + \tau_v + K_e G_d K_t K_f} \tag{2}$$

where G_d is driver gain, K_t is motor torque constant, J is flywheel inertial, K_e is Motor Back-EMF, K_f is voltage feedback gain and τ_v is viscous friction parameter. However, for the purpose of numerical simulations we have conducted a nonlinear model of the components.

Body Dynamics
From Euler's moment equation and considering the satellite body frame aligned with the principal axes, we can get:

$$\tau_x = \dot{\omega}_x I_{xx} + \omega_y \omega_z (I_{zz} - I_{yy}) \tag{3}$$

$$\tau_y = \dot{\omega}_y I_{yy} + \omega_z \omega_x (I_{xx} - I_{zz}) \tag{4}$$

$$\tau_z = \dot{\omega}_z I_{zz} + \omega_x \omega_y (I_{yy} - I_{xx}) \tag{5}$$

where τ represents torque, ω represents angular velocity, I represents inertia and x, y, z represent the principal axes of inertia.

Disturbance
There are certain disturbances that are imposed on the satellite, such as gravity-gradient effects, magnetic-field torques, impingement by solar-radiation and aerodynamic torques [5].

Controller
We choose a PID controller ($F_c = K_p + K_d s + K_i/s$) for this problem, where K_i is selected according to the steady state accuracy and performance requirements. By selecting the desired frequency and phase margin, we can get K_p and K_d through PID controller design methodologies.

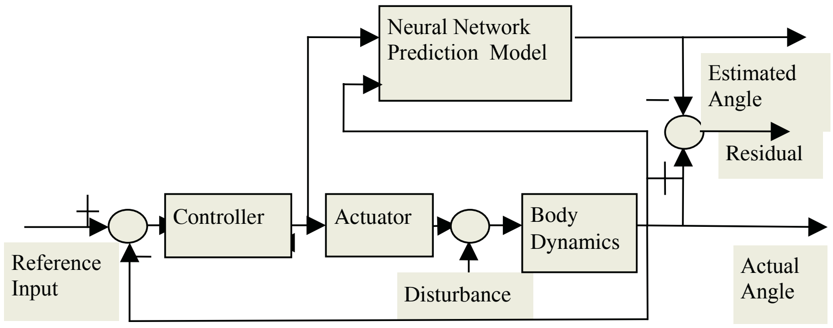


Fig. 2. Fault detection scheme using neural network.

We design three observers for fault detection in three axes separately. The scheme for fault detection in one single axis is shown in figure 2. The residual for fault detection is equal to the difference between the estimated angle and measured angle of the satellite. As we have seen before, to observe the real angle of satellite we need to consider three nonlinear parts: actuator, disturbance and body dynamics. It is well-known that neural networks are capable of forming an arbitrarily close approximation to any continuous nonlinear function, given suitable weighting factors and a network architecture [6]. The simplest approach to model nonlinear dynamical systems is to use a combination of feed-forward network (such as MLP) with tapped delay units. Let us assume that a nonlinear dynamic system is described by:

$$y(k) = F(y(k-1),\ldots,y(k-n),u(k),\ldots,u(k-n)) \tag{6}$$

where $u(k)$ is the input vector and $y(k)$ is the output vector and F represents a general nonlinear function. We can use the one step prediction model shown in figure 3 to model this nonlinear function [7] as represented formally by the following expression:

$$\hat{y}(k) = NN(W, y(k-1),\ldots,y(k-n),u(k),\ldots,u(k-n)) \tag{7}$$

where n is the system order. Consequently, the residual error representing the difference between the actual satellite signals and the neural network estimated may be expressed as $r(k) = y(k) - \hat{y}(k)$. Note that the neural network is trained to model the nonlinear dynamical satellite system (including actuators, disturbances and body

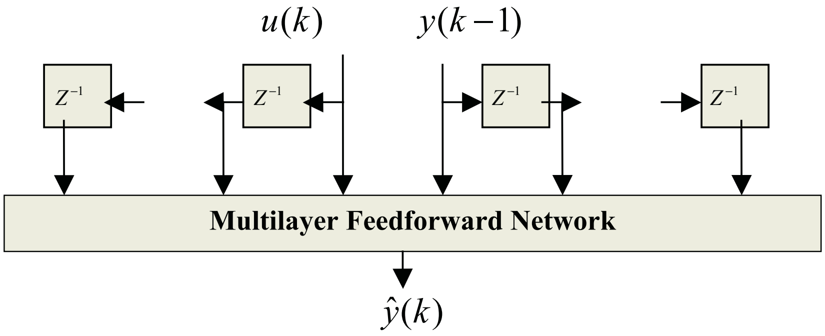


Fig. 3. The neural network model of a nonlinear dynamic system.

dynamics). Therefore the residual thresholds that are determined will correspond to the nominal case (fault free). In the absence of faults, the residual is only due to the unmodeled noise and disturbances. When a fault occurs, the residual deviates from zero (residual errors exceed the threshold).

3 Simulation Results

The initial conditions for the whole attitude control system are given as follows: (a) Initial roll rate: 0 rad/sec; Initial pitch rate: 0 rad/sec; Initial yaw rate: 0 rad/sec; (b) Initial roll angle: 2.3 deg; Initial pitch angle: 3.7 deg; Initial yaw angle range: 4.5 deg; (c) Initial wheel angular speed on X-axis: 0.106 rad/sec; (d) Initial wheel angular speed on Y-axis: 0.108 rad/sec; (e) Initial wheel angular speed on Z-axis: 0.099 rad/sec; and (f) Angle set point change range is: $\Delta\theta = 10 - 15$.

In our simulations, we set the input to the neural network based observer as the output signals of the PID controller and set the output of the observer as the estimated output angles. We introduce 3 time delays in the modeled system, that is

$$\begin{aligned}\hat{y}(k) = NN(W, y(k-1), y(k-2), y(k-3), u(k),\\ u(k-1), u(k-2), u(k-3))\end{aligned} \tag{8}$$

Note that the network architecture used is $Net_{(7\times12\times1)}$ for modeling the satellite system, implying we have 7 neurons in the input layer, 12 neurons in the hidden layer and 1 neuron in the output layer. The following specific information is used in network training (the network training is performed in the system fault free mode and all the training data are normalized to the (-1, 1) range before applying them to the network), namely: the activation function is hyperbolic tangent function; the learning rate is $\eta = 0.0025, \alpha = 0$ and the training performance goal is set to 1e-8. As shown in figure 4, the performance goal has been met after 793 epochs. After the network is trained, we use this trained network to generate the residual signals (the difference between the actual roll angle and the estimated roll angle based on the neural network-based observer) in the fault free cases. As see in figure 5, the esti-

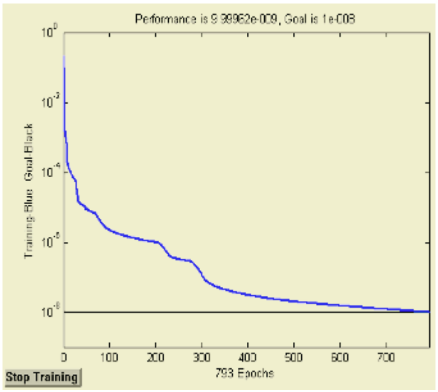


Fig. 4. Network training error.

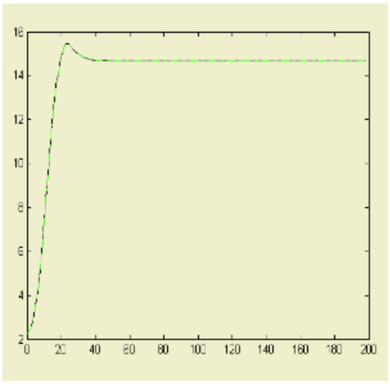


Fig. 5. The actual roll and the estimated roll angle (solid and dashed lines, resp.).

mated roll angle can follow the actual roll angle quite well. By utilizing the residual signals in the experimental fault free cases in the set point change range, the threshold range for the fault detection is selected as [-0.00085 0.0009].

Case I: Bus Voltage Fault Detection

When a bus voltage fault occurs at 1500 sec, which changes from 6.0V to 5.70V, we use the neural network based scheme introduced above as well as the method based on a generalized Luenberger linear observer to detect the fault. As shown in figures 6 to 8, we can detect the fault earlier by using the neural network based scheme, specifically 27 sec as compared to 86 sec after the occurrence of the fault.

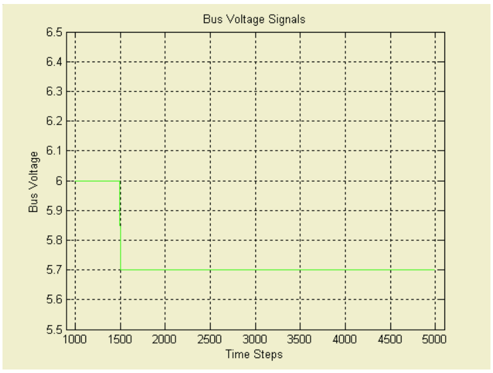
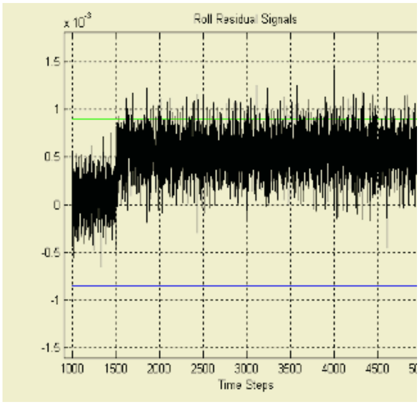


Fig. 6. Bus voltage signals.



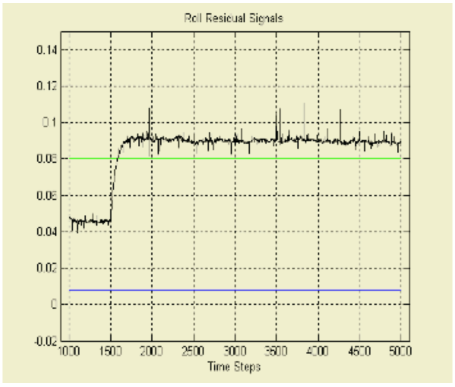


Fig. 7. Roll residual signal based on observer where fault can be detected after 27 seconds.

Fig. 8. Roll residual signal based on NN a generalized Luenberger linear observer where fault can be detected after 86 seconds.

Case II: Current Loss Fault Detection

When a current loss fault occurs at 1500 sec resulting in the roll current limiter changing from 100% to 70%, we use the neural network based scheme introduced in this paper and the method based on a generalized Luenberger linear observer to detect the fault. As shown in figures 9 to 11, we can detect the fault earlier by using the neural network based scheme, that is 127 sec as compared to 238 sec.

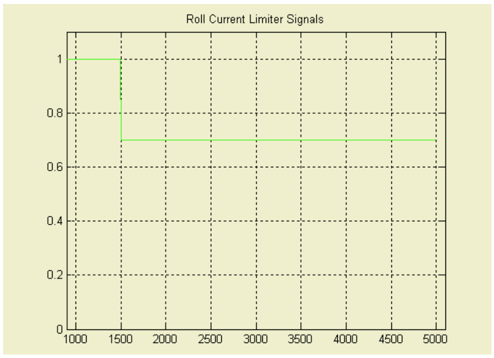
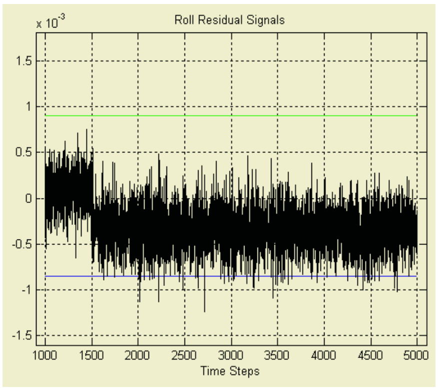


Fig. 9. Current limiter signal.



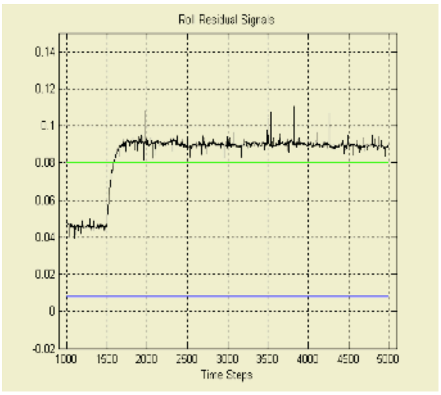


Fig. 10. Roll residual signal based on NN observer and fault can be bserver detected after 127 seconds.

Fig. 11. Roll residual signal based on a generalized Luenberger linear where fault can be detected after 238 seconds.

Case III: Temperature Fault Detection
When a temperature fault occurs at 1500 sec, where the temperature changes from 23 to 60, we use the neural network based scheme introduced above and the method based on a generalized Luenberger linear observer to detect the fault. Unfortunately as shown in figures 12 to 14, we cannot detect the fault by using both schemes.

4 Conclusions

A dynamic neural network based observer scheme is introduced in this paper for fault detection in reaction wheel of the satellites attitude control system. By considering the tapped delay lines for constructing this dynamic feedforward neural network, we can model the nonlinear dynamics of the satellite system quite accurately. Based on this nonlinear observer, we can generate residual signals for fault detection. By comparing the results with those obtained based on a generalized Luenberger linear observer, the neural network based scheme shows a number of advantages in earlier detection of the faults. However, both of these two methods were not able to detect the tempera-

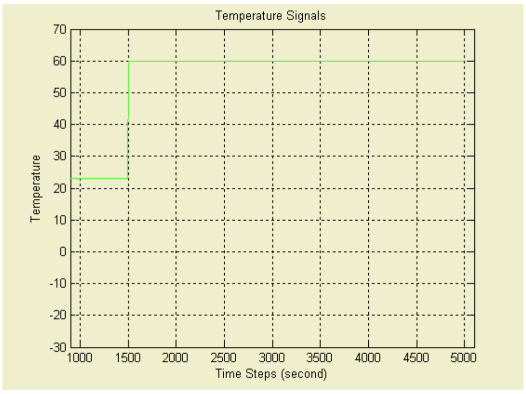
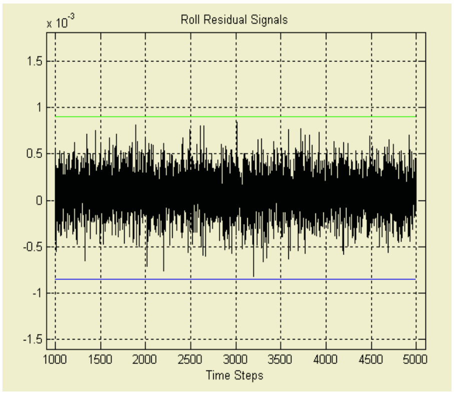


Fig. 12. Temperature signal.



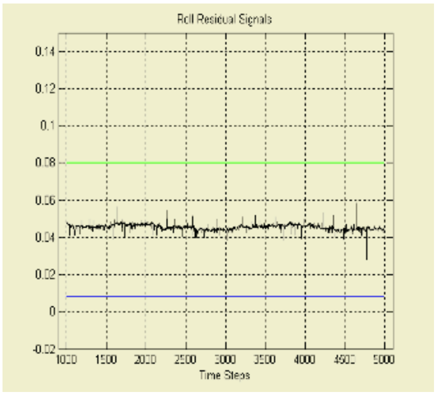


Fig. 13. Roll residual signal based a generalizec Luenberger linear observer cannot be detected.

Fig. 14. Roll residual signal bas-ed on NN- Based Observer and fault observer and faul cannot be detected.

ture fault in the reaction wheel. This is the topic of future research, where the training algorithm and the network structure may have to be optimized for this purpose.

References

1. Hughes, P. C.: Spacecraft Attitude Dynamics. John Wiley & Sons (1986)
2. Kaplan, M. H.: Modern Spacecraft Dynamics and Control. John Wiley and Sons (1976)
3. Kane, T. R., Likins, P. W., Levinson, D. A.: Spacecraft Dynamics. McGraw-Hill (1983)
4. Bialke, B.: High Fidelity Mathematical Modeling of Reaction Wheel Performance. Advances in Astronautical Sciences, **98** (1998) 483-496
5. Singer, S. F.: Torques and Attitude Sensing in Earth Satellites. ed. National Weather Satellite Center, Washington, D.C. (1964)
6. Haykin, S.: Neural Networks. A comprehensive Foundation, Prentice Hall (1999)
7. Chen, J., and Patton, R.J.: Robust Model-based Fault Diagnosis for Dynamic Systems. Kluwer Int. Series on Asian Studies in Computer & Information Science (1999)

Adaptive Wavelet Packet Neural Network Based Fault Diagnosis for Missile's Amplifier

Zhijie Zhou[1,2], Changhua Hu[1], Xiaoxia Han[3], and Guangjun Chen[4]

[1] High-Tech Institute of Xi'an, Xi'an, Shaanxi 710025, China
[2] Department of Automation, Tsinghua University, Beijing 100084, China
zhouzj04@mails.tsinghua.edu.cn, hch6603@263.net
[3] China United NorthWest Engineering Design Institute, Xi'an, Shaanxi 710068, China
[4] Northwestern Polytechnical University, Xi'an, Shaanxi 710072, China

Abstract. Amplifier is a very important device in the missile control system and its working state directly decides the missile's flying stability and hitting precision. In the past, some traditional methods were applied to pick up the features from the signal which was disturbed by noises, but the transient signal can't be identified. In this paper, an adaptive wavelet packet neural network (WPNN) method is presented for diagnosing amplifier fault based on pattern recognition at the first time. This method consists of two stages: firstly, the wavelet packet decomposition is used for feature extraction and secondly, a feed-forward neural network is utilized for pattern classification. Moreover, during the learning phase of WPNN, the wavelet packet entropy and the weights are updated adaptively to minimize the learning error. The experimental results show that the adaptive WPNN method is effective in detecting and diagnosing the amplifier fault of the missile.

1 Introduction

The amplifier is a very important device in the missile control system, which decides the missile's flying stability and hitting precision. Due to the complexity, coupling and specialty of the missile control system, traditional fault diagnosis methods such as estimation techniques [1] are not eligible for the missile fault diagnosis problem anymore. Meanwhile, due to the absence of sufficient and accurate knowledge for missile fault diagnosis, rule-based fault diagnosis expert system [2] is also not an applicable solution. In this paper, an adaptive wavelet packet neural network (WPNN) method is used to detect and diagnose the amplifier fault. This method combines aspects of the wavelet packet transformation for purpose of feature extraction and selection with the characteristic decision capabilities of neural network approaches. Because wavelet has a varying window size and has no requirement for stationarity, this method has more advantages than other methods in dealing with the fault signal of the amplifier which is non-stationary.

The paper is organized as follows. In Section 2, some basic properties of the pattern recognition, the amplifier signal and WPNN are reviewed. An intelligent system is shown in Section 3. This method can extract the fault feature vector and classify the fault patterns of the amplifier. The effectiveness of the proposed method for detecting and diagnosing the amplifier fault is demonstrated in Section 4. Finally, Section 5 presents the conclusion.

2 Preliminaries

2.1 Pattern Recognition

Pattern recognition can be divided into a few stages. In the first stage, the features are extracted from the occurring patterns, i.e. the patterns are converted to the features that are regarded as a condensed representation. Ideally the features contain all the necessary information. In the next stage, the feature selection step, a smaller number of meaningful features that best represent the given pattern without redundancy are identified. Finally, classification is carried out: a specific pattern is assigned to a specific class according to its characteristic features, selected for it [3]. Applying this idea to the diagnostic process of the missile's amplifier, the fault patterns can be identified.

2.2 The Missile Amplifier Signal

The occurring of faults would lead to the change of the transfer function, which results in the change of magnitude and phase characteristic in the different frequency. (a) and (b) of Fig. 1 give the magnitude and the phase curves of the amplifier before and after the fault happens respectively. The figure shows that both the magnitude and the phase characteristics change obviously if the system is fault.

When the input signal stimulates the system, the fault restrains some frequency components of the output, and boosts up the other frequency components at the same time. Here the request for the input signal is that it must have the abundant frequency components, and can cover all the frequency extent. So the pseudo stochastic signal, i.e. m series [4], which is often used in system identification, is selected to be the input.

The output signal of the amplifier is non-stationary and the entropy is treated as an ideal tool for quantifying the ordering of non-stationary signals, so the wavelet packet can be adopted to decompose the whole frequency strip of the amplifier output signal into some small strips, and the entropy of every small strip can be treated as the feature value. Therefore, compared with the normal system, the entropy of every small strip will change when the system is fault. The calculation of entropy will be given in Section 3.2 in detail.

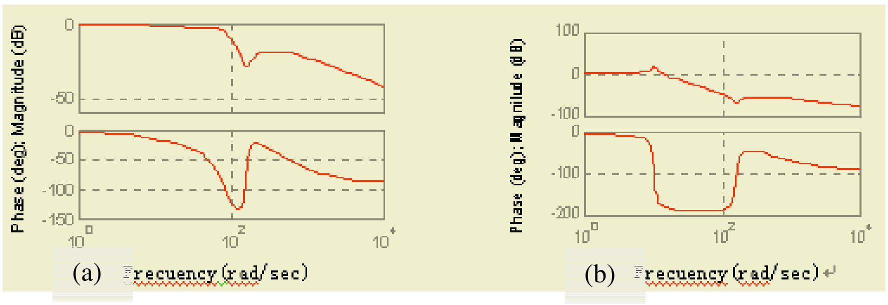


Fig. 1. The normal and fault curves of the magnitude and the phase characteristic: (a) in the normal condition and (b) in the fault condition

2.3 Wavelet Packet Neural Network

The method combining neural network and wavelet is called wavelet neural network. For a system with multi-input and single output $f : R^m \to R^n$, wavelet packet decomposes x using a wavelet packet function $\psi : R^n \to R$. Based on the wavelet packet decomposition, the network function [5] is defined by

$$f(x) = \sum_{i=1}^{N} \omega \psi\left[d_i * \left(x - t_i\right)\right] + \bar{f}\,. \tag{1}$$

where $x = \left(x(1), x(2), \cdots, x(3)\right)$ is the input vector, $D = \left(d_1, d_2, \cdots, d_m\right)$ is module of dilation vector specifying the diagonal dilation matrices D, $\omega(\omega \in R)$ is the weight coefficient, N is the number of wavelet neural cell, $T = \left(t_1, t_2, \cdots, t_m\right)$ are translation vectors, and the additional parameter $\bar{f}$ is to deal with non-zero mean functions on finite domains.

3 The Methodology of Amplifier Fault Diagnosis

3.1 Data Pre-processing

Because of the man-made and external environment factors, the factual testing data of the missile amplifier often contain the exceptional data, which are wrong in nature. So the exceptional data must be eliminated in advance by using the $PauTa$ rule which is given in detail in literature [6].

3.2 The Concrete Adaptive Learning Algorithm for Amplifier Fault Diagnosis

The intelligent system of the adaptive WPNN method for classification of fault waveform patterns from the amplifier testing data is shown in Fig. 2. Feature extraction is crucial in designing the intelligent system based on pattern recognition since even the best classifier will perform poorly if the features are not chosen well. A feature extractor should reduce the pattern vector, i.e., the original waveform, to a lower dimension, which contains most of the useful information from the original vector. In

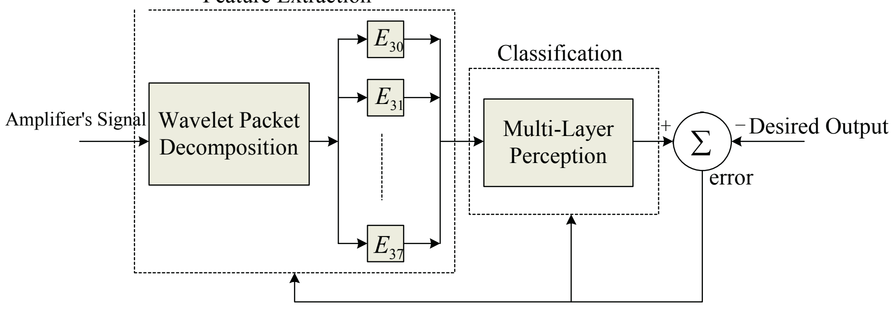


Fig. 2. The pattern classification using wavelet packet neural

sion, which contains most of the useful information from the original vector. In this paper, 1000 testing data of every group are decomposed into 8 feature vectors by wavelet packet.

The waveform patterns of the amplifier fault from the testing data are rich in detail and highly non-stationary. After the data pre-processing, two steps are used to define the kind of these signals using MATLAB6.x with the wavelet toolbox and the neural network toolbox:

Step 1. Wavelet layer: This layer is responsible for feature extraction from the amplifier testing data. The feature extraction includes two stages:

In the first stage, wavelet packet decomposition is completed. For wavelet packet decomposition [7] of the amplifier output signal, the tree structure is used as a binary tree at depth $m = 3$. Wavelet packet decomposition is applied to the fault signal using the Daubechies-1 wavelet packet filters ψ with the Shannon entropy as defined in (2). The detailed processes are given in [8]. Thus $2^3 = 8$ terminal node signals are obtained.

$$E(o) = -\sum_i o_i^2 \log(o_i^2). \tag{2}$$

where o is the output signal and o_i is the i-th coefficients of wavelet packet decomposition of o.

In the second stage, the norm entropy is calculated as defined in (3) of the waveforms at the terminal node signals obtained from wavelet packet decomposition.

$$E(s) = \frac{\sum_i |s_i|^P}{N}. \tag{3}$$

where the wavelet packet entropy E is a real number, s is the terminal node signal and s_i is the waveform of terminal node signals. According to the MATLAB user's guide, P is the power and there must be $1 \le P < 2$ in norm entropy. During the WPNN learning process, the P parameter is updated together with weights to minimize the sum-squared error. The resultant entropy data are normalized with $N = 8$. Thus, the feature vector is extracted by computing the 8-wavelet packet entropy values of every group of amplifier output signal.

Step 2. Multi-layer perception (MLP) layer: This layer [9] completes the intelligent classification using features from wavelet layer. Similarly, the training procedure is terminated when the sum-squared error gets to the minimum. The training parameters and the structure of MLP used in this paper are shown in Table 1. They are selected for the best performance after several different experiments.

4 Experimental Results

The amplifier has seven working states which include one normal state and six fault states. In every state, 1000 output testing data are gained when the input signal is m series. A fault code is given to every group of testing data, i.e., the first group which is normal is corresponding to 00000000, the second one which is fault is to 00000001

Table 1. MLP architecture and training parameters

Network architecture	Training parameters
The number of layers	2
The number of neuron on the layers	Input:8; Hidden:5; Output:8
Initial weights and biases	The Nguyen-Widrow method
Activation functions	Log-sigmoid
Learning rule	Back-propagation
Momentum constant	0.98
Adaptive learning rate	Initial:0.0001 Increase:1.05 Decrease:0.7
Sum-squared error	0.0001

Table 2. The fault code of amplifier

Feature vector								Fault code
0.7445	0.4658	0.2403	0.3113	0.0949	0.1390	0.1523	0.1504	00000000
0.7385	0.5002	0.1965	0.2605	0.0933	0.1197	0.0514	0.0791	00000001
0.6326	0.5292	0.2237	0.2747	0.1316	0.0959	0.1476	0.2070	00000010
0.5638	0.4529	0.3133	0.2740	0.1354	0.2124	0.0489	0.0950	00000011
0.5827	0.4102	0.3865	0.4178	0.2220	0.2708	0.0983	0.1900	00000100
0.6430	0.5467	0.2317	0.2918	0.2183	0.2688	0.0646	0.1573	00000101
0.7707	0.4675	0.2807	0.1768	0.1458	0.1699	0.0832	0.1429	00000110

Table 3. The recognition results of the new fault testing data

Feature vector								Fault code
0.0009	0.0002	0.0009	0.0004	0.0005	0.0410	0.0336	0.9599	00000001
0.0029	0.0006	0.0023	0.0017	0.0013	0.0505	1.0499	1.0649	00000011
0.0104	0.0022	0.0088	0.0055	0.0048	1.0216	0.0291	0.0544	00000100

and the others which are fault may be deduced similarly. Then WPNN can be trained with fault testing data as the input and the fault codes as the output. Table 2 gives the seven groups of fault feature vectors, which are obtained using the adaptive wavelet packet decomposition, and every group includes 8 feature vectors which are the wavelet packet entropies. When the sum-squared error of WPNN satisfies the demand, the weight values are kept. Fig. 3 shows WPNN training performance. If a new group of amplifier fault data is given, it can be recognized. Table 3 gives three groups of new testing data. The fault codes are gained with the adaptive WPNN which has been trained in advance. The faults which these codes represent are consistent with the actual faults of the amplifier. The statistic analysis of many experimental results shows that this adaptive WPNN method can recognize all kinds of fault patterns and the accurate rate of recognition is more than 90%.

5 Conclusions

To our knowledge, this is the first report of utilizing the adaptive WPNN method for diagnosing amplifier fault of missile. The proposed method can efficiently extract the features from pre-processed signal for the purpose of detecting and diagnosing the

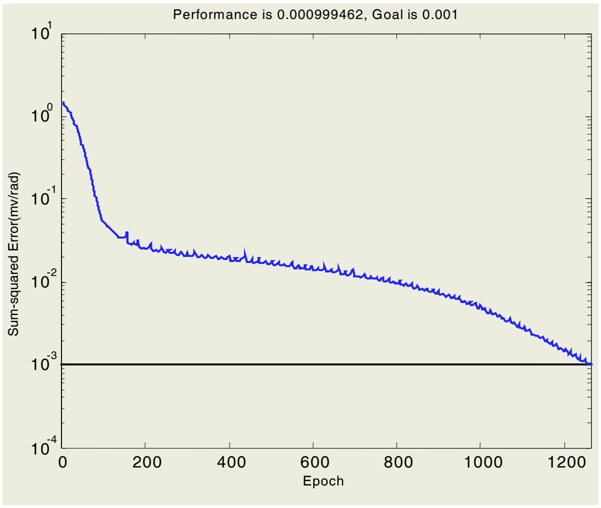


Fig. 3. The WPNN training performance

amplifier fault. The experimental and statistical results show that this method is valid in classification of the fault and the correct rate of classification is very high. The future work is to analyze the change trend of the fault feature and predict the amplifier fault of missile.

Acknowledgements

This work was supported by the NSFC (Grant No. 69931040).

References

1. Wen, X., Zhang, H.Y., Zhang, L.: Fault Diagnosis of Control System and Fault Tolerant Control. Mechanics Industry Press, Beijing (1998)
2. Wu, J.P., Xiao, H.J.: Intelligent Fault Diagnosis and Expert System. Science Press, Beijing (1997)
3. Philpot, E.F., Yogangathan, A.P., Nanda, N.C.: Future Directions in Doppler Echocardiography. Doppler Echocardiography, Lea & Febeiger, Philadelphia London (1993)
4. Wang, Zh. X.: Optimization Estimation and System Identification. Xi'an, The Second Aritillery Engineering Institute, (2000)
5. Cheng, H.T., Huang, W.H.: Wavelet Network Based Time Series Prediction and its Application. Journal of mechanical strength, **1** (1999) 1-3
6. Zhang, M., Yuan, H.: The PauTa Criterion and Rejecting the Abnormal Value. Journal of Zhengzhou University of Technology, **1** (1997) 84-88
7. Zhou, Zh.J., Hu, Ch.H., Wang, Zh.X.: Applying Research of the Wavelet Packet Analysis Based on the Identification of the System in the Fault Diagnosis. Missiles and Space Vehicles, **1** (2003) 45-49
8. Hu, Ch.H.: System Analysis and Design Based on MALAB. Xi'an: The Xi'an Electronic Science and Technology University, (1999)
9. Hagan, M.T., Demuth, H.B., Beale, M.H.: Neural Network Design. China Machine Press, Beijing (2002)

Crack Detection in Supported Beams Based on Neural Network and Support Vector Machine

Long Liu and Guang Meng

State Key Laboratory of Vibration, Shock and Noise, Shanghai Jiaotong University, Shanghai 200240, China
Liu_long@sjtu.edu.cn

Abstract. A study is presented to compare the performance of crack detection using neural network(NN) and support vector machine (SVM) based on natural frequencies. The SVM is a machine learning algorithm based on statistical learning theory, and it is also a class of regression method with the good generalization ability. Firstly, the basic theory of the back-propagation neural network and support vector regression is briefly reviewed. Then the feasibility of the crack detection using these methods are investigated by locating and sizing cracks in supported beams for which a few natural frequencies are available. It is observed that crack's location and depth can be estimated with a relatively small size error. The results show that the SVM is a powerful and effective method for crack identification.

1 Introduction

Crack is a damage that often occurs in structures and may lead to catastrophic premature failure. In the last two decades, a lot of research efforts have been devoted to develop an effective approach for detecting crack in structures.

Artificial neural network(ANN) [1],[2], as a powerful tool, has been applied in damage diagnosis. The relationship between damage feature and physical parameters of structural is established by ANN. However, the traditional neural network approaches have limitations on generalization giving rise to models that can overfit to the training data.

Support vector machine (SVM) [3],[4] a novel machine learning method based on statistical learning theory (SLT), is a small-sample statistical theory introduced by Vapnik. The SVM training process always seeks a global optimized solution and avoids over-fitting, so it is powerful for the problems characterized by small samples, non-linearity and high dimension. Now SVM is an active field in artificial intelligent technology, and has been applied to pattern recognition, computer vision, function estimation and signal processing, etc.

In this study, back-propagation neural network(BPNN) and support vector regression(SVR) are used to analyze crack identification. Some brief descriptions of these two methods are showed in the first section. Then a approach based on modal frequencies is presented to quantify the extent and location of the crack. A simple supported beam with different crack scenarios is used to compare the performance of the BPNN and SVR.

2 The Basic Theory of BPNN and SVR

2.1 Back-Propagation Neural Network(BPNN)

Although researchers have tried different neural network architecture selection methods to build the optimal neural network model, a standard three-layer fully connected BPNN is used in this research because it provides comparable results and works well as a benchmark for comparison.

A typical BPNN consists of a three layer structure: input layer, output layer and hidden layer. Each layer is composed of variable nodes. The number of nodes in the hidden layers is selected to make the network more efficient and to interpret the data more accurately. The relationship between the input and output can be non-linear or linear, and its characteristics are determined by the weights assigned to the connections between the nodes in the two adjacent layers. Changing the weight will change the input–to-output behavior of the network.

A BPNN analysis consists of two stages, namely training and testing. During the training stage, an input-to-output mapping is determined iteratively using the available training data. The actual output error, propagated from the current input set, is compared with the target output and the required compensation is transmitted backwards to adjust the node weights so that the error can be reduced at the next iteration. The training stage is stopped once a pre-set error threshold is reached and the node weights are frozen at this point. During the testing stage, data with unknown properties are provided as input and the corresponding output is calculated using the fixed node weights.

The BPNN has been shown to perform well in many areas in previous research. For structural crack detection, the BPNN is applied to establish the mapping relationship between structural crack feature and crack status(location and severity).

2.2 The Support Vector Regression [5],[6]

In simple terms, the SVM can be thought of as creating a line, or hyper-plane between two sets of data. Imagining in a two-dimensional case, a series of data points for two different classes of data, class A and class B. The SVM attempts to place a linear boundary between the two different classes, and orientate it in such a way that the margin is maximized. In other words, the SVM tries to orientate the boundary in such a way as to ensure that the distance between the boundary and the nearest data point in each class is maximal. The boundary is then placed in the middle of this margin between the two points. The nearest data points are used to define the margin, and are known as support vectors.

SVM can be used for multi-class classifications and regression problems as well. The basic idea of the support vector regression is to map the input data into a feature space via a nonlinear map. In the feature space, a optimum linear decision function is constructed based on the structural risk minimization(SRM) principle; then SVM nonlinearly maps the inner product of the feature space to the original space via kernels.

It has been shown that SVR has several theoretical and practical advantages. One is that a global solution can be obtained which is often unique (this follows from the specific optimization problem). In addition, due to the use of a constrained optimization problem, the data enter the model in an inner product—which means that, numerically, the dimension of the data is irrelevant. Furthermore, due to this inner product, it is possible to use a kernel that enables nonlinear regression in an efficient way.

The kernel can be any symmetric function satisfying the Mercer's condition. Different kernels can be picked up to form different types of the SVM. Typical kernels include: Polynomial kernel, Radial basic function(RBF) kernel, Sigmoid kernel.

3 Crack Detection Based on Modal Frequencies

In this section, the method of crack identification based on modal frequencies is discussed.

3.1 Modal Frequencies as the Characteristics [7]

Many vibration-based methods have been developed in order to detect the crack, which rely on the fact that occurrence of crack leads to changes in the dynamic properties of the structure.

Numerous studies have indicated that an occurrence of the crack reflects a decrease in natural frequencies of the structure. Modal frequencies are used as characteristics to predict the information of the crack. The choice has been made based on the following advantages: (1) the length of the input is limited to several times of the number of the measured DOFs; (2) the natural frequencies represent global behaviors of the structure; and (3) they are cheap to obtain and easy to extract.

The strategy of the crack identification is to train the BPNN and SVR to recognize different cracks from the changes in the frequencies.

3.2 Two-Stage Identification

For the identification of a cracked structure with many unknowns, it is not practical to identify all the parameters at the same time, because this requires complex computation. The two-stage identification is an idea to reduce the number of variables. In this paper, the location of the crack is predicted in the first stage, then the second network uses the crack location information estimated in the first network as part of the input parameters to determine its severity.

3.3 Pre-processing Data

To better and smoother network generalization, a certain amount of data processing is required before presenting the data to the network. Data scaling is a essential step for network training. Prior to training the network, all the data should be linearly normalized to [-1,1].

4 Experiments

The supported beam with different types of crack is used as the example to compare the performance of the BPNN and SVR.

4.1 Description of the Supported Beam [8]

The supported beam has length $L = 450$ mm, width $W = 3$mm, and height $H = 40$mm, made of steel and it has the following properties: modulus of elasticity E =21GPa, the density ρ =7.8×10^{3}kg/m^3, Poisson's ratio $\mu = 0.28$.

The beam is subject to different crack scenarios, i.e., various locations and depth. The cracks are realized by cuts made using a very thin cutting tool. The breadth of the crack is 0.14mm.

The training set includes twenty-four data. Cracks are initiated at four different locations 100mm, 200mm, 300mm and 400mm. Crack depth ratios range from 5mm to 30mm in steps of 5mm, at each crack location. Moreover, five different damage scenarios are used as the test data to check the precision.

By experiment, the first five modal frequencies are obtained. The location and depth of the cracks are presented in terms of the following non-dimensional parameters.

4.2 The BPNN Training [9]

The BPNN is trained using the back-propagation algorithm with adaptive learning and momentum. The BPNN is trained for various different sizes of hidden layer, from 2 to 15 neurons, and the best one is selected. The Neural Network Toolbox in MATLAB [10] is used for the BPNN analysis presented here.

4.3 SVR Training [11]

SVR is trained using the same input features as the BPNN. The SVR function is constructed using the package LIBSVM [12]. An ε-based support vector regression is used with a RBF kernel type. The RBF kernel is chosen because RBF kernel offer the better performance of a number of different kernels on this type of data.

The tolerance of termination criterion ε is set to 0.001, and the cost function C is to 500. The performance did not strongly depend on small variations in these parameters.

4.4 Results

The results of predicting crack's location and depth using BPNN and SVR are showed in Table 1 and Table 2 , respectively.

Table 1. Crack prediction results using BPNN

No.	Actual Location	Predicted Location (x/L)	% Error	Actual Depth	Predicted Depth (a/H)	% Error
1	0.222	0.242	9.01	0.438	0.406	7.31
2	0.444	0.434	2.25	0.188	0.176	6.38
3	0.444	0.485	9.23	0.313	0.213	31.95
4	0.667	0.619	7.20	0.563	0.508	9.77
5	0.889	0.930	4.61	0.688	0.747	8.58

Table 2. Crack prediction results using SVR

No.	Actual Location	Predicted Location (x/L)	% Error	Actual Depth	Predicted Depth (a/H)	% Error
1	0.222	0.238	7.21	0.438	0.447	2.05
2	0.444	0.429	3.38	0.188	0.186	1.06
3	0.444	0.456	2.70	0.313	0.303	3.19
4	0.667	0.652	2.25	0.563	0.572	1.60
5	0.889	0.922	3.71	0.688	0.701	1.89

It is observed that the localization error using BPNN is from 2.25% to 9.01%, and it range from 2.25% to 7.21% using SVR. The absolute error by the SVM is controlled in the range of 15 mm and by BPNN, it is in the range of 20 mm.

The accuracy of the crack sizing is evaluated by measuring the size error that is the difference between actual and predicted depth of crack. It is observed that for No. 1-5, the depth errors range from 7.31% to 31.95% using BPNN and from 1.06% to 3.19% using SVR.

5 Conclusions

As demonstrated in the experiment, SVR generally performs better than the BPNN in the location and severity identification of cracks, especially in predicting the severity. The reason lies mainly on that the SVM implements the SRM principle, which minimizes an upper bound of the generalization error rather than minimizes the training error.

SVM is very promising to tackle complicated problems in damage identification and it could be enlarged to more complex structures in future research. As a reliable technique to identify the structural damage in practice, this method should be tested in more real measurement cases since noise or measurement errors are always present in practice and huge data are involved.

Acknowledgements

This work is supported by Space Support Technology Foundation of China.

References

1. Yun, C.B., Bahng, EY: Neural Network Approach to Damage Assessment of Civil Structure. Proceedings of the Structural Engineers World Congress (SEWC '98), San Francisco, California, USA (1998) 19-23
2. Zapico, J.L., Gonzalez, M.P.: Damage Assessment Using Neural Networks. Mechanical Systems and Signal Processing, **17** (2003) 119-125
3. Vapnik V. N.: Estimation of Dependences Based on Empirical Data. Springer-Verlag, Berlin (1982)
4. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning. Springer-verlag, Berlin (2001)
5. Vapnik, V., Golowich, S.E., Smola, A.: Support Vector Method for Function Approximation, Regression Estimation, and Signal Processing. Advances in Neural Information Processing Systems, San Mateo, CA (1996)

6. Smola, A.J. and Schflkopf, B.: A Tutorial on Support Vector Regression. NeuroCOLT Technical Report NC-TR-98-030, Royal Holloway College, University of London, UK, (1998)
7. Salawu, O.S.: Detection of Structure Damage Through Changes in Frequency: a Review. Engineering structures, **19** (1997) 718-723
8. Cheng, C. (ed): Structure Damage Monitoring and Intelligent Diagnosis. Science Press, Beijing (2001)
9. Samanta, K.R., Al-Balushi, S.A., Al-Araimi: Artificial Neural Networks and Support Vector Machines with Genetic Algorithm for Bearing Fault Detection. Engineering Applications of Artificial Intelligence, **16**. (2003) 657-665
10. Mathworks: Neural Network Toolbox User's Guide. Mathworks, Inc, Natick, MA (1998)
11. Ge, M., Du, R., Zhang, G., Xu, Y.: Fault Diagnosis Using Support Vector Machine with an Application in Sheet Metal Stamping Operations. Mechanical Systems and Signal Processing, **18** (2004) 143-159
12. 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

Early Loosening Fault Diagnosis of Clamping Support Based on Information Fusion*

Weixiang Sun[1], Jin Chen[1], Xing Wu[1,2], Fucai Li[1], Guicai Zhang[1], and GM Dong[1]

[1] The State Key Laboratory of Vibration, Shock & Noise,
Shanghai Jiaotong University, Shanghai 200030, China
{wxsun,jinchen,km_wx,fcli,gczhang,gmdong}@sjtu.edu.cn
[2] School of Mechanical and Electrical Engineering,
Kunming University of Science and Technology, Kunming, Yunnan 650093, China

Abstract. In this paper, a novel global non-destructive evaluation (NDE) technique based on information fusion is proposed to diagnose loosening fault of clamping support. Two feature extraction methods are used to extract feature, which are wavelet package transform and power spectrum density analysis. During diagnosing loosening fault, two local decisions are made by using WP feature and PSD feature respectively. Then the two features are fused to make another local decision. Lastly, the three local decisions are fused to make global decision. The information fusion result have high correct diagnosis ratio and good antinoise performance. The correct diagnosis ratios with no noise and random noise reach 94.3% and 88.6% respectively.

1 Introduction

Early loosening fault diagnosis of clamping support is a kind of structure damage detection. The local Non-destructive evaluation (NDE) technique has been successfully applied in detecting structural damage such as crack [1]. But it still has some disadvantages in real-time damage detecting. So it is necessary to develop a global detecting technique which comprise two methods[2], one is based on dynamics model, the other is based on signal analysis without model. Currently, it is very hard to use model-based method for applications. Therefore, it is important to use the non-modeled global NDE method. In recent years, many experts made researches on global NDE method by using modern signal processing techniques, such as time serial analysis[3], wavelet transform[4] etc.. In this paper, a novel global detecting method based on information fusion (IF) technique is proposed to diagnose loosening fault.

The rest of the paper is organized as follows. In section 2, The construction of feature vectors, including wavelet package feature and power spectrum density feature, is described. The feature fusion and decision fusion framework based on neural networks is discussed in section 3. The loosening experiment of clamping support and fault diagnosis based on information fusion are given in section 4, followed by conclusion in section 5.

* This work is supported by Natural Science Foundation of China (No. 50335030 and No. 10176014)

2 Feature Extraction and Feature Reduction

2.1 Feature Vector Construction Based on Wavelet Package (WP) Analysis

The construction of feature vector based on wavelet package transform, wavelet package (WP) feature vector for short, is described as follow.
Firstly, decompose raw time serial $\{x_i\}(i=1,2,\cdots,k)$ using wavelet package at five levels. At the fifth level, we get 32 decomposition coefficients d_5^n, $n=0,1,\cdots,32$.

Secondly, reconstruct wavelet packet coefficients and extract every frequency range signal. The reconstructed signal by coefficients d_5^n is denoted as $\{x_i\}_5^n$, then $\{x_i\}$ can described as below

$$\{x_i\}=\{x_i\}_5^0+\{x_i\}_5^1+\cdots+\{x_i\}_5^{31}\,. \tag{1}$$

Thirdly, calculate energy of every frequency range. Suppose the energy of $\{x_i\}_5^n$ is $E_5^n(n=0,1,\cdots,31)$, then

$$E_5^n=\sum_{i=1}^{k}\left|x_i^n\right|^2\,. \tag{2}$$

Lastly, WP feature vector, constructed based on E_5^n, is $\vec{A}_{WP}=[E_5^0,E_5^1,\cdots,E_5^{31}]$.

2.2 Feature Vector Construction Based on PSD Analysis

Based on PSD analysis, another feature extraction method is used. The extracted feature is called PSD feature. According to reference [5], the analysis bandwidth of spectrum is less than 1000 Hz. The equation of PSD estimate is given in equation 3,

$$\hat{G}(f)=\frac{1}{N}|X(f)|^2\,. \tag{3}$$

where, $X(f)$ comes from FFT transformation. In order to decrease the random error of PSD estimate, a kind of smoothing process is proposed. After dividing the limited data set into M segments, two neighbours of which overlap 50%, we calculate the mean of every five neighbours' $\hat{G}(f)$, and get average spectrum serial $\bar{G}(f)$.

Usually, FFT uses tens of hundred data, so we will get tens of hundred PSD features. It is not easy to deal with so many features. Also there are many unuseful or redundant features. Hence, it is necessary to do feature reduction. Since the frequency feature of early loosening fault of clamping support is not distinct, it is very hard to point out the feature frequency or feature frequency range. Furthermore, there is no expert knowledge about early loosening fault of clamping support. Therefore, an unsupervised reduction method based on entropy measurement is proposed to realize automatic identification of compact features.

2.3 Feature Reduction Based on Entropy Measurement

When feature samples are numerical-type data, similarity measurement *S* of two *n* dimension samples u_i and u_j can be defined as,

$$S_{ij} = e^{-\alpha D_{ij}} . \tag{4}$$

where D_{ij} is the distance of u_i and u_j,

$$\alpha = -(\ln 0.5)/\bar{D} . \tag{5}$$

$\bar{D}$ is the mean of D_{ij}. D_{ij} is measured as the standard Euclidean distance.

$$D_{ij} = \sqrt{\sum_{k=1}^{n}\left[(u_{ik} - u_{jk})/(\max_k - \min_k)\right]^2} . \tag{6}$$

where $\max_k$ and $\min_k$ represent the standard maximum and minimum value of k-th dimension data.

When feature samples are nominal-type data, similarity measurement S of two samples u_i and u_j can be defined as Hamming distance,

$$S_{ij} = \left(\sum_{k=1}^{n} \left|u_{ik} - u_{jk}\right|\right) \Big/ n . \tag{7}$$

where, if $u_{ik} = u_{jk}$, then $\left|u_{ik} - u_{jk}\right| = 1$; else $\left|u_{ik} - u_{jk}\right| = 0$.

The entropy of a given data set $\{u_i\}$ $(i = 1, 2, \cdots, N)$ is

$$E = -\sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \left[S_{ij} \times \lg S_{ij} + (1 - S_{ij}) \times \lg(1 - S_{ij})\right] . \tag{8}$$

From Equation 8, it is found that when samples are all the same or different from each other, the entropy reaches minimum, when they cannot be discriminated, the entropy reaches maximum. Therefore, if an attribute is removed and the entropy goes down, it means that it is reasonable to remove the selected attribute.

3 Information Fusion Framework Based on Neural Networks

Information fusion (IF) is a data processing process which automatically analyzes and synthesizes the data gathered from various sources under certain rules, then gives right decision and task estimate. In fault diagnosis field, IF is often divided into three levels: data level fusion, feature level fusion and decision level fusion. The later two are studied in this paper.

The general process model of feature and decision fusion is shown in Figure 1. In the first place, we extract different features by using FFT and WP transform from raw data set. Such feature is called rough feature. If rough feature is a high dimension

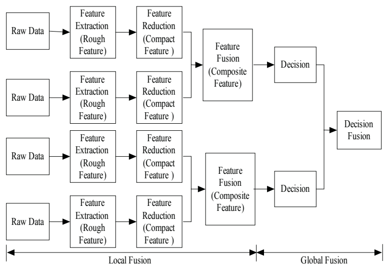

Raw Data
Feature Extraction (Rough Feature)
Feature Reduction (Compact Feature)
Feature Fusion (Composite Feature)
Decision
Decision Fusion
Local Fusion
Global Fusion

Fig. 1. General process model of feature and decision fusion

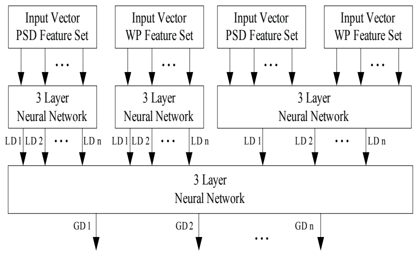


Fig. 2. Flow of feature and decision fusion based on neural networks

data, we must reduce dimension to get the compact feature. After that, we fuse the compact feature to obtain composite feature and get a local decision. Lastly, we fuse all the local decisions to gain global decision. During the whole IF, the process from raw data to composite feature is called local fusion, the rest is called global fusion.

Two neural networks, one for WP feature and the other for PSD feature, are used to make local decisions. A third neural network is used to fuse WP and PSD features and make a local decision. At last a fourth neural network is used to fuse three local decision and make a global decision. Hence four neural networks are needed to implement feature and decision fusion, and the implement flow is shown as Figure 2.

4 Loosening Experiment and Diagnosis Analysis

4.1 Broad Band Random Vibration Experiment for Clamping Support

The cross section diagram and 3D sculpt of clamping support are shown in Figure 3 and Figure 4. Clamping support is composed of a base, 2 blot-braces, a wrapper, 2 half steel-tubes and 24 bolts. On the bottom, the two rectangle blot-braces (3) are fixed on the base (1) by the 12 bolts (2), each side 6. On the top, the blot-braces are fixed with cylindrical wrapper (5) by 12 bolts (4), also each side 6. In application, the bolts connected with the base are always tightened. Then in experiment we tighten and loosen the 12 blots connected wrapper with blot-brace to study the response of clamping support under vertical broad band random excitation.

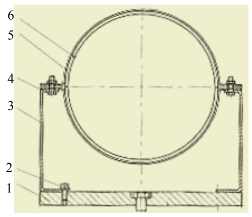


Fig. 3. Cross section diagram of clamping support

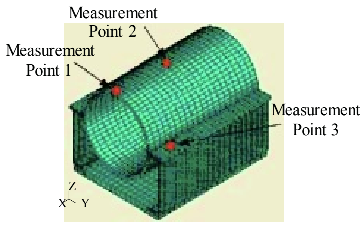


Fig. 4. 3D sculpt of clamping support and distribution of measurement points

The random vibration experimental system includes vibration controlling subsystem and data acquisition subsystem. The former is composed of sensor, charge amplifier, low pass filter, DP550 vibration controller, and power amplifier. The latter is composed of sensors, charge amplifier, low pass filter and data acquirer. The data acquisition subsystem has three sensors. They are placed at measurement point 1, 2, 3 shown in Figure 4.

The frequency range of random vibration experiment is set to [0-1000] Hz. The sampling frequency is 2560 Hz. In order to simulate early loosening of clamping support at different degrees, seven kinds of working conditions are tested. They are: ① 12 bolts (the ones connect wrapper and blot-braces) are all tightened, ② 2 bolts closest to test point 3 are loosening, ③ 2 bolts diagonal to test point 3 are loosening, ④ 4 bolts at the four corners are loosening, ⑤ 4 bolts close to the ones in working condition 4 are loosening, ⑥ 4 bolts at corners are tightened, the left 8 bolts are loosening, ⑦ all the 12 bolts are loosening. The seven working conditions are viewed as seven early faults and marked as F-1, F-2, F-3, F-4, F-5, F-6 and F-7.

4.2 Feature Extraction and Reduction

We use db1 wavelet packet to decompose time data at 5 levels, then reconstruct signal and calculate energy to get the 32-dimension WP feature vector. On the other hand, We make FFT of time data and compute PSD as equation 3 with smoothing technique. After feature reduction based on entropy measurement, we get 32-dimension PSD feature vector. Doing experiments and processing data many times, we abtain 30 groups of feature samples, 2 thirds of which are used in neural network learning, the left are used in diagnosis testing.

4.3 Information Fusion and Discussion

An information fusion (IF) neural network and two single feature (including WP feature and PSD feature) neural network are used to make diagnosis analysis and the results of three networks are compared. The single feature neural network has three layers: input layer, hidden layer and output layer. They have 32, 48 and 7 nodes respectively. IF network is composed of two single feature networks, feature fusion network and decision fusion network. The single feature networks also have three layers with 32, 48 and 7 nodes respectively. Feature fusion network is a three-layer net with 64, 100 and 7 nodes. Decision fusion network is also a three-layer net with 21, 30 and 7 nodes. All the neural networks use back-propagation net. Sigmoid function is selected as transfer function. The training parameters are: goal=1e-3, epochs=2000, learning rate=0.05. Twenty group exemplars are used in training, the other 10 are used in testing. At the same time, in order to compare robustness or antinoise performance of the three networks, the 10 exemplars adding noise are analyzed again. The minimum Signal-to-Noise Ratio (SNR) tested is about 6 dB. The results of three networks are list in Table 1.

Table 1 shows that the result of IF neural network is better than that of single feature network. When the exemplars have no noise, the correct diagnosis ratios of six faults (except fault 2) given by IF NN is higher than the other two NNs. The total diagnosis ratio given by IF NN is 94.3%, higher than 90% and 47.2%. When the exemplars have noise, the correct diagnosis ratios given by IF NN are also the highest. It reaches 88.6%, higher than 81.5% and 46.4%. From the figure, it also can be seen that the antinoise performance of IF NN is the best.

Table 1. Comparision of the results of three networks

	Correct ratio with no noise(%)								Correct ratio with noise(%)							
	F-1	F-2	F-3	F-4	F-5	F-6	F-7	total	F-1	F-2	F-3	F-4	F-5	F-6	F-7	total
WP feature NN	60	10	0	40	90	40	90	47.2	35	10	0	50	100	30	100	46.4
PSD feature NN	100	100	70	90	90	90	90	90	90	70	75.5	90	70	85	90	81.5
Information fusion NN	**100**	**90**	**90**	**90**	**100**	**100**	**90**	**94.3**	**100**	**90**	**60**	**75**	**95**	**100**	**100**	**88.6**

The reason why performance of IF NN is better than others can be interpreted as following. Different feature has different fault mapping ability. For example, WP feature is fit for diagnose F-5 and F-7, but bad for other 5 faults. However, PSD feature is fit for diagnose F-1, F-4, F-6 and F-7, and bad for other 3 faults. When the two features are fused, the composite feature takes on every feature's advantages and decreases their disadvantages. Hence the fusion network has the best performance.

5 Conclusions

Considering the disadvantage of local NDE technique, a novel global NDE technique based on information fusion is proposed to diagnose loosening fault of clamping support. And a hierarchical neural network structure is put forward to implement feature fusion and decision fusion. The experimental results verify that the global NDE method is effective and it can diagnose most of the early loosening faults. Further work will be focused on the incremental learning of neural networks.

References

1. Chen, C.Z. et al., Structure Damage Detection and Intelligent Diagnosis. Science press, (2001)
2. Doebling, S.W., Farrar, C.R., Prime, M.B.: A Summary Review of Vibration-based Damage Identification Methods. The Shock and Vibration Digest, **30** (1998) 91-105
3. Sohn, H., Farrar, C.R.: Damage Diagnosis Using Time Series Analysis of Vibration Signals. Smart Materials and Structures, **10** (2001) 1-6
4. Robertson, A.N., Farrar, C.R., Sohn, H.: Singularity Detection for Structural Health Monitoring Using Holder Exponents. Mechanical Systems and Signal Processing, **17** (2003) 1163-1184
5. Kwanghee, S.L.: Diagnosis of Rotating Machines by Utilizing A Backpropagation Neural Net (1992)

Insulating Fault Diagnosis of XLPE Power Cables Using Multi-parameter Based on Artificial Neural Networks

Xiaolin Chen[1], Yonghong Cheng[1], Zhelei Zhu[1], Bo Yue[2], Xiaojun Xie[1]

[1] State Key Laboratory of Electrical Insulation for Power Equipment,
Xi'an Jiaotong University, Xi'an, Shaanxi 710049, China
xlchen@mail.xjtu.edu.cn
[2] Graduate School at Shenzhen, Tsinghua University, Shenzhen, Guangdong 518055, China

Abstract. An online monitoring system of XLPE power cables was introduced in the research at first. It could detect the parameters, including partial discharge, dielectric loss, and central insulation resistance and sheathing resistance. The BP artificial neural networks were applied to diagnose the insulating status of XLPE cables using the 16 parameters. The adopted transfer functions in the neural networks were hyperbolic tangent function and S-type function. In order to reduce the training time, the Levenberg-Marquardt training method was used. The experimental results showed that the BP artificial neural networks could be applied in fault diagnosis of XLPE power cables using multi-parameter and when the number of nerve unit in the implied layer was fourteen, the output error was least.

1 Introduction

High voltage cables are important in the power system. With the great development of HV and EHV technique, XLPE power cables have been applied more and more widely because of their wonderful properties. However, when they are in service, there may be impurity, semiconductor projection, and space charges accumulating, which will result in the convergence of electrical field. The electrical tree will grow under the condition of electrical field convergence [1]. Moreover, there always is humidity in the environment of XLPE cables, and the water tree can develop with humidity in the operating condition. The treeing ageing is the dominating factor inducing the insulator breakdown.

Much research has been carried on insulating monitoring technique of XLPE cables. The previous research has showed that the treeing aging condition could be revealed by some insulating parameters. For example, when the aging tree grows in XLPE cables, partial discharge will occurs in the insulation. There is close relationship between the partial discharge parameters and the aging status of XLPE insulator. Moreover, the central insulation resistance and sheathing resistance will descend and the dielectric loss will increase during the tree aging in XLPE insulation.

By far, there have been lots of monitoring methods for XLPE cables, including detecting partial discharge, dielectric loss, and so on. However, it was only one kind of insulating information that could be obtained by the existing monitoring devices, and there is impossible to diagnose objectively the insulation condition by it [2][3].

A multi-parameter monitoring system of XLPE cables has been developed in this research. This monitoring device can detect the parameters of partial discharge, dielectric loss, AC leakage, insulating resistance of XLPE and sheath of cables, and so on. These parameters will be calculated by artificial neural network and fuzzy mathematics to obtain the comprehensive diagnosis results. This method can make full use of all the parameters detected by the system and could deduce the diagnosing results more objectively.

2 Experimental Setup

2.1 Measurement System

The online monitoring system introduced in our research consists of main insulator detecting part and sheath detecting part. The former can detect the partial discharge, dielectric loss, and the DC insulating resistance. The latter can obtain the DC insulating resistance of sheath. Fig.1 shows the schematic diagram of the comprehensive monitoring system of XLPE power cables.

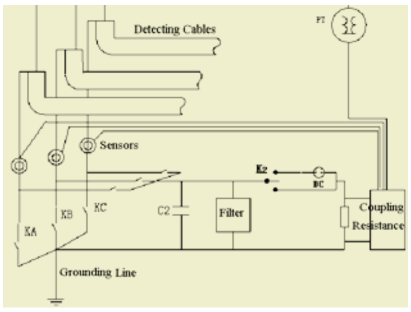


Fig. 1. Schematic diagram of the comprehensive monitoring system of XLPE power cables

The partial discharge signals are coupled by a sensor, which is mounted in the ground line of XLPE cables. In order to sample and process the signals, the output of sensor will be transferred to a industrial computer. The partial discharge signals will be calculated with the statistic-analysis technique. In this method, there will be the following characteristic parameters: max and mean amplitude, PD number, skewness, kurtosis, and so on.

The sensor for detecting AC leakage current is mounted in the ground line of cables, too. In order to obtain the dielectric loss, the voltage from the same phase electrical equipment as the current has to be detected synchronously. The voltage signal is captured by voltage divider and then it is transferred to computer. The phase difference will be calculated by software and its tangent function output will be dielectric loss value. Therefore, the parameters of dielectric loss, phase difference of loss, and the leakage current of XLPE insulator can be obtained by this detecting process. In order to measure the resistance of central insulation and sheathing insulation, a DC voltage under 50V is applied and the DC current produced by this voltage is detected. Then, the resistance value can be calculated by Ohm's law. The applied voltage to central insulator is 50V, while that to sheathing insulator is 15V.

2.2 Detecting Object

The bundling types of XLPE insulator in cables include three-phase bundling together and each-phase bundling. There are three different ground connection modes according to the length of cables. If the cable is shorter than one kilometer, the single ground connection will be adopted. If the cable is longer, the alternative interconnection will be used. While the cables are placed at the bottom of sea, the continuous interconnection will be applied.

The monitoring system introduced in our research is only fit for the each-phase bundling cables of single ground connection due to the detecting mode, because the signals of both partial discharge and leakage current are obtained from the ground line. And if the detecting object is three-phase bundling together cable, the signals from the ground line cannot indicate the insulation information of each phase. Moreover, if the grounding mode is alternative, the same question will come out.

3 Insulating Fault Diagnosis with BP Artificial Neural Network

3.1 Theory of BP Artificial Neural Network

The BP neural network is a kind of one-way transferring forward network with multilayer. Fig. 2 shows a structural diagram of three-layer BP neural network. This network has not only input and output nodes, but also one-layer or multi-layer implied nodes. In addition, there is not any coupling mode among the nodes located in a same layer.

The training process of BP neural network consists of forward propagation and backward propagation. In forward propagation process, the neural cells can only affect that of next layers. If the output results are not expected, i.e. there is great error between the real output and the expected result, the propagating process will change to the backward propagation. In order to reduce the output error, the weight of all layers will be changed in order that is from the output layer to the input layer. And then, the forward propagation will be adopted again. And these two processes will repeat again and again till the error between the real output and the expected result is less than the expected one. The above procedure is named the training process of neural network.

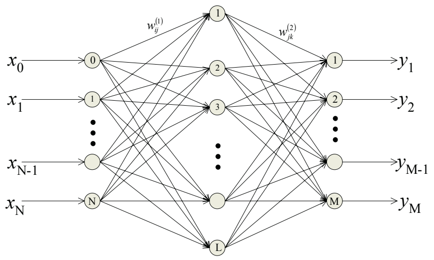


Fig. 2. Structural diagram of three-layer BP neural network

The number of nodes of input layer, implied layer and output layer is respectively N, L and M. The threshold of each node is equal to the weight of input (=1). The weights between the input layer and implied layer are signed wij(1) (i=0,2,3,…,N:

j=1,2,…,L), while the weight between the implied layer and output layer are wjk(2) (j=0,1,…,L: k=1,2,…,M). Then, the output of the implied layer nodes (Ij) and the output layer nodes (yk) are shown in Eqs. (1) and (2), respectively,

$$I_j = f\left(\sum_{i=0}^{N} w_{ij}^{(1)} x_i\right) \; (j=1,2,\ldots,L) \; w_{0j}^{(1)} = \theta_j, \; x_0=1 \tag{1}$$

$$y_k = f\left(\sum_{j=0}^{L} w_{jk}^{(2)} I_j\right) \; (k=1,2,\ldots,M) \; w_{0k}^{(2)} = \varphi_k, \; I_0=1 \tag{2}$$

We can assume that there is a training sample assemble named $p(p=1,2,3,\ldots,q)$, its expected output is $d_{\mathrm{pk}}(k=0,1,2,\ldots,M)$, while the real output is $y_{\mathrm{pk}}(k=0,1,2,\ldots,M)$, then the error of network can be defined $\varepsilon_{\mathrm{pk}}=d_{\mathrm{pk}}-y_{\mathrm{pk}}$. The error quadratic sum of all cells located in the output layer is given by

$$E_p = \sum_{k=1}^{M} \varepsilon_{pk}^2 \tag{3}$$

In fact, the training procedure of BP neural network is the process of adjusting the connecting weight and making the E_{p} small as possible in response to an algorithm. According to LMS steepest descent algorithm and inertial adjusting strategy, the adjusting value of weight is

$$\Delta w_{jk}^{(2)}(t+1) = \eta \sum_{p=1}^{q} (d_{pk} - y_{pk}) y_{pk} (1 - y_{pk}) I_{pj} + \alpha \Delta w_{jk}^{(2)}(t) \tag{4}$$

$$\Delta w_{ij}^{(1)}(t+1) = \eta \sum_{p=1}^{q} \left[\sum_{k=1}^{M} (d_{pk} - y_{pk}) y_{pk} (1 - y_{pk}) w_{jk}^{(2)} I_{pj} (1 - I_{pj}) \right] x_{pi} + \alpha \Delta w_{ij}^{(1)}(t) \tag{5}$$

where η is the training step amplitude, α is the inertia coefficient, t is the training time sequence.

If the weights are given stochastically, there must be error between the real output and the expected result. BP neural network can adjust the weight to make the error small as possible by training.

To diagnose the insulating status of XLPE power cables, the nonlinear mapping relationship of the detecting parameters of insulation and the insulating status must be made firstly. In our research, a three-layer BP neural network was applied. S-type function has the nonlinear property of sorting. Moreover, it has the differentiability of fulfilling the Least Mean Square (LMS) algorithm [4]. Therefore, the transferring function in the neural network was S-type function in our research.

3.2 Input and Output Parameters of Neural Network

The input nerve cells of neural network are the 16 parameters detected by the online monitoring system for XLPE power cables. These parameters include the partial discharge statistical parameters, the dielectric loss parameters and insulating resistance. The number of the first kind parameters is eleven, such as the maximum partial discharge amplitude (Q_{max},), the mean partial discharge amplitude (Q_{mean}), partial discharge number (Q_{num}), and their skewness and kurtosis in each positive and negative half cycle of voltage signals. The number of dielectric loss parameters is three, including the dielectric loss tangent, its angle difference and the leakage current amplitude of main insulation. The insulating resistance includes two parameters (main insulation and sheathing insulation). All the input parameters of neural network is shown in Table 1.

Table 1. The 16 input parameters of neural network

1-4	5-8	9-11	12-14	15-16
$H_{qmax}(\varphi)$的Sk^+, Sk^-, Ku^+, Ku^-	$H_{qn}(\varphi)$的Sk^+, Sk^-, Ku^+, Ku^-	Q_{max}, Q_{mean}, Q_{num}	$tan\delta$, δ, I_L	R_1, R_2

The output parameters of neural network are the four insulation statuses of XLPE power cables, i.e. Excellence, Good, Passed, Bad. “Excellence” and “Good” mean that the power cables are in good condition of insulation and they can be in service without any insulating problem. “Passed” means that there may be some certain symptoms in insulation of cables despite they can run. In this insulation status, much attention must be paid in the objective cables. If permitted, the further detection had better to make. “Bad” means that the insulation of cables is aging seriously and they can not be in service any longer.

3.3 Training and Testing of Neural Network

The nonlinear multiple transferring function applied in our research is a hyperbolic tangent one that simulates the nonlinear inference process. The transferring function of output layer is a logarithmic S-type one. And its output locates in the range from o to 1, agreeing to the degree of membership. Fig.3 shows the transferring functions.

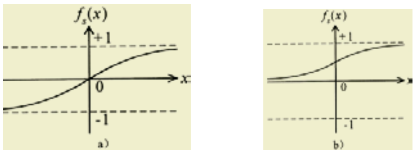
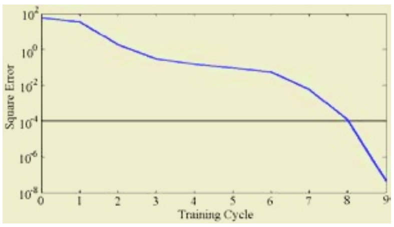


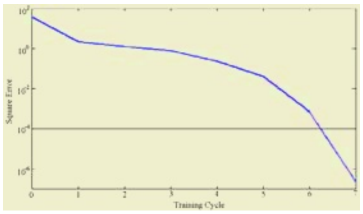
Fig. 3. Transferring function of neural network; a) Hyperbolic tangent type, b) logarithmic S type

It is not easy to determine the number of the nerve cells in implied layer, because there are not any rules that can be applied. In our research, eight, ten, fourteen, twenty and twenty-eight nerve cells in implied layer was analyzed and compared, respectively. The defined square error in the training process was 0.0001. The experimental results showed that the calculation of network was not convergent if the number of nerve cells was less than eight. In the condition of that the sixteen detected parameters were the input and the four insulation statuses were the output of the network, the neural network was trained. Fig.4 shows the errors of different nerve cells of implied layer during training.

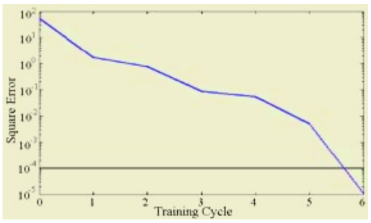
Although the defined square error in the train was very small (0.0001), the number of training cycle was nine at most with the Levenberg-Marquardt fast training algorithm, as shown in Fig.4. It also shows that the training time didn’t change greatly when the number of nerve cells were adjusted.



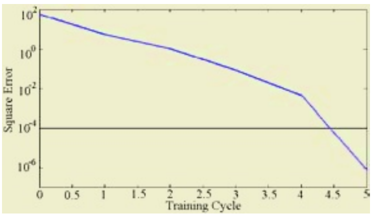
a) N=8



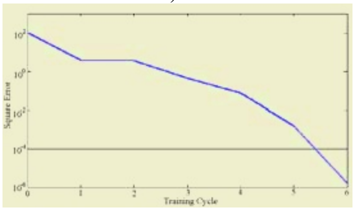
b) N=10



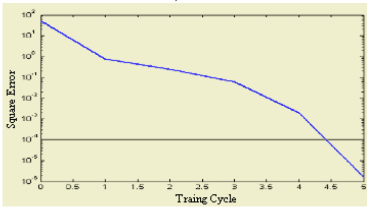
c) N=14



d) N=20



e) N=24



f) N=28

Fig. 4. Training error of the different number of implied nerve cells

Table 2 shows the testing result of different training samples. When the number of nerve cells in implied layer is equal to fourteen, the error is the least. When the number of them is less than eight, the calculation cannot come into convergent. And when the number of them is bigger than fourteen, the error will increase gradually. Therefore, in order to obtain the least error of output, the number of nerve cells in implied layer is equal to fourteen.

Table 2. The testing result of different training samples

8	10	14	20	24	28
-0.009	0.002	0.19966	-0.24	-0.06257	0.018251
-31.527	-72.499	-11.055	-63.6	-100.18	-64.457
-42.584	-17.414	-10.586	-11.95	-19.801	7.5895
14.845	13.022	2.0043	0.588	4.575	6.3139
35.183	16.264	6.6316	-10.63	-11.585	-49.106
9.2462	-33.562	4.4997	-10.83	-29.634	-26.583
51.938	-23.587	31.671	32.98	-2.0002	-1.9783
20.969	29.883	14.543	43.37	21.287	18.967
9.5342	-20.438	1.3234	7.2	-16.261	-0.9385
21.32	16.3	0.19966	13.68	9.385	20.1

4 Conclusion

In order to diagnose the insulation status of XLPE power cables objectively, the multi-parameter should be taken into account sufficiently. And the BP neural network can meet this challenge. There are sixteen parameters detected by the monitoring system for XLPE cables. They are the input of the neural network, and when the number of nerve cells is fourteen in implied layer, the error of output is least.

References

1. Cheng, Y.: Detection and Diagnosis of Power Equipments. Power Press, Beijing (2001)
2. Gish, N.B., Howson, P.A., Howlett, R.J.: Condition Diagnostics of a Physical Breakdown Mechanism in High Voltage Dielectrics Utilizing AI Evaluation Techniques. Fourth International Conference on knowledge-based Intelligent Engineering Systems & Allied Technologies, Brighton, Uk, (2000) 644-650
3. Gulski, E.: Computer-aided Recognition of Partial Discharges Using Statistical Tools. Delft University Press (1991)
4. Salama, M.M.A., Bartnikas, R.: Determination of Neural network Topology for Partial Discharge Pulse Pattern Recognition, IEEE Transactions on Neural Networks, **13** (2002) 446-456

A Hybrid Method and Its Application for Power System*

Xusheng Yang[1,2], Yong You[1], Wanxing Sheng[3], and Sunan Wang[1]

[1] Xi'an Jiaotong University, Xi'an , Shaanxi 710049, China
yxshlz@163.com
[2] Lanzhou Jiaotong University, Lanzhou, Gansu 730070, China
[3] Electric Power Research Institute, Beijing 100085, China

Abstract. The precision of short-term load forecasting for electric power systems directly affects the economic benefit of power systems. A hybrid method based on chaos and neural network was used in the study of the electric power system short-term load forecasting. This article presents the application of chaos method to reconstruct attractors in phase spaces and a multi-layer feed forward neural network to fit the attractor's global map, to construct a hybrid prediction model. Moreover, this article shows the efficiency of a noise-suppressing method based on single value decomposition (SVD), and a new Neural Network learning algorithm, chaotic learning algorithm, is proposed. The result indicates that the method is effective.

1 Introduction

Load forecasting is always the essential part of an efficient power system planning and operation. Several electric power companies have adopted the conventional methods for forecasting the future load [1]. In conventional methods, the models are designed based on the relationship between load power and factors influencing load power. The conventional method has the advantage that we can forecast load power with a simple prediction model. However, since the relationship between load power and factors influencing load power is nonlinear, it is difficult to identify its nonlinearity using conventional methods.

Recently, several methods have been reported for load forecasting, such as time series method [2], gray theory [3], Least Square Method [4] and Neural Networks [5] etc. Because load power is multi-dimensions and nonlinear chaos, these methods have some limitations in applicability.

Chaos is produced by determinate rules. In the period decided by the most plus Lyapunov exponent, chaos forecasting methods using chaos determinacy theory are better than those forecasting methods that systems are regarded as simple stochastic ones. This article presents the use of chaos method to reconstruct attractors in phase spaces and a multi-layer feed forward neural network to fit the attractor's global map, to construct a hybrid prediction model.

* This work was supported by 'Qing Lan' Talent Engineering Funds by Lanzhou Jiaotong University and the special fund for major state basic research projects. Project No.: G1998030405

2 Hybrid Forecasting Method

Short-term load power is a nonlinear chaos system influenced by some factors such as weather, special days and so on [6]. The chaos time series forecasting method is essentially a time series forecasting method of nonlinear system and its series can be reconstructed phase spaces and keep most of characteristics. The chaos time series forecasting method does not require the time series to be completely strict chaotic.

The method reconstructing phase spaces from one dimension load power data:

First, use FFT (Fast Fourier Transform) to get the average period T of one dimension data from time series data. Then calculate time window w, and use following equations: $\tau=t\Delta t$, $\tau_w=(m-1)\tau$ to calculate time delay τ and embedding dimension m [6],[8].

If load power time series data is: $\{x(t_i), i=1, 2, \ldots, n\}$, reconstructing phase spaces are:

$$X_i(t)=\left(x(t_i),x(t_i+\tau),\ldots x(t_i+(m-1)\tau)\right). \quad i=1,2,\ldots M \tag{1}$$

Where $X_i(t)$ is the point of reconstructing phase spaces, m is embedding dimension, τ is time delay, M is the number of points in phase spaces and $M=n-(m-1)\tau$.

Matrix of attractors in phase spaces is:

$$\boldsymbol{N}=[X_1(t)^T, X_2(t)^T,\ldots, X_M(t)^T]^T \tag{2}$$

Where every row vector is a state point of reconstructing phase spaces, and attractors is constructed by contiguous state points.

If the mapping of system dynamics is F,

$$X_{i+1}(t)=\boldsymbol{F}(X_i(t)),\ n=1,2,\ldots M \tag{3}$$

Neural networks are composed of simple elements operating in parallel. These elements are inspired by biological nervous systems. In nature, the network function is determined largely by the connections between elements. We can train a neural network to perform a particular function by adjusting the values of the connections (weights) between elements. Because of its high ability of function approximation, neural network has been trained to perform the mapping function of equation (3).

To obtain the predicted value $x(t_k)$ from the time series data, we define $j=k-(m-1)\tau$ and regard $X_{j-1}(t)$ as the input of neural network and $X_j(t)$ the output after reconstructing phase spaces , then we train neural network to obtain output $X_j(t)$. The last vector of $X_j(t)$ is the predicted value $x(t_k)$.

3 Filtering Method Based Singular Value Decompose (SVD)

The data of actuarial measurement is influenced by a variety of noise. Noise obscures the clarity of a signal and it causes attractors to be diffusive in phase spaces. Therefore the attractor can't be smooth, which does harm to observe. A filtering method based SVD (Singular Value Decompose) is proposed to weaken noise's effect.

Because there is a special SVD function in MATLAB and because of space of this article, the filtering method is briefly introduced.

Do SVD to the matrix *N*, which is the attractor matrix of reconstructing phase spaces, in equation (2).

$$N=U*S*V^T \cdot \qquad (4)$$

where $U \in \mathbf{R}^{M \times M}$ and $V \in \mathbf{R}^{m \times m}$ are orthogonal matrix, *M* is the number of rows of the attractors matrix, *m* is the number of columns and is also the embedding dimension of phase spaces. For $m < M$, the $m \times m$ order submatrix is diagonal matrix dia$g(\delta_1,\delta_2,\ldots,\delta_m)$. We reserve the higher-order singular value and inverse equation (4) to obtain *N**. *N** is the matrix after using Filtering Method based *(r/m)* SVD. Then we average the same data points and gain the data after SVD filtering.

4 Chaos Learning Algorithm

Because chaos action has some characteristics such as ergodicity, regularity and randomicity, we can optimize parameters of Neural Networks using a kind of Chaos Learning Algorithm (CLA). A chaos mapping is utilized to do iterative search. Equation (5) is the chaos mapping:

$$x_{n+1}=\sin(2/x_n) \quad n=1,2,3,\ldots \qquad (5)$$

The nature of chaos makes it possible that the Chaos Learning Algorithm can jump out the trap of locally optimal solutions to find the globally optimal solution. Therefore the CLA overcomes shortcomings of BP algorithm. Moreover, we use a new method of reducing search space step by step and combine gradient method in CLA to increase convergence rate.

The variables of the optimization problem are the values of the connections (weights) and the threshold values. The sample is [*Xk, Dk*], *k*=1, 2, …, *P,* where *Xk* is the input, and *Dk* is the desire of samples. And the actual output of every *Xk* is *Yk*. The Sum Squared Error is:

$$E = \overset{P}{\underset{k=1}{}} (Y_k - D_k)^2. \qquad (6)$$

And equation (6) is the object function of NN optimization problem.

The values of the connections (weights) *W* and the threshold values *B* are mapped into spaces of chaos variables and a new method of reducing search space step by step is used to search optimum.

If there are *k* variables include connections (weights) and threshold, and *f* is the value of the objective function of chaos optimization problem. *X** is the variable with max fitness and *f** is the max fitness now. *N* is a certain large positive integer but not too large to finish on time.

The algorithms are described in the following diagram.

5 An Example of Short-Term Load Forecasting

A load note of a certain 10kv distributing net in Beijing and its data are adopted as the example. The historical load data is used to validate the method that this article proposed.

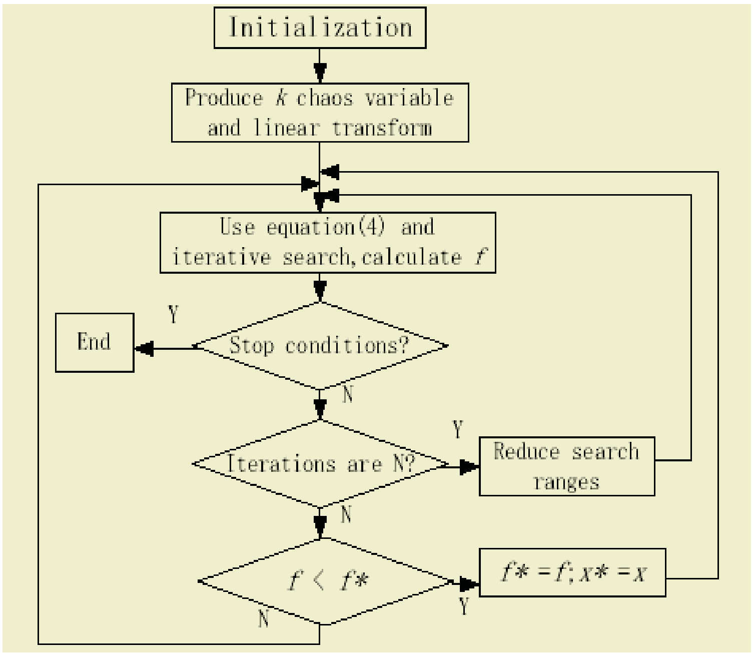


Fig. 1. Chaos Optimization Algorithm for training Neural Networks

The number of the test data, including active load data and reactive load data, is total 24×55×2=2640. Fig. 2 is the observed value of active load. The first 960 data points, representing the data from Jul.1 to Aug.9, are used to reconstruct phase spaces and train Neural Networks as sample. The last 360 data points are used to test. Based on the method using chaos and Neural Networks, the embedding dimension m and the time delay τ of this example are obtained. Moreover, to see the shape of attractors visually, the embedding dimension m is selected 3 when we protract curves of attractors. Fig.3 (a) and Fig.3 (b) describe the attractors. The figures show that the attractors become smooth after filtering.

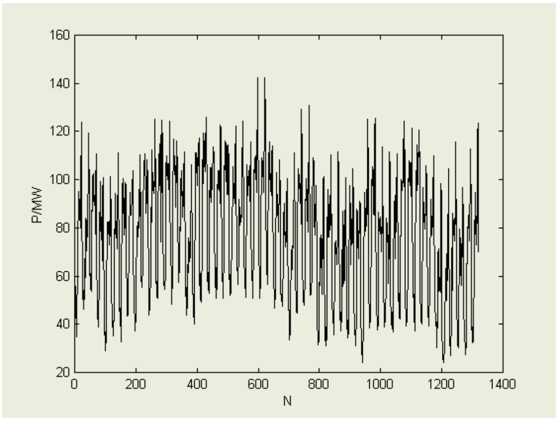
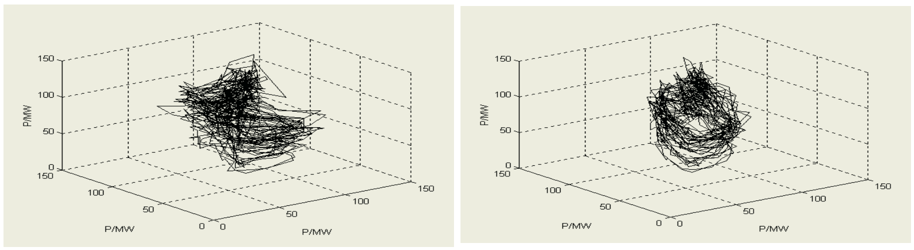


Fig. 2. Actual load data of a certain node (Jul.1-Aug.24, 2000)



(a) Shape of attractors before filtering (b) Shape of attractors after filtering

Fig. 3. The change of reconstruct attractors using noise-suppressing method based on single value decomposition (SVD)

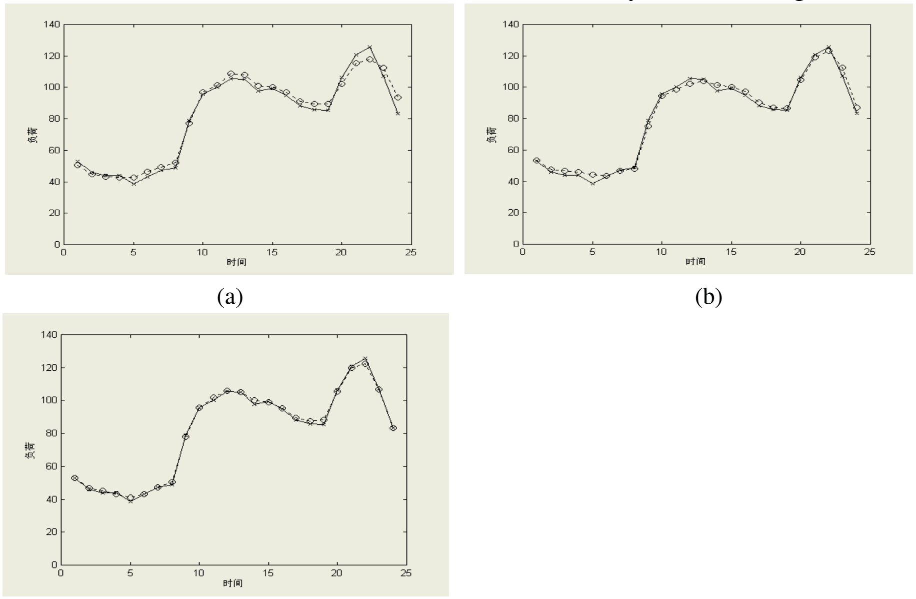
The number of input units is the same as the number of embedding dimension. Through examination, the number of hidden units is selected suitably so that input information can be used enough and the weakness such as over training and poor generalization ability can be overcome. The Neural Networks model is trained by the Chaos Learning Algorithm proposed in this article and compared with 24×3×24 Neural Networks model.

Average error is defined as equation (7).

$$E_M = \frac{1}{24}\sum_{i=1}^{24}\frac{|Y_i - D_i|}{D_i}. \tag{7}$$

Where Y_i are the forecasting data and D_i are real data.

Fig.4 is the result of the example. The real line "—×—" is constructed by the data of actual measurement and the broken line is constructed by the forecasting data.



(a) (b)

(c)

Fig. 4. The results of forecasting load data using different methods

Fig.4 (a) shows the result of 24×3×24 Neural Networks model. The error is EM=4.52%. The result of the model using chaos and Neural Networks proposed above but not using Filtering Method based SVD is exhibited in Fig.4 (b). EM =3.24% is obtained. And the result of the hybrid method using chaos method to reconstruct attractors in phase spaces, a multi-layer feed forward neural network to fit the attractor's global map and Filtering Method based SVD can be seen in Fig.4 (c), EM=1.77%. The results and their intercomparson indicate that the method using chaos and NN is effective.

In the examination, a phenomenon was found. The forecasting precision EM reduced along with the increase of time. So the phase space should be reconstructed again in order to meet demand of error.

6 Conclusions

In this article, load observations are used to reconstruct phase spaces and map the attractors of phase spaces to obtain predicted value. Because Neural Networks can fit this nonlinear mapping, a hybrid prediction model using chaos and Neural Networks is proposed. In the pretreatment of load data, a filtering method based SVD is adopted in order to eliminate noise. Moreover, Chaos Learning Algorithm is used to advance learning speed and precision. The hybrid prediction method is applied in a short-term load-forecasting example. The result indicates that the proposed method is better than NN prediction method. The electric power system short-term load-forecasting model using chaos and neural networks is feasible and effective.

References

1. Haida, T., Muto., S.: Regression Based Peak Load Forecasting Using a Transformation Technique [J]. IEEE Trans on Power System, **9** (1994) 1788-1794
2. Charytoniuk, W., Chen, M S, Van Olinda, P.: Nonparametric Regression Based Short-Term Load Forecasting. IEEE Trans on Power System, **13** (1998) 725-730
3. Akagi, H.: New Trends in Active Filter for Power Conditioning. IEEE Transaction on Power Electronics, **32** (1996) 1312-1322
4. Mastorocostas, P.A., Theocharis, J.B., Bakirtzis, A.G.: Fuzzy Modeling for Short-Term Load Forecasting Using the Orthogonal Least Squares Method. IEEE Trans on Power System, **14** (1999) 29-36
5. Chow, T.W.S., *et al.*: Neural Network Based Short-Term Load Forecasting Using Weather Compensation. IEEE Trans on Power System, **11** (1996) 1736-1742
6. Lv, J., Zhan, Y., Lu, J.: The Non-Linear Chaotic Improved Model of The Electric Power System Short-Term Load Forecasting Proceeding of the CSEE, **20** (2000) 80-83
7. Abarbanel, D. I.: Analysis of Observed Chaotic Data. Springer-Verlag, New York (1996)
8. Kim, H S, Eykholt, R, Salas, J D.: Nonlinear Dynamic, Delay Times, and Embedding Windows. Physica D, **127** (1999) 48-60

Fuzzy Neural Very-Short-Term Load Forecasting Based on Chaotic Dynamics Reconstruction[*]

Hongying Yang, Hao Ye, Guizeng Wang, and Tongfu Hu

Department of Automation, Tsinghua University, Beijing 100084, China
yanghy02@mails.tsinghua.edu.cn, haoye@tsinghua.edu.cn

Abstract. This paper proposes an improved fuzzy neural system for very-short-term load forecasting problem, based on chaotic dynamics reconstruction techniques. The Grassberger–Procaccia algorithm and the Least Squares Regression method were applied to obtain the accurate value for the correlation dimension, which is used as an estimation of the model order. Based on that model order, an appropriately structured Fuzzy Neural System (FNS) for load forecasting was designed. Satisfactory experimental results were obtained in 15 minutes ahead electrical load forecasting from the electric utility in Shandong Heze area. And the same experiments using conventional Artificial Neural Network are also performed as a comparison with the proposed approach.

1 Introduction

Short-term load forecasting plays an important role in power system planning and operation. Basic operating functions such as unit commitment, economic dispatch, fuel scheduling and unit maintenance can be performed efficiently with an accurate forecast [1]. Particularly, it is more indispensable to precisely perform very-short-term load forecasting of minutes ahead in order to avoid system's undesirable disturbances [2].

During the last two decades, a wide variety of methods have been proposed, such as regression analysis, exponential smoothing, ARMA model and so on, to solve the load forecasting problem. However, it is still a difficult task as the power system is a nonlinear complicated system which can often mix with the chaotic behaviors. And also, the traditional prediction methods can not always supply the required accuracy for the engineering application because there are only limited history data sets can be obtained and the factors that affect the load forecasting are complex [3].

As an important way to study the characteristics of complicated systems, the interests in chaotic time series prediction have been increased over past few years. There are many chaotic prediction methods, such as local-region method, Lyapunov Exponents method, and artificial neural network method [4], which are based on dynamic reconstruction techniques. However, the prediction precisions are still need to be improved.

In this paper, a new framework employing the fuzzy logic theory combined with dynamic reconstruction techniques are proposed for the very-short-term load forecasting. The proposed method was applied to 15 minutes ahead forecasting and the ex-

[*] The research is supported by the National Natural Science Fund of China and the 863 program of China.

perimental results based on real data illustrated the simplicity and adequate forecasting accuracy of the proposed model. In Section 2, the methods of dynamics reconstruction are introduced to obtain the estimate value of correlation dimension. Section 3 presents the structure of Fuzzy Neural System (FNS). Section 4 reports the experimental results using the Shandong Heze load data. In Section 5, some conclusions are drawn.

2 Reconstruction of Chaotic Time Series

The idea of chaotic dynamics reconstruction techniques stems from the embedding theorem developed by Takens [5]. The theorem regards a one-dimensional chaotic time series as the compressed information of higher dimension. Then, the time series $x(t), t = 1,2,3,...,N$ can be represented as a vector $X(t)$ in a d-dimensional space

$$X(t) = (x(t), x(t-1),..., x(t-(d-2)), x(t-(d-1))) . \tag{1}$$

where d is called embedding dimension of the system. According to Takens' embedding theorem [5], in order to obtain an available reconstruction of the dynamics system, the embedding dimension d must satisfy

$$d \geq 2D+1 . \tag{2}$$

where D is the dimension of the attractor, which is the key point to obtain correct minimum embedding dimension.

2.1 Grassberger-Procaccia Method

Grassberger-Procaccia (GP) algorithm [6] is one of the popular methods for determining the so-called correlation dimension D_m , which is an estimation of the dimension of the attractor D . If the correlation integral $C_d(r)$ is defined as

$$C_d(r) = \frac{2}{N(N-1)} \sum_{i=1}^{N} \sum_{j=i+1}^{N} H(r - |X_i - X_j|) \cdot \tag{3}$$

where H is the Heaviside function and d is the embedding dimension. It is shown that for r sufficiently small, and the number of observed values N sufficiently large

$$D_m = \lim_{r \to 0} \frac{\ln C_d(r)}{\ln(r)} . \tag{4}$$

The algorithm plots $\ln C_d(r)$ against $\ln(r)$ curves through increasing the value of d until the slope of the curve's linear part is almost an invariable. Then correlation dimension D_m can be estimated.

2.2 Regression Correlation Dimension Using Least Square

Although GP algorithm is perfect in theory, it does not allow computation error of correlation dimension estimations [4] in practice. In order to overcome the weakness and improve the computation stability of GP algorithm, Zhao, *et al* [7] proposed the method of regression correlation dimension by least squares. This method first defined x_{ij} and y_{ij} in $\ln(r) \sim \ln C_d(r)$ curve region

$$\begin{cases} x_{ij} = [\ln(r)]_{ij}, \\ y_{ij} = [\ln C_i(r)]_{ij}. \end{cases} \tag{5}$$

where i is the embedding dimension mentioned above and j is the computation point of curve region which filled linear relation [7], then

$$y_{ij} = a x_{ij} - b_i . \tag{6}$$

According to the theory of least squares, the estimated values $\overline{a}$ and $\overline{b}$ can be obtain when the Equation (7) reaches the least. Then, the value of correlation dimension can be computed and the minimum embedding dimension m can be attained.

$$Q(a,b_i) = \sum_i \sum_j [y_{ij} - (\overline{a}x_{ij} + \overline{b_i})]^2 \cdot \tag{7}$$

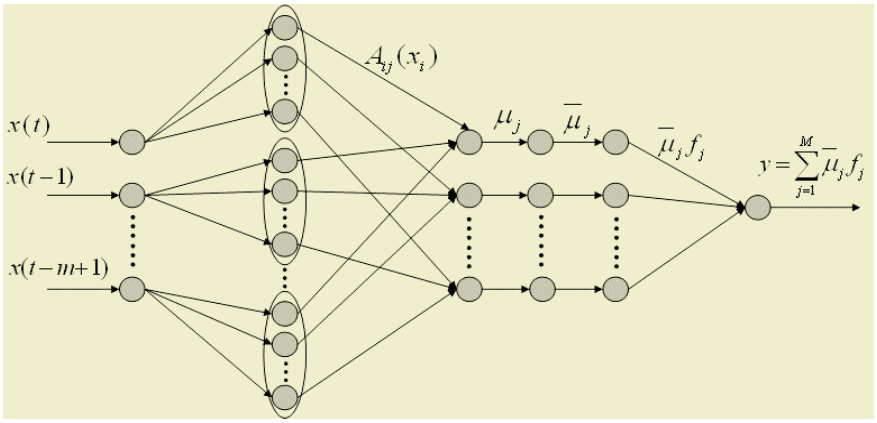


Fig. 1. The structure of Fuzzy Neural System

3 Fuzzy Neural System Model

Fuzzy logic is a research area based on the principles of approximate reasoning and computational intelligence. This method has an advantage of dealing with nonlinear parts of the forecasted load curves [1]. In this study, a TSK (Takagi-Sugeno-Kang) fuzzy model was developed to provide future estimate of the load at the reference point, based on recent variations at historical neighboring points.

The proposed Fuzzy Neural System (FNS) is a MISO (Multiple Input Single Output) system whose structure is shown in figure 1.The input parameters are the load vector data $x(t), x(t-1), \ldots\ldots x(t-m+1)$ whose order is determined by dynamics reconstruction techniques and the input nodes are confirmed by the value of m which is estimated in section 2. The output value y of the model is the load forecasted result of the next step. The FNS consists of linguistic fuzzy rules represented in the following form:

$$\begin{aligned} &IF\ x_1\ is\ A_{1j}\ AND\ \ldots\ldots\ AND\ x_m\ is\ A_{mj}, \\ &THEN\ \ y = f_j = a_{0j} + a_{1j}x_1 + \ldots + a_{mj}x_m,\ j = 1,\ldots,M. \end{aligned} \tag{8}$$

The rule of the first layer corresponds to an available fuzzy set A_{ij} of the inputs. Its output gives the membership value $A_{ij}(x_i)$ of the input x_i to the fuzzy set A_{ij} [8]. In this work the membership function is

$$A_{ij}(x_i) = \exp\left(-\frac{1}{2}\left(\frac{x_i - c_{ij}}{\omega_{ij}} \right)^2 \right). \tag{9}$$

where c_{ij} and ω_{ij} are parameters for each fuzzy set A_{ij}. And the outputs of the second and third layers are μ_j and $\overline{\mu}_j$ (the normalized value of the μ_j).

$$\mu_j = \prod_{i=1}^{m} A_{ij}(x_i). \tag{10}$$

In the fourth layer, $\overline{\mu}_j$ is multiplied by the function f_j of the consequent part of the rule. And the fifth layer outputs the one-step predicted value $\overline{x}(t+1)$ at instant $t+1$.

$$y = \overline{x}(t+1) = \sum_{j=1}^{M} \overline{\mu}_j f_j. \tag{11}$$

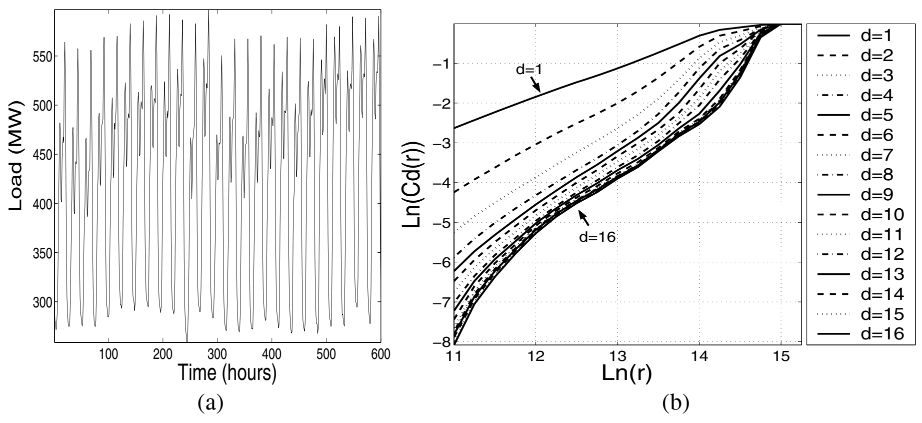


Fig. 2. (a) One part of the load time series supplied in Shandong Heze aera (b) Plot of $\ln C_d(r)$ against $\ln(r)$ by Heze time series data

4 Experimental Results

4.1 Test Data

In this study, the load data of Shandong Heze area in 2004 were used to test the performance of the proposed method. The data were arranged at intervals of 15 minutes (one day, 96 points). Figure 2(a) shows one part of the load and figure 2(b) gives the reconstructed result by GP algorithm. And the computed values using least squares regression correlation dimension are as follows

$$D_m = 1.6529, \ m = 4 . \tag{12}$$

4.2 Very-Short-Term Load Forecasting by the FNS

At each prediction process, the tests were performed using the former two weeks' data (1344 points) as training set to predict one week's (672 points) load. In this search, 2688 samples in March (4 weeks), 2004 are tested. The mean absolute percent errors (MAPE) of each day approximate to 1% and the average MAPE of the whole month is 0.9651%. Table 1 shows absolute percent error (APE) percentage analysis of the last week in March. The table indicates that the APE errors below 3% can reach 98.2%.

Table 1. Error percentage analysis for 15 minutes ahead forecasts (Mar 21-27, 2004)

APE .%	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Average
<1%	71.875	67.708	64.583	64.584	67.709	56.250	60.417	64.732
1%~3%	28.125	31.250	35.417	33.333	30.208	38.542	37.500	33.482
>3%	0.000	1.042	0.000	2.083	2.083	5.208	2.083	1.786

In order to highlight the FNS model process, the prediction of the daily load for a special weekday, e.g. Tuesday is presented as an illustrative example. Figure 3 shows the forecasted results using the FNS. The error curves indicate that the proposed approach has an excellent forecasting accuracy.

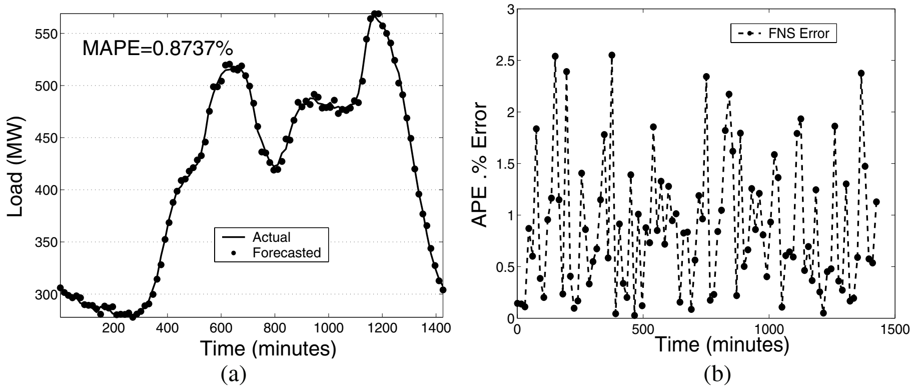


Fig. 3. 15 minutes ahead forecasted results on Tuesday, Mar 23, 2004 (96 points): (a) forecasted load curves (b) The forecasted APE Error

4.3 Comparison with the ANN Approach

In this part, the same data are also processed by conventional BP Neural Network approach as a comparison with the proposed method. Figure 4(a) and figure 4(b) show the error comparison between FNS and ANN methods. And the detailed MAPE values of the figure 4(a) are reported in Table 2. It can be seen from the diagrams that the proposed method improved the estimation accuracy and stability.

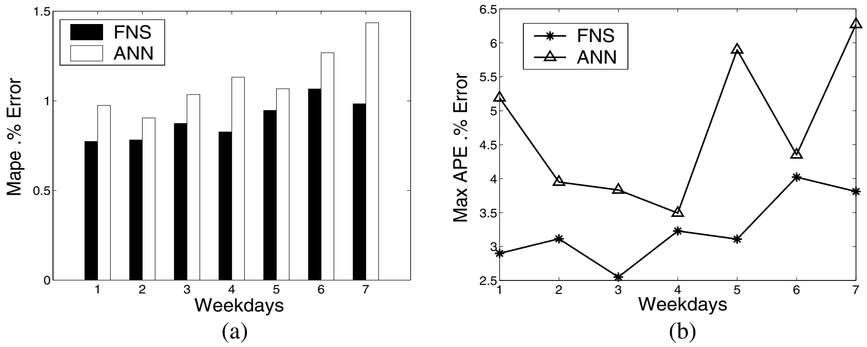


Fig. 4. The error comparison between FNS and ANN: (a) MAPE errors (b) Max APE error

Table 2. The MAPE errors forecasted by the FNS and the ANN (Mar 21-27, 2004)

Method	MAPE. % Error							
	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Average
ANF	0.7729	0.7818	0.8737	0.8267	0.9467	1.0662	0.9836	0.8931
ANN	0.9748	0.9044	1.0351	1.1317	1.0673	1.2681	1.4351	1.1166

5 Conclusions

This paper presented an approach for very-short-term load forecasting problem based on chaotic dynamics reconstruction techniques and fuzzy logic theory jointly. After the model order was estimated by the reconstruction theory, a proper structured Fuzzy Neural System (FNS) was designed. The experimental results demonstrated that the proposed approach can deal with 15 minutes load forecast with good performance. And also, compared with the ANN method, the approach can not only reduce the effect of predicted error, but also improve the prediction stability of the system.

References

1. Senjyu, T., Mandal, P., Uezato, K., Funabashi, T.: Next Day Load Curve Forecasting Using Hybrid Correction Method. IEEE Transactions on Power Systems 20 (2005) 102-109
2. Kawauchi, S, Sugihara, H, Sasaki, H: Development of Very-Short-Term Load Forecasting Based on Chaos Theory. Electrical Engineering in Japan 148 (2004) 55-63
3. Xu, T., He, R.M., Wang, P., Xu, D.J.: Input Dimension Reduction for Load Forecasting Based on Support Vector Machines. IEEE International Conference on Electric Utility Deregulation, Restructuring and Power Technologies, Hong Kong, China 2 (2004) 510-514
4. Camastra, F. and Colla, A.M.: Neural Short-Term Prediction Based on Dynamics Reconstruction. Neural Processing Letters 9 (1999) 45–52
5. Takens, F.: Detecting Strange Attractors in Turbulence. In: Rand D. and Young L-S. (eds.): Dynamical Systems and Turbulence, Warwick 1980. Lecture Notes in Mathematics, Vol. 898. Springer-Verlag, Berlin (1981) 366–381
6. Grassberger, P. and Procaccia, I.: Characterization of Strange Attractors. Physics Review Letters 50 (1983) 346-349
7. Zhao, G.B., Shi, Y.F., Duan, W. F., Yu, H. R.: Computing Fractal Dimension and The Kolmogorov Entropy from Chaotic time series. Chinese Journal of Computational Physics 16 (1999) 309-315
8. Tzafestas, S, Tzafestas, E: Computational Intelligence Techniques for Short-Term Electric Load Forecasting. Journal of Intelligent and Robotic Systems 31 (2001) 7–68

Application of Neural Networks for Very Short-Term Load Forecasting in Power Systems

Hungcheng Chen[1], Kuohua Huang[2], and Lungyi Chang[2]

[1] National Chin-Yi Institute of Technology, Institute of Information and Electrical Energy, Taiping, Taichung, 411, Taiwan, China
hcchen@chinyi.ncit.edu.tw
[2] National Chin-Yi Institute of Technology, Department of Electrical Engineering, Taiping, Taichung, 411, Taiwan, China
{huangkh,lychang}@chinyi.ncit.edu.tw

Abstract. Load forecasting has become in recent years one of the major areas of research in electrical engineering. In a deregulated, competitive power market, utilities tend to maintain their generation reserve close to the minimum required by an independent system operator. This creates a need for an accurate instantaneous-load forecast for the next several minutes. An accurate forecast eases the problem of generation and load management to a great extent. This paper presents a novel artificial neural network (ANN) for very short-term load forecasting. The model with tapped delay line input is simple, fast, and accurate. Obtained results from extensive testing on Taipower System load data confirm the validity of the proposed approach.

1 Introduction

Reliable operation of a power system and economical utilization of its resources require load forecasting in a wide range of time leads, from minutes to several days. Accurate short-term load forecasting (STLF) is essential for planning startup and shut-down schedules of generating units, reserve planning and load management. In addition, the load frequency control and economic dispatch in power system require load forecasts within shorter time leads, from single minutes to several dozen minutes. They are referred to as very short-time load forecasts (VSTLF). These forecasts, integrated with the information about generation cost, spot market energy pricing, transmission availability, and spinning reserving requirements imposed by an independent system operator, are used to determine the best strategy for the utility resources. Very short-term load forecasting has become of much greater importance in today's deregulated power industry.

Many methods have been developed for short-term load forecasting [1],[2],[3]. These methods are mainly classified into two categories: classical approaches [4],[5] and artificial intelligence based techniques [6],[7]. Recently, artificial intelligence based methods using artificial neural networks have been applied to STLF. The main advantage of using neural networks lies in their abilities to team the mentioned dependencies directly from the historical data without the necessity of selecting an appropriate model. There are a few types of neural networks that have been applied for load forecasting. A multilayer feedforward network with one hidden neuron layer is most commonly used [8],[9].

Very short-term load forecasting requires a different approach [10]. Instead of modeling relationships between load, time, weather conditions and other load affecting factors we are rather focused on extrapolating the recently observed load pattern to the nearest future. This paper presents a novel ANN for very short-term load forecasting by the application of ANN to modeling load dynamics. When actual loads are forecasted and used as input variables the proposed model with the tapped delay line input is more simple, fast, and accurate. It is less sensitive to the requirement of having the training data representative of the entire spectrum of possible load and weather conditions. The method has been successfully implemented and tested for on-line load forecasting in Taipower system. The results will testify the availability of the application of artificial neural networks to very short-term load forecasting.

2 The Artificial Neural Networks

An ANN can be defined as a highly connected ensemble of processing elements (PEs) called neurons or nodes. A neuron shown in Fig. 1 is a multi-input-single-output PE consisting of a summation operation and an activation function. PEs can be interconnected in various network topologies and can be globally programmed (trained) for various purposes such as pattern recognition, combinatorial optimization, estimation of sampled function whose form is not known, etc.

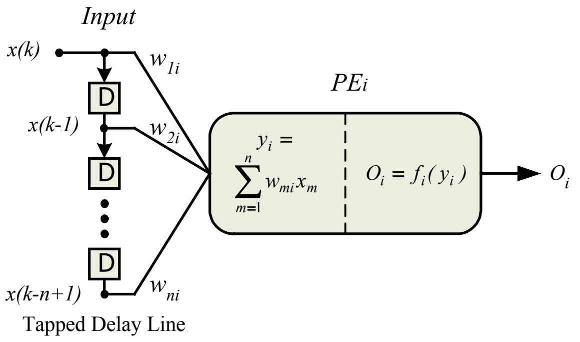


Fig. 1. Processing element with the tapped delay line input

As seen from Figure 1, each PE performs two functions: (i) sum the weighted input signal to the PE, and (ii) produce an output that is a function of the weighted sum. We are rather focused on extrapolating the recently observed load pattern to the nearest future for VSTLF. The input signals enter from the left. At the output of the tapped delay line we have an R-dimensional vector, consisting of the input signal at the current time and at delays of from 1 to n-1 time steps. The actual loads are forecasted and used as input variables. It is less sensitive to the requirement of having the training data representative of the entire spectrum of possible load and weather conditions.

The PEs produce output that is a function of sum of the weighted inputs. The PE function, also called activation function or squashing function, map the unbounded activation (sum of weighted inputs) into a bounded output signal. Among the infinite number of possible PE functions, generally sigmoidal or hyperbolic tangent function is preferred. Linear functions are not preferred because linear functions do not suppress noise. Non-linearity increases computational richness and facilitates noise suppression but also introduces computationa1 intractability as well as dynamical instability. Monotonic continuous nonlinear function appear to be the natures compromise. Most biological neurons have sigmoidal signal characteristics.

3 Implementation of ANN-Based VSTLF

The implementation of the load forecasting system required carrying out several tasks such as selection of ANN architecture, selection of input variables, data normalization, and training of the design networks. These issues are briefly discussed below:

3.1 Selection of ANN Architecture

The three-layer fully connected feedforward neural network is used here. It includes an input layer, a hidden layer and an output layer. Signal propagation is allowed only from the input layer to the hidden layer and from the hidden layer to the output layer. The actual loads are forecasted and used as input variables. The output is the desired forecasting load at 6-min ahead. The number of inputs, the number of hidden nodes, transfer functions, scaling schemes, and training methods affect the forecasting performance and hence need to be chosen carefully.

3.2 Selection of Input Variables

We are rather focused on extrapolating the recently observed load pattern to the nearest future for VSTLF. The input variables are consisted of the actual loads at the current time and at delays of from 1 to 9 time steps.

3.3 Data Normalization

Once the historical data are gathered, the next step in training is to normalize all the data so that each value falls within the range from 0 to 1. This is done to prevent the simulated neurons from being derived too far into saturation.

3.4 Training of the Design Networks

Learning in ANN means change in its parameters based on training data. Learning methods determine how the system parameters change when presented with new samples. Among the different learning methods available, the error back-propagation learning algorithm is a supervised learning method. It is the most popular and almost universally used because of its computational simplicity, ease in implementation and good results generally obtained for large number of problems in many different areas of application. The back-propagation learning algorithm was used as training method here.

4 Test Results

The historical data of a substation load in Taipower system for June 2004 was used for testing the proposed ANN-based VSTLF. A neural network with 10 inputs, 10 hidden node, and one output was used here. The forecaster was trained using the data from recent week and then used to forecast the 6-min ahead load for next week. A tangent sigmoid function was chosen as the transfer function for the hidden layer, and a linear function for the output layer. The training performance for a learning rate and a momentum set at 0.55 and 0.85 respectively is shown in Fig. 2. The fast convergence without oscillation is due to the relative large values of the learning rate and the momentum, which were chosen based on multiple tests carried out with different values of these two parameters.

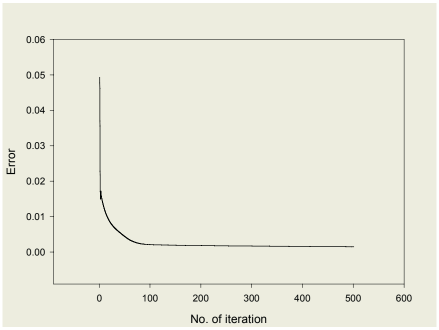


Fig. 2. Training performance of the proposed ANN

The VSTLF results for Monday July 21, 2004 and Saturday July 26, 2004 are depicted in Figs. 3 and 4 along with the actual load. The adequate performance of the proposed VSTLF is also illustrated in the mean absolute percentage error (MAPE) for 1,670 data forecasted per 6-minute ahead on a week shown in Table 1. The MAPE of the whole week, including holidays, is 0.0264 (2064%). These results indicate that the proposed ANN provides a good application to very short-term load forecasting in power system.

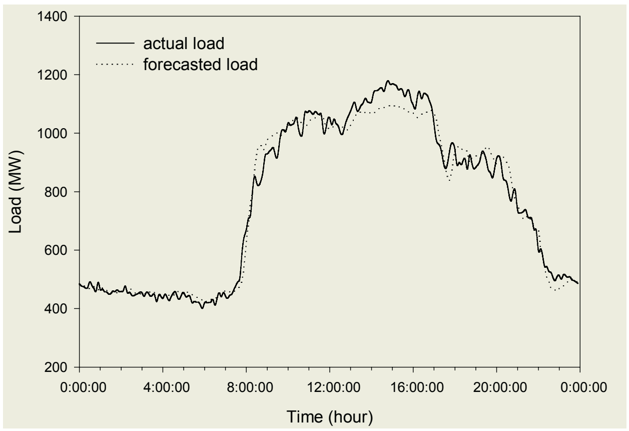


Fig. 3. Forecasting results for Monday June 21, 2004

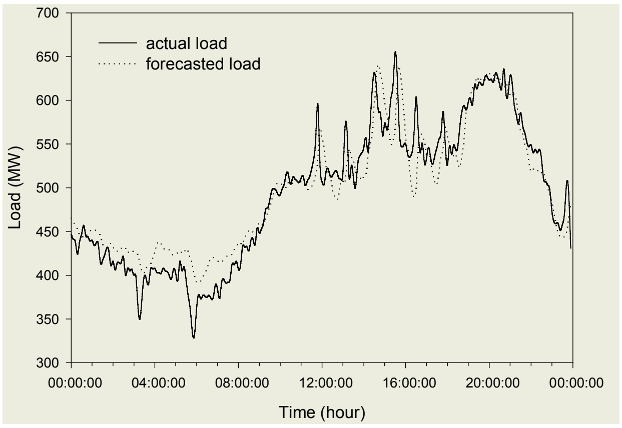


Fig. 4. Forecasting results for Saturday June 26, 2004

Table 1. Forecasting results including holidays

Weekday	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Average
MAPE%	2.98	2.80	2.57	2.40	2.43	2.37	2.98	2.64

5 Conclusions

A novel ANN-based VSTLF method that uses a three-layer feedforward neural network with the tapped delay line input and a back-propagation training method is presented. Instead of dealing with actual loads, neural networks focus on modeling load dynamics. The forecasting results indicate that this model can meet the need of improving forecast accuracy and enhancing the performance of the network. Artificial neural networks have been successfully used for instantaneous-load forecasting with time leads in a range of several to several dozen minutes. It proves the effectiveness of the application of artificial neural networks to very short-term load forecasting.

References

1. Ruzic, S., Vuckovic, A., Nikolic, N.: Weather Sensitive Method for Short Term Load Forecasting in Electric Power Utility of Serbia. IEEE Trans. on Power Systems **18** (2003) 1581-1586
2. Abdel-Aal, R.E.: Short-Term Hourly Load Forecasting Using Abductive Networks. IEEE Trans. on Power Systems , **19** (2004) 164-173
3. Kim, K., Youn, H.S., Kang, Y.C.: Short-Term Load Forecasting for Special Days in Anomalous Load Conditions Using Neural Networks and Fuzzy Inference Method. IEEE Trans. on Power Systems , **15** (2000) 559-565
4. Huang, S.-J., Shih, K.-R.: Short-Term Load Forecasting via ARMA Model Identification Including Non-Gaussian Process Considerations. IEEE Trans. on Power Systems , **18** (2003) 673-679
5. Amjady, N.: Short-Term Hourly Load Forecasting Using Time-Series Modeling with Peak Load Estimation Capability. IEEE Trans. on Power Systems , **16** (2001) 498-505

6. Ling, S.H., Leung, F.H.F., Lam, H.K., Yim-Shu Lee, Tam, P.K.S.: A novel Genetic-Algorithm-Based Neural Network for Short-Term Load Forecasting. IEEE Trans. on Industrial Electronics , **50** (2003) 793-799
7. Khotanzad, A., Enwang Zhou, Elragal, H.A: Neuro-Fuzzy Approach to Short-Term Load Forecasting in a Price-Sensitive Environment. IEEE Trans. on Power Systems ,**17** (2002) 1273-1282
8. Hippert, H.S., Pedreira, C.E., Souza, R.C.: Neural Networks for Short-Term Load Forecasting: a Review and Evaluation. IEEE Trans. on Power Systems , **16** (2001) 44-55
9. Hippert, H.S., Pedreira, C.E.: Estimating Temperature Profiles for Short-Term Load Forecasting: Neural Networks Compared to Linear Models. IEE Pro.- Generation, Transmission and Distribution ,**151** (2004) 543-547
10. Charytoniuk, W., Chen, M.-S.: Very Short-Term Load Forecasting Using Artificial Neural Networks. IEEE Trans. on Power Systems , **15** (2000) 263-268

Next Day Load Forecasting Using SVM

Xunming Li, Dengcai Gong, Linfeng Li, and Changyin Sun

College of Electrical Engineering, Hohai University, Nanjing, Jiangsu 210098, China
cysun@ieee.org

Abstract. Based on similar day method and SVM, this paper proposes a new method for next day load forecasting. The new method uses the parameters of several similar days, instead of only selecting one similar day as in similar day method. The parameters of selected similar days are used as inputs to SVM for forecasting the loads of 24 points (one hour per point) of the next day. The method behaves the advantages of both similar day method and SVM method, Corresponding software was developed and used to forecast the next day load in a practical power system and the final forecasting error is low.

1 Introduction

Load forecasting plays an important role in power system planning and operation. Basic operation functions such as unit commitment, economic dispatch, fuel scheduling and unit maintenance can be performed efficiently with an accurate forecast.

A wide variety of methods have been proposed in the last two decades owing to the importance of load forecasting, such as linear regression, exponential smoothing, stochastic process, data mining approach [1]-[4]. However, load forecasting is a difficult task as the load at a given hour is dependent not only on the load at the previous hour but also on the load at the same hour on the previous day, and on the load at the same hour on the day with the same denomination in the previous week. Generally, these methods are based on the relationship between load and factors influencing the load. However, the techniques employed for those models use a large number of complex and nonlinear relationships between the load and factors influencing the load. The traditional prediction methods are difficult to estimate these nonlinear relationships. Therefore, some new forecasting models have been recently introduced as expert systems, artificial neural networks (ANN), and fuzzy systems. Among these different techniques of load forecasting, application of ANN technology for electric load forecasting has received much attention in recently years. The main reason of ANN becoming so popular lies in its ability to learn complex and nonlinear relationships that are difficult to model with conventional techniques. However, there are some disadvantages of ANN method such as network structure is hard to determine and training algorithm has the danger of getting stuck into local minima.

Recently, a novel type of learning machine, called support vector machine (SVM), has been receiving increasing attention in areas from its original application in pattern recognition to the extended application of regression estimation [5]. This is brought about by the remarkable characteristics of SVM, such as good generalization performance, the absence of local minima and sparse representation of solution. One key characteristic of SVM is that training SVM is equivalent to solving a linearly constrained quadratic programming problem so that the solution of SVM is always unique and globally optimal, unlike ANN' training which is time-consuming and requires nonlinear optimization with the danger of getting stuck into local minima. Recently, there are also a great deal of researches concentrating on applying regression SVM to short-term electric load forecasting [6],[7],[8]. This paper proposes a new method for next day electric load forecasting, based on similar day method [9] and SVM method. Unlike the similar day method which only uses a similar day with the minimal value of difference evaluation function, here, a series of days are selected as similar day when the value of difference estimation function is less than a given value. The parameters of these selected similar days are used as samples data to train SVM. Finally, the correction should be done with the forecasted value when it violates the general rule. This method is used to forecast the next day load of a practical power system in a week. The experimental result shows that the proposed method performs well on both the forecasting accuracy and the computing speed.

2 SVM for Regression Estimation

Given a set of data points $\{(X_i, y_i)\}_i^N$, ($X_i \in \mathrm{R}^n$, $y_i \in \mathrm{R}$, N is the total number of training sample)randomly and independently generated from an unknown function, SVM approximates the function using the following form:

$$f(X) = <\omega, \varphi(X)> + b \quad . \tag{1}$$

where $\varphi(X)$ represents the high-dimensional feature spaces which is nonlinearly mapped from the input space X. The coefficients ω and b are estimated by minimizing the regularized risk function (2):

$$\text{minimize} \quad \frac{1}{2}\|\omega\|^2 + \mathrm{C}\sum_{i=1}^{N}|y_i - <\omega, \varphi(X_i)> - b|_\varepsilon \quad . \tag{2}$$

$$|y_i - <\omega, \varphi(X_i)> - b|_\varepsilon = \begin{cases} 0 & |y - <\omega, \varphi(X)> - b| < \varepsilon \\ |y - <\omega, \varphi(X)> - b| - \varepsilon & |y - <\omega, \varphi(X)> - b| \geq \varepsilon \end{cases} \quad . \tag{3}$$

The first term $\|\omega\|^2$ is called the regularized term. Minimizing $\|\omega\|^2$ will make a function as flat as possible, thus playing the role of controlling the function capacity. The second term $\sum_{i=1}^{N}|y_i - <\omega, \varphi(X_i)> - b|_\varepsilon$ is the empirical error measured by the ε-insensitive loss function(3). This loss function provides the advantage of using sparse data points to represent the designed function (1). C is referred to as the regularized constant. ε is called the tube size. They are both user-prescribed parameters and determined empirically.

To get the estimation of ω and b , (2) is transformed to the primal objective function(4) by introducing the positive slack variables $\xi_i^{(*)}$ ((*) denotes variables with and without *)

$$\text{minimize } \frac{1}{2}\|\omega\|^2 + C\sum_{i=1}^{N}(\xi_i + \xi_i^*)$$

subject to

$$\begin{aligned} & y_i - <\omega, \varphi(X_i)> - b \le \varepsilon + \xi_i \\ & <\omega, \varphi(X_i)> + b - y_i \le \varepsilon + \xi_i^* \\ & \xi_i^{(*)} \ge 0 \qquad\qquad i = 1,\dots,N \end{aligned} \quad . \tag{4}$$

Final, by introducing Lagrange multiplier and exploiting the optimality constraints, the decision function (1) has the following explicit form:

$$f(X) = \sum_{i=1}^{N}(\alpha_i - \alpha_i^*)K(X_i, X) + b \quad . \tag{5}$$

In function(5), $\alpha_i^{(*)}$ are the so-called Lagrange multipliers. They satisfy the equalities $\alpha_i \times \alpha_i^* = 0$, $\alpha_i \ge 0$, and $\alpha_i^* \ge 0$ where i =1.... N ,and they are obtained by maximizing the dual function of (4),which has the following form:

$$W(\alpha_i, \alpha_i^*) = \sum_{i=1}^{N} y_i(\alpha_i - \alpha_i^*) - \varepsilon\sum_{i=1}^{N}(\alpha_i - \alpha_i^*) - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)K(X_i, X_j) \cdot \tag{6}$$

subject to

$$\sum_{i=1}^{N}(\alpha_i - \alpha_i^*) = 0 . \qquad 0 \le \alpha_i, \alpha_i^* \le C . \quad i = 1\dots\dots N$$

$K(X_i, X_j)$ is defined as the kernel function. The value of the kernel is equal to the inner product of two vectors X_i and X_j in the feature space $\varphi(X_i)$ and $\varphi(X_j)$,that is, $K(X_i, X_j)= <\varphi(X_i), \varphi(X_j)>$. The elegance of using the kernel function that one can deal with feature spaces of arbitrary dimensionality without having to compute the map $\varphi(X)$ explicitly. Any function that satisfies Mercer's condition can be used as the kernel function. Common examples of the kernel function are the polynomial kernel $K(X_i, X_j)= (<X_i,X_j>+1)^d$ and the Gaussian kernel $K(X_i, X_j) = \exp(-(1/\sigma^2)(X_i,-X_j)^2)$, where d and σ are the kernel parameters.

From the implementation point of view, training SVM is equivalent to solving the linearly constrained quadratic programming problem (6) with the number of variables twice as that of the number of training data points. The sequential minimal optimization (SMO) algorithm extended by Scholkopf and Smola is very effective in training SVM for solving the regression estimation problem. In this paper, an improved SMO algorithm [10] is adopted to train SVM.

3 Method Based on Similar Days and SVM

In short-term load forecasting, loads of two days are generally very close when they have similar weather condition, same day class (workday or weekend) and several other similar factors. From this view, the similar day method has been developed. The advantages of the method are simple, practical and comparatively precise. However, this method has the following disadvantages. First, the selected similar day is not certainly the day which has the closest load to the forecasting day, sometimes the disparity is large. Second, due to the nonlinear characteristic of the relationship between all influencing factors and the load, it is difficult to get a good result and stability by using curve fitting or experiential method to correct the load.

Toward the first problem in similar day method, the load of selected similar day with the least value of difference estimation function is not always close to the load of the forecasting day, But load of some days with comparatively small value of difference estimation function are also possible close to the load of forecasting day. Therefore, in this paper, a series of similar days are selected rather than only the most similar day in similar day method.

Toward the second problem in similar day method, it is hard to fit the load curve using traditional method. This paper applies regression SVM to describe the complex nonlinear relationship between influencing factors and load. To a given forecasting day, it is hard to find out a large number of similar days. SVM is a good learning machine based on SRM principle, and has the capacity to solve the small samples learning problem with a good generalization performance.

The specific steps are presented in following:

First, transforming the non-numerical factors into numerical form. Taking sunlight for example, fine is set as 3, cloudy as 2, rain as 1. The maximal and minimal temperature can apply the actual value.

Second, treating disorder sample. Considering the proportion of the loads in serial time, it is sure appearing disorder data when the change of load violates the general rule. Checking every point according to this principle to pick out all disorder data and correct them.

Third, selecting similar days from four weeks before the forecasting day. When the value of difference estimation function is less than a given value γ, this day will be selected as a similar day, γ is an experiential value.

Fourth, taking parameters of these similar days as training sample to train SVM. Due to the restriction that value of difference estimation function should be less than γ, influencing factors in similar day and forecasting day are close in vector space. It is effective to make use of the generalization capacity of SVM to estimate the nonlinear relationship between load and influencing factors.

Finally, taking the influencing factor vector of forecasting day into the trained SVM, the 24 points load of next day will be forecasted. In the last, the forecasted load also should be corrected. Because load influenced by a series of uncertain factors, it is hard to get a satisfy result absolutely using historical data. Operators should correct the forecasting result from experience. Sometimes, the change of forecasted load violates the regular pattern, then this result is insecure usually, which can be instead by the mean of fore-and-aft loads.

4 Experimental Results

The example data is historical load data in a practical electric network in May and June in 2001. The data includes the data of weather condition and the data of 24 points data in every day. Using these data, 24 load points in a day are forecasted. This paper adopts the improved SMO algorithm to train SVM and RBF kernel is selected as the kernel function.

There are many error evaluation indexes to evaluate the result of daily load forecasting. In this paper, four relative error indexes are selected to evaluate the forecasting result of the proposed method. The four error indexes of forecasting result are represented in table 1.

Table 1. Relative errors of experimental result

Day	E_{MAPE}	E_{MSE}	$E_{\max}$	$E_{\min}$
Monday	3.25	3.92	0.33	2.03
Tuesday	2.85	3.30	1.36	2.08
Wednesday	2.34	2.90	3.13	0.61
Thursday	3.46	3.72	3.02	4.01
Friday	3.08	3.35	1.36	4.80
Saturday	2.99	3.43	4.90	3.29
Sunday	1.90	2.32	1.07	0.60

Table 1 indicates that the maxima of average relative error E_{MAPE} is 3.46 and the minima is 1.90. The forecasting precision of this new method is very high. The mean square root relative error E_{MSE} is bigger than average relative error E_{MAPE} which means that the relative error of forecasting value is large in some points. The forecasting models in these time points should be improved.

5 Conclusions

This paper proposed a new approach for short-term load forecasting based on similar day method and SVM. This new method uses a series of similar days as training data for SVM, which utilizes the information of all these similar days which ensures stabilization and precision of the algorithm. There does not exist the problem of correcting algorithm using SVM to fit the nonlinear relationship between load and influencing factors. Combined with similar day method, new method still keep the characteristic of easy application and effectiveness and the training time of SVM is very short. Experimental result shows that this method is an effective method of high application value for short-term load forecasting.

Acknowledgements

This work was supported by the National Natural Science Foundation of China under Grant 50595412, China Postdoctoral Science Foundation, Jiangsu Planned Projects for Postdoctoral Research Funds and a start-up grant from Hohai University of China.

References

1. Papalexopoulos, A.D., Hesterberg, T.C.: A Regression-based Approach to Short-Term Load Forecasting. IEEE Trans. Power Syst, **5** (1990) 1535-1550

2. Haida, T., Muto, S.: Regression Based Peak Load Forecasting Using a Transformation Technique. IEEE Trans. Power Syst., **9** (1994) 1788-1794
3. Rahman, S., Hazim, O.,: A Generalized Knowledge-based Short Term Load-forecasting Technique. IEEE Trans. Power Syst, **8** (1993) 508-514
4. Wu, H., Lu, C.: A Data Mining Approach for Special Modeling in Small Area Load Forecast. IEEE Trans. Power Syst, **17** (2003) 516-521
5. Vapnik, V.N., Golowich, S.E., Smola, A.J.: Support Vector Machine for Function Approximation, Regression Estimation and Signal Procession. Adv. Neural Information Procession Syst, **9** (1996) 281-287
6. Zhao, D.F., Wang, M.: A Support Vector Machine Approach for Short Term Load Forecasting. Proceedings of the CSEE, **17** (2002) 26-30
7. Yang, J.F, Cheng, H.Z.: Application of SVM to Power System Short Term Load Forecasting. Electric Power Automation Equipment, **24** (2004) 30-32
8. Li, Y.C, Fang, T.J., Zhang, G.X.: Wavelet Support Vector Machine for Short-term Load Forecasting. Journal of University of Science and Technology of China, **3** (2003) 726-732.
9. Cheng, S.: A New Approach to Load Forecasting Based on Similar Day. Proceeding of Jiangsu Electrical Engineering Association, **18** (1999) 28-32
10. Zhang, H.R., Han, Z.Z.: An Improved Sequential Minimal Optimization Learning Algorithm for Regression Support Vector Machine. Journal of Software, **14** (2003) 2006-2013.

Peak Load Forecasting Using the Self-organizing Map

Shu Fan[1], Chengxiong Mao[2], and Luonan Chen[1]

[1] Osaka Sangyo University, Daito, Osaka 574-8530, Japan
sz04013@sub.osaka-sandai.ac.jp
chen@elec.osaka-sandai.ac.jp
[2] Huazhong University of Science and Technology, Wuhan, Hubei 430074, China
cxmao@mail.hust.edu.cn

Abstract. This paper aims to study the short-term load forecasting of electricity by using an extended self-organizing map. We first adopt a traditional Kohonen self-organizing map (SOM) to learn time-series load data with weather information as parameters. Then, in order to improve the accuracy of the prediction, an extension of SOM algorithm based on error-correction learning rule is used, and the estimation of the peak load is achieved by averaging the output of all the neurons. Finally, as an implementation example, data of electricity demand from New York Independent System Operator (ISO) are used to verify the effectiveness of the learning and prediction for the proposed methods.

1 Introduction

The electrical load forecasting has already become one of the major research fields in the electric power systems [1]. Accurate load forecasts are highly required for power system operators and planners. In particular, with the rise of competitive electricity markets, short-term load forecasts become increasingly important for the trade in power markets. A wide variety of models and methods have been tried in the problem of load forecasting, which include linear regression models, exponential smoothing, stochastic process, data mining approach, autoregressive and moving averages (ARMA) models, Box-Jenkins methods, and Kalman filtering based methods [2]-[8]. However, most of those methods are mainly based on linear model, and the load series that they try to explain are usually nonlinear functions of the exogenous variables.

In the recent years, much more attention has been paid on the application of artificial neural networks (ANNs) to load forecasting [9]-[12]. Neural networks have been shown to have the ability not only to learn the time-series load curves but also to model an unspecified nonlinear relationship between load and weather variables. Although ANNs allow the estimation of possible models without the need to specify a precise functional form, the task of designing a neural network (NN) based forecasting system for a particular application is far from easy. So far, many types of ANNs architecture have already been developed in forecasting applications. The multiplayer perception, usually a feed-forward one, and recurrent networks are the mostly popular ones, which use supervised neural learning techniques. On the other hand, a few papers proposed NN architectures other than the MLP. One of them is the Self-organizing map (SOM), which uses unsupervised neural learning techniques [13]. However, in most cases, the SOM is simply adopted as a classification tool for the training samples, and the forecast is mainly performed by combining with other methods, such as MLP network and fuzzy logic [14],[15].

In this paper, a SOM network is directly applied to forecast the next day's electrical peak load, which is also a nonlinear discrete-time dynamical system. Firstly, we use the common SOM network for learning load time series curve of electricity with the standard learning algorithm. Then, to achieve more accurate forecasting results, an extension of Kohonen self-organizing map algorithm is proposed. In this extended SOM network, in addition to the weight vector, a local gradient (Jacobian) matrix $\boldsymbol{A}_i$ of the input-output pair is also learned and stored in the neuron. Since such gradient information can be explored to produce a first-order expansion in the output space around the representative output weight, an improved estimate of the output can be achieved. Accordingly, an adaptation rule of the Jacobian matrix $\boldsymbol{A}_i$ is also developed. Finally, we adopt the load curve data of New York Independent System Operator (ISO) to verify the effectiveness of the learning and forecasting for the proposed methods. The criterion to compare the performance is the mean absolute percentage error (MAPE) in this paper, which indicates the accuracy of recall. In addition, for comparative study, we also examine several network topologies with different number of neurons, where detail numerical results and their discussion for training errors and MAPE are also given in the paper.

2 ANN Model and Learning Algorithm

In this paper, a neural network is applied to identify the following nonlinear discrete-time dynamical system,

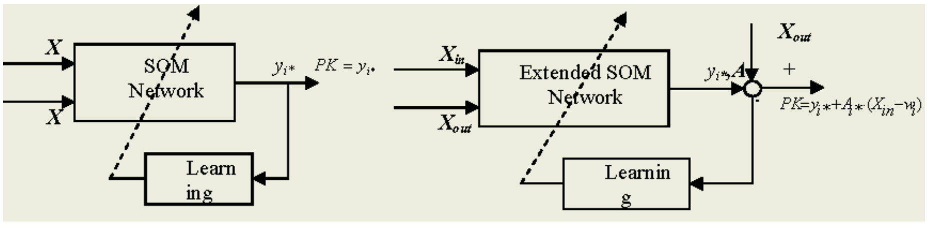
$$y(t+1) = f(y(t),...,y(t-m+1);T) \,. \tag{1}$$

where *y(t)* is a scalar representing the daily peak load of electricity at time t, and m is the orders of the dynamical system. *T* is a vector representing the control parameters of the dynamical system, such as temperature, humidity, wind-speed, and day types.

2.1 A SOM Model

The SOM is an unsupervised neural model designed to build a representation of neighborhood relationships among vectors of an unlabeled data set. The neurons in the SOM are put together in an output layer $\boldsymbol{A}$ in one, two, or even three dimensional arrays. Each neuron i has a weight vector $\boldsymbol{w}$ with the same dimension of the input vector $\boldsymbol{X}$, The network weights are trained according to a competitive-cooperative scheme in which the weight vectors of a winning neuron and its neighbors in the output array are updated after the presentation of an input vector.

The SOM network used in this paper for load forecasting is schematically shown in Fig.1. (a). For the convenience of analysis, the input vector $\boldsymbol{X}$ to the SOM has been divided into two parts. The first part, denoted $\boldsymbol{X}_{in}$, contains data of the dynamic mapping to be learned, including peak load of previous days, type of the day and weather parameters. The second part, denoted X_{out}, is the desired peak load data of this mapping. Accordingly, the weight vector $\boldsymbol{w}$ of neuron i, also comprise two parts, namely, $\boldsymbol{w}_i$ and y_i, having the same dimension with the input vector, which store the information about the inputs and outputs of the mapping being studied.



(a) SOM network (b) SOM with error correction rule

Fig. 1. The diagram of the SOM model

Therefore, to approximate the discrete-time dynamics of f by SOM, we define

$$X_{in}(t) = [y(t), \ldots, y(t-m+1), T]; X_{out} = y(t+1) . \tag{2}$$

The learning procedure consists of a series of training steps. During the training stage, the winning neuron i^* at step n is determined based on the input of the training sample, $\boldsymbol{X_{in}}$,

$$i^*(n) = \arg\min_{i \in A} \{\|X_{in} - w_i(n)\|\} . \tag{3}$$

which means that neuron i^* has the minimal distance between the weight vector $\boldsymbol{w_i}$ and the input vector $\boldsymbol{X_{in}}$ among all neurons of the output layer $\boldsymbol{A}$. The learning rules for updating the weights are stated as follows.

$$w_i(n+1) \leftarrow w_i(n) + \alpha(n) \cdot h_{ii^*}(n) \cdot (X_{in} - w_i(n)) . \tag{4}$$

$$y_i(n+1) \leftarrow y_i(n) + \alpha(n) \cdot h_{ii^*}(n) \cdot (X_{out} - y_i(n)) . \tag{5}$$

$\alpha(n)$ is the learning rate and $h_{ii^*}(n)$ is a Gaussian neighborhood function given by

$$h_{ii^*}(n) = \exp(-\frac{\|r_i(n) - r_{i^*}(n)\|}{2\sigma(n)^2}) . \tag{6}$$

where $r_i(n)$ and $r_{i^*}(n)$ are the locations of neurons i and i^* in the output array respectively. Here, $\sigma(n)$ determines the size of the neighborhood region. The parameters of $\alpha(n)$ and $\sigma(n)$ are chosen fairly large at first for rapid adaptation of the neurons, and they decay fast with time by

$$\alpha(n) = \alpha_0 (\frac{\alpha_f}{\alpha_i})^{(\frac{n}{N})} , \sigma(n) = \sigma_0 (\frac{\sigma_f}{\sigma_i})^{(\frac{n}{N})} . \tag{7}$$

where α_i and σ_i denote their initial values, while α_f and σ_f are the final ones, N is the number of total iterations. Eqn.(5) is actually similar to the cooling schedule of simulated annealing process.

After training, the SOM network can be used to forecast the next day's peak load (PK) directly from the output portion of the weigh vector y_i, as follows:

$$PK = y_{i^*}(n) . \tag{8}$$

with the winning neuron i^* found in (3).

2.2 A SOM Algorithm with Error-Correction Rule

Although the SOM models learn an input-output mapping globally, their outputs are generally not smooth due to the characteristics of SOM structures. Therefore, small training errors and accurate forecasting performance can be achieved only when the number of neurons is large enough. However, such a condition is usually not feasible in practice, in particular for load curves in wide areas and with a variety of attributions. In order to make the estimation more accurate, a local linear mapping (LLM) is used in this paper [16]. The LLM method is a linear interpolation technique which smoothes the original output.

The SOM network with error-correction rule is shown in Fig.1. (b), where an input weight $\boldsymbol{w_i}$ with its corresponding output weight y_i is stored for each neuron. Additionally, we also store the local gradient (Jacobian) matrix $\boldsymbol{A_i}$ $(=\partial f/\partial y_i)$ of this input-output pair calculated during the learning phase for each neuron. Such gradient information is used to produce a first-order expansion in the output space around the representative output weight leading to an improved estimation of the output $\hat{y}_i$, given by a linear Taylor expansion,

$$\hat{y}_i = y_i + A_i(X_{in} - w_i)\,. \tag{9}$$

Hence, the overall learning procedure of the proposed algorithm can be summarized by two steps.

Step 1: in Step-1, we only train the weight vector $\boldsymbol{w_i}$ and y_i of the neural network from the training samples, just like the learning rule (4)-(5) of the standard SOM algorithm stated above. In fact, this phase of training divides the training sample space V into a number of subspaces. In other words, each neuron i is assigned to a subregion V_i,

$$V_i = \{X_{in} \in V \mid \|X_{in} - w_i\| \le \|X_{in} - w_j\|, \qquad \forall j \in A\}\,. \tag{10}$$

Step 2: in Step 2, based on the partition of the space of training samples completed in Step 1, we design a local model for each subspace to smooth the output y_i and to infer the Jacobian matrix A_i. The adaptation rules of y_i and $\boldsymbol{A_i}$ are derived from the following cost function of mean squared error for each neuron and subspace.

$$E_i = \frac{1}{2}\sum_{j\in V_i}(X_{outj} - \hat{y}_i) = \frac{1}{2}\sum_{j\in V_i}(X_{outj} - y_i - A_i(X_{inj} - w_i))\,. \tag{11}$$

According to an error-correction rule based on the Widrow-Hoff, the steepest-descent method on the error surface is applied. Specifically, a gradient descent with respect to y_i and $\boldsymbol{w_i}$ yields

$$\Delta y_i = \sum_{j\in V_i}\varepsilon_i \cdot (X_{outj} - y_i - A_i(X_{inj} - w_i))\,. \tag{12}$$

$$\Delta A_i = \sum_{j\in V_i}\varepsilon_i' \cdot (X_{outj} - y_i - A_i(X_{inj} - w_i))\cdot(X_{inj} - w_i)\,. \tag{13}$$

where ε_i and ε_i' is the coefficient of each neural unit in the learning procedure.

Convergence: the training procedure is terminated after a fixed number of iterations or when the desired training error is achieved. The daily peak load (PK) is then forecasted by

$$PK = y_{i*} + A_{i*} \cdot (X_{in} - w_i) \,. \tag{14}$$

Since the Jacobian information is exploited in the SOM for both smoothing and extrapolation, local modeling scheme with the SOM is expected to give more accurate prediction than global modeling.

3 The Implementation

3.1 Collection of Data

The daily electrical peak load and weather data of three years in New York City and Long Island have been considered for the study. Data from July 1, 2001 to December 31, 2003 were used to estimate model parameters (*w,y,A*), whereas data from January 1, 2004 to September 31, 2004 were used to verify the trained NN or prediction. The training pattern consists of the input/output pairs, which is stated as follows, respectively.

The input of training data consists of past peak load demand, the day types, and four weather variables. In order to capture the time series style in load, we include the past peak load demand of the previous seven days. Considering the load demand on weekend and holiday is usually lower than that on workdays, we classify the data into four groups: 1) Week day; 2) Saturday; 3) Sunday; 4) holiday. There are many other variables that must be considered in the forecasting, among which the most important parameters are weather-related variables. In this paper, four weather parameters are taken to make the load forecasting more accurate, i.e., average temperature of the next day, average temperature of previous three days, the relative humidity and wind speed. Hence, there are 15 variables in the input pair of training pattern, and the overall input training pattern matrix is 15×906.

On the other hand, the output of the training pattern is the daily peak load, and the output training pattern matrix is 1×906. Once trained, the NN is then used to predict the peak load of the next day.

3.2 Training Procedure

The training procedure includes selection of geometric topology of the SOM networks, the number of the neural units, the learning rate, the neighborhood radius, and the training error, etc. The accurate learning is dependant on such selections. In this paper, we choose the topology of a two-dimension geometric arrangement of the neurons. Different number of neurons has been tried in the studies. During the learning phase of SOM algorithm, randomly chosen patterns are used to adjust the weights of neurons. The train procedure terminates after a fixed number of iterations. On the other hand, for the training phase of error-correction, all the neurons are trained in turn in the subspace for a fixed training steps or when the desired training error is achieved.

3.3 Forecasting Procedure

Once trained, the SOM network is used to forecast the peak load of the next day. As stated above, the output part of the weight of the wining neuron is actually the forecasting results. In this paper, not only the linear approximation associated with the unit i^* closest to $\boldsymbol{X}_{in}$ but also the output of a whole subset of neurons with their vector close to $\boldsymbol{X}_{in}$ are taken into account to determines the peak load. In other words, we acquire the forecasting results by averaging (14) over the neurons in a subset, weighted by the neighborhood function h_{ii^*avg}. In such a way, the neuron i^* with its $\boldsymbol{w}_{i^*}$ closest to $\boldsymbol{X}_{in}$ contributes the most, whereas the neuron with its $\boldsymbol{w}_i$ far from $\boldsymbol{X}_{in}$ has little contribution, and so forth. Then, the estimated peak load is given by.

$$PK = s^{-1}\sum_i h_{ii^*avg}(y_{i^*} + A_{i^*} \cdot (X_{in} - w_i)) . \tag{15}$$

where

$$s = \sum_i h_{ii^*avg} \text{ and } h_{ii^*avg} = \exp(\frac{\|r_i - r_{i^*}\|}{2\sigma_{avg}^2}) . \tag{16}$$

σ_{avg} determines the scope of the subset.

The quality of PLF is evaluated on the basis of MAPE

$$MAPE = \sum_{i=1}^{n} (|X_{in} - PK| * 100 / X_i) / n . \tag{17}$$

where X_{in} is the actual value; PK is the forecast value; and n is total number of value predicted.

4 Case Study and Results

Tests have been conducted on the load data of 38 months from New York ISO. For comparative studies, SOM networks with different number of neurons were tested for the two methods (i.e., a standard SOM and an extended SOM) stated above. The training errors and MAPE in the forecasting for the two methods were given in table 1 and table 2, respectively.

4.1 Results of SOM

The standard SOM was adopted to learn the training samples. We choose four topological structures of the neural networks, which are 2-D lattice of 10×10, 15×15, 20×20, and 30×30 units. The initial Jacobian matrix was chosen by assigning a random value from the interval [-1, 1] independently to each element of A_i. The initial and final values of α were set as α_i=1 and α_f=0.01. The initial values of σ were chosen as σ_i=8, 10, 15, 20 for the four different structures respectively, and the final value is σ_f=0.1. In the forecasting phase, we set σ_{avg}=1.

From table 1, clearly when the number of neurons is small, the training error and MAPE are considerably large, but the error decreases with the increase of the number of neurons. However, even when the network has 30×30 units that are almost equal to the total number of training samples (906), the performance is still not satisfactory. One of the main reasons is that the output of the SOM network is not smooth.

Table 1. Training errors and MAPE for different number of neurons

Number of neurons	10×10	15×15	20×20	30×30
Training error (%)	5.45	4.53	3.77	1.46
MAPE	5.38	4.68	4.26	3.61

4.2 Results of SOM with Error-Correction Rule

The extended SOM was adopted. We choose three topologies for the arrangement of the neurons, which are 2-D lattice of 8×8, 10×10, and 15×15 units. The initial values of σ were chosen as σ_i=5, 8, 10 for the three different arrangements respectively. The values of ε_i and ε_i' were initialized as 0.01 in every training step and then halve their values to repeat the training until the cost function E_i decreases. The values of the other parameters were set as the same as the case above.

Table 2. Training errors and MAPE for different number of neurons

Number of neurons	8×8	10×10	15×15
Training error (%)	2.89	1.32	0.54
MAPE	3.54	2.47	1.93

According to table 2, the training error decreases with the increase of the number of neurons, and it is significantly small when the number of neurons is 15×15. Clearly, the values of MAPE are in the range of 1-2%, which are considered as high accurate forecasting for the electrical load. On the other hand, although we achieved 1-2% of MAPE, the training error is still much smaller than MAPE in the numerical examples. There are several possible reasons for such phenomenon: firstly, the number of training samples is not sufficient large to cover the whole characteristics to be learned; secondly, the training may lead to over fitting of the model. In other words, there are rooms to improve the accuracy of the prediction, e.g., by enlarge the training samples or prevent the overfitting by earlier termination of the learning.

5 Conclusions

This paper proposed an extended SOM algorithm to study the problem of peak load forecasting. In order to improve the accuracy of the prediction, an adaptation rule of the Jacobian matrix $\boldsymbol{A}_i$ was developed based on the error-correction learning scheme in this paper. The estimation of the peak load is achieved by averaging the output of all the neurons. The proposed method has two learning steps. All training samples are classified and mapped to SOM in Step 1, whereas the Jacobian matrix of each classified pattern is constructed by the training samples in Step 2 to improve the learning accuracy. Numerical simulation by load curve data of New York ISO show the high accurate load forecasting, and verified the effectiveness of the learning and prediction of the proposed method.

References

1. Henrique, S.H., Carlos, E.P., Reinaldo, C.S.: Neural Networks for Short-Term Load Forecasting: A Review and Evaluation. IEEE Trans. Power Systems, **16** (2001) 44-55
2. Haida, T., Muto, S.: Regression Based Peak Load Forecasting Using a Transformation Technique. IEEE Trans. Power Systems, **9**.(1994) 1788-1794
3. Papalexopoulos, A.D., Hesterberg, T.C.: A Regression-Based Approach to Short-Term Load Forecasting. IEEE Trans. Power Systems, **5** (1990) 1535-1550
4. Rahman, S., Hazim, O.: A Generalized Knowledge-Based Short Term Load-Forecasting Technique. IEEE Trans. Power Systems, **8** (1993) 508-514
5. Huang, S.J., Shih, K.R.: Short-Term Load Forecasting via ARMA Model Identification Including Nongaussian Process Considerations. IEEE Trans. Power Systems, **18** (2003) 673-679
6. Wu, H., Lu, C.: A Data Mining Approach for Spatial Modeling in Small Area Load Forecast. IEEE Trans. Power Systems., **17** (2003) 516-521
7. Box, G.E.P., Jenkins, G.M.: Time Series Analysis – forecasting and Control. Holden-day, San Francisco (1976)
8. Infield, D.G., Hill, D.C.: Optimal Smoothing For Trend Removal In Short Term Electricity Demand Forecasting. IEEE Trans. Power Systems, **13** (1998) 1115-1120
9. Czernichow, T., Piras, A., Imhof, K., Caire, P., Jaccard, Y., Dorizzi, B., Germond, A.: Short Term Electrical Load Forecasting with Artificial Neural Networks. Engineering Intelligent Systems, **2** (1996) 85-99
10. Lalit, M.S., Mahender, K.S.: Artificial Neural Network-Based Peak Load Forecasting Using Conjugate Gradient Methods. IEEE Trans. Power Systems, **12** (2002) 907-912
11. Taylor, J.W., Roberto, B.: Neural Network Load Forecasting with Weather Ensemble Predictions. IEEE Trans. Power Systems, **17** (2002) 626-632
12. Khotanzad, A., Afkhami-Rohani, R., Maratukulam, D.: ANNSTLF – Artificial Neural Network Short-Term Load Forecaster – Generation Three. IEEE Trans. Power Systems, **13** (1998) 1413-1422
13. Guilherme, A., Barreto, Aluizio, F.R.: Identification and Control of Dynamical Systems Using Self-Organizing Map. IEEE Trans. Neural Networks, **15** (2004) 1244-1259
14. Lamedica, R., Prudenzi, A., Sforna, M., Caciotta, M., Cencelli, V.O.: A Neural Network Based Technique for Short-Term Forecasting of Anomalous Load Periods. IEEE Trans. Power Systems, **11** (1996) 1749-1756
15. Srinivasan, D., Tan, S.S., Chang, C.S., Chan, E.K.: Parallel Neural Network-Fuzzy Expert System Strategy for Short-Term Load Forecasting: System Implementation and Performance Evaluation. IEEE Trans. Power Systems, 1**4** (1999) 1100-1006
16. Walter, J.A., Schulten, K.J.: Implementation of Self-organizing Neural Networks for Visuo-motor Control of an Industrial Robot. IEEE Trans. Neural Networks, **4** (1993) 86-95

Ship Power Load Prediction Based on RST and RBF Neural Networks*

Jianmei Xiao, Tengfei Zhang, and Xihuai Wang

Department of Electrical and Automation, Shanghai Maritime University, Shanghai 200135, China
jmxiao@cen.shmtu.edu.cn

Abstract. Rough Set Theory (RST) is a powerful mathematics tool, which can deal with fuzzy and uncertain knowledge, and radial basis function (RBF) neural network has the ability to approach any nonlinear function precisely. According to the non-linear relation characteristics of ship power load, a short-term load prediction method based on RST and RBF neural network is presented in this paper. Using RST on the advantage of data analysis, the important input nodes can be selected, followed by a second stage selecting the important centers and leaning the weights of hidden nodes. The experimental results proved that this method could achieve greater predictive accuracy and generalization ability.

1 Introduction

Ship power load is a stochastic and time dependent process, which affected by waves, sea state, speed, direction, etc. Marine engines of today already possess a high degree of "intelligence" with respect to their variation in application. So, with the rapid development of modern ships, the demand for automation of ship power system to be much higher. While electric power system is on-line controlling, we should use the information of short-term load forecasting to arrange the startup and halt of generators. Thus, the capacity of generating electricity and power supplying can be distributed in reason, and the requirement of system running is satisfied. Up to date, many load prediction methods have been proposed and implemented, nevertheless, mostly used in the land load forecasting. Traditional techniques include regression models [1] and time series models [2]. The power load can also be predicted using the recently developed support vector machine (SVM) method [3]. The classical SVM algorithm is based on the quadratic programming. How to improve the real-time in data processing, to shorten the training time is still a problem urgently to be solved. Besides the above methods, artificial neural networks are used extensively in this field, and have obtained definite success [4].

However, due to the highly nonlinear characteristics in power load system, how to select the important input nodes is still a difficult problem before network training. Some input nodes, may not be significant and should be deleted because they mislead local learning and produce deterioration in the expected fit to future data. Therefore, if no preprocessing is carried out before constructing neural networks, the computation

* This work was supported by National Natural Science Foundation of China (60074004) and Science Foundation of Shanghai Education (03IK09, 04IK02).

required may be too heavy with a poor generalization capability. Rough set theory [5] put forward by Z. Pawlak in the early 1980s is suitable for dealing with imprecise, incomplete and inconsistent data. It has been successfully applied to solve many real-life problems in medicine learning, data mining, etc. In recent years, RBF neural networks have become a most promising approach for developing forecasting tools with enhanced performance capabilities. In this paper, we present a new power load prediction method based on RST and RBF network. RST is first applied to data analysis in this algorithm, including incompatible data elimination, important input nodes selection, followed by training the weights of hidden nodes in the second stage. The effectiveness of the new approach is illustrated by simulation results.

2 RST Data Analysis

Rough set theory has proved its usefulness as a new mathematical tool for data analysis. The main advantage of rough set theory is that it does not need any preliminary or additional information about data.

A To Establish the Information Table

Knowledge representation in rough set theory is done via information system. An information system can be characterized as a decision table, where the information is stored in a table. Each row in the table represents an individual record. Each column represents some an attribute of the records or a field.

B To Use RST to Get Suitable Input Nodes

It is well known that an information system or a decision system may usually have irrelevant or superfluous knowledge (attributes), which is inconvenient for us to get concise and meaningful decision. Since then, the reduction of attributes is demanded greatly. In rough set theory, the reduction is defined as a minimal set of attributes that enables the same classification of elements of the universe as the whole set of attributes.

Thus, we can use the RST to get the key attributes. Here, a reduction strategy based on the positive region is adopted [6]. After this operation, we get n factors as the input of the networks.

C To Eliminate Inconsistent Knowledge of the Training Data

Each row in decision table can regard as a decision rule. If the condition attributes of any two rules are equal whereas the decision attributes are not, we say that the two rules are inconsistent. When this happens both rules should be removed from the training data because they will mislead the local leaning.

If two or more rules represent the same class, all but one of the rules should be eliminated. This reduces computational time.

3 RBF Networks Configuring and Training for Load Prediction

RBF neural network is a three-architecture with no feedback, which has only one hidden layer [7]. The input layer units and the output units are determined by the practical problems and only the number of hidden layer units can be variable.

The performance of RBF neural network is decided by the hidden layer, which consists of H hidden neurons (radial basis units) with radial activation functions, and each neuron only responds to an input that is close to its center. Different basis functions $\varphi(\cdot)$ can be adopted; a typical choice for this function is the Gaussian function, which is given by the following equation:

$$\varphi(x) = \exp(-\|x - c_i\|^2 / 2\sigma^2) . \tag{1}$$

The detailed description of the network structure can be found in many inferences and can be considered as a multidimensional interpolation technique implementing general mappings $f : R^N \to R$ according to:

$$f(x) = w_0 + \sum_{i=1}^{H} w_i \varphi(\|x - c_i\|) . \tag{2}$$

Where $x = [x_1, x_2, \cdots, x_n]^T$ is the input vector and $c_i = [c_{i1}, c_{i2}, \cdots, c_{in}]^T$ are the center vectors; w_i are the weights of the output layer, $\varphi(\cdot)$ is the basis function, and $\|*\|$ denotes the Euclidean norm.

Using the advantage of RST in data processing, we can extract the connotative rules from the training data, and each rule is compact formally. Such every rule represents a certain class of the batch of data. Because the decision rules set is minimum and has covered all relations in initial data, the condition of rules is exactly the ideal center vector space in RBF that we should seeking for based on clustering method, and each vector is a representative point in initial data. So, it is reasonable to regard each rule's condition as a hidden neuron center vector in RBF neural networks.

After having determined the hidden units, in the last step of the training process the weight matrix w is evaluated. The objective is to find the weight w which minimize the squared norm of the residuals:

$$E = \|Y - wz\|^2 . \tag{3}$$

where Y is the $M \times K$ matrix of training targets, $z=(z_1, z_2, \ldots, z_k)$ is a $H \times K$ matrix, and w is the $M \times H$ weight matrix.

The weights w are initialized randomly and are updated by gradient descent method:

$$w(w+1) = w(n) - \eta \frac{\partial E^n}{\partial w} . \tag{4}$$

Where η is the learning rate, which is small enough and decreases gradually.

4 Simulation

It is important that proper selection of input variables to meet the load characteristics of ship power system. Initially, the following notation is used for creating the input variable candidate set:

$$\begin{gathered} X(t) = [u(t), m(t), y(t-1), y(t-2), \cdots, y(t-n_y)]^T . \\ y(t) = f[X(t)] . \end{gathered} \tag{5}$$

where $u(t)$ represent the state of the ship, such as startup or halt, loading or unloading, well-balanced running or in a emergency, and so on; $m(t)$ is the average load value before the forecasting moment; and $y(t\text{-}1)$, $y(t\text{-}2)$,…, $y(t\text{-}n_y)$ are the actual load values when the time t-1,t-2, $t\text{-}n_y$. While $y(t)$ is the output of prediction at the moment t.

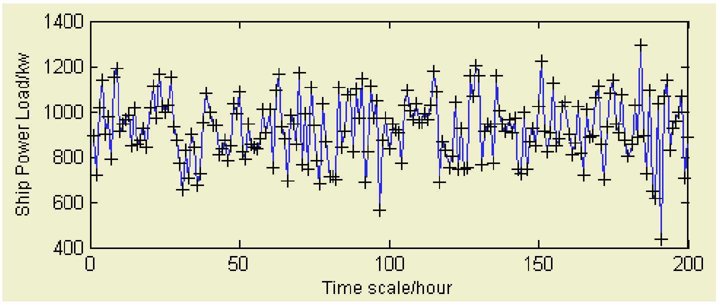
RST is used first to determine the significant lag n_y and then to select the most appropriate regressors which make up the RBF network.

A ship power load prediction is mainly researched in this section to test the performance of the proposed method. The initial data come from a ship power load, which can be available at [8]. Gathering 300 samples from the ship power load history dataset, namely the time series t=300. And the first 200 samples used as the training data, the remaining 100 samples as the prediction data.

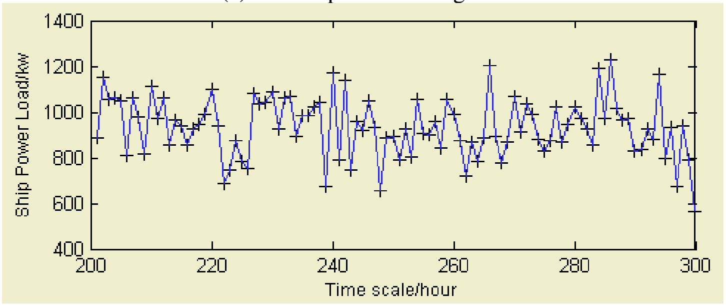
At the beginning, the maximum lag was set to be 10. Thus, the establishing information table consists of 12 condition attributes. After analysis by RST, the 7 input nodes [$u(t)$, $m(t)$, $y(t\text{-}1)$, $y(t\text{-}2)$, $y(t\text{-}3)$, $y(t\text{-}4)$, $y(t\text{-}5)$] were selected as the best subset from the 12 input nodes. The centers of RBF network were then formed from the extracted 128 rules to configure the hidden units of networks. In order to verify this method being superior, conventional RBF predictors are presented suing all the training data without pretreatment. The training and prediction results of ship power load are shown in Fig.1 ((a), (b)), Fig.2 respectively. Fig.3 show the relative errors between predictions with actual power load values by both two methods. Here, the relative error defined as:

$$RE = | y_i - \hat{y}_i | / y_i \ . \tag{6}$$

where y_i is actual load value, and $\hat{y}_i$ is the predictive load value.



(a) The output of training results



(b) The output of prediction

Fig. 1. The simulation results with this method proposed in this paper.

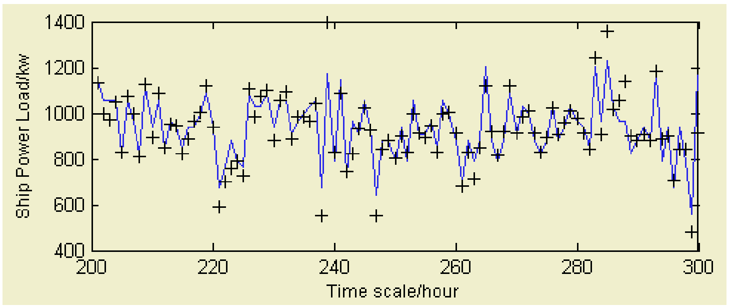


Fig. 2. The prediction results by conventional RBF network.

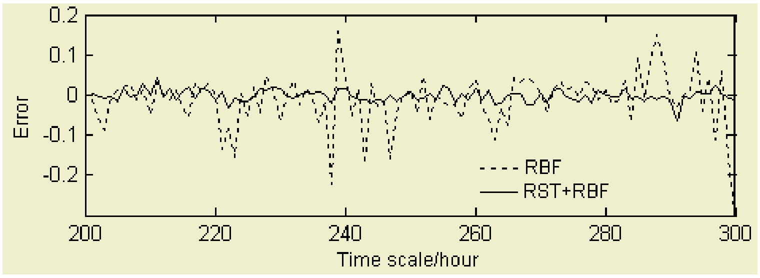


Fig. 3. The errors between prediction with actual power load values.

The root mean square error (*RMSE*) and the mean absolute error (*MAE*) are selected as the error measure to evaluate the predictive performance, and defined as follows:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2} \quad , \quad MAE = \frac{1}{N}\sum_{i=1}^{n}\left|y_i - \hat{y}_i\right| . \tag{7}$$

The test results with both methods are shown in Table.1.

From Table.1, we can know that proper selection of input variables and preprocessing the training data could predict the ship power load with sufficient accuracy.

Table 1. The test results error of both methods.

Network model		RST+RBF	RBF
Training	RMSE	0.0094	0.0320
	MAE	0.0080	0.0249
Prediction	RMSE	0.0198	0.0710
	MAE	0.0136	0.0525

5 Conclusions

Being strongly influenced by various and complicated factors, variations of ship power load have much non-linear relation, which cannot be easily predicting accurately. A new load prediction method based on RST and RBF neural network is presented in this paper. This method succeeds in combining the RBF neural network study and analysis ability with RST. And it is tested in the experimental prediction successfully.

References

1. Papalexopopoulos, A.D., Hesterberg, T.C.: A Regression Based Approach to Short-term Load Forecasting. IEEE Trans. Power Systems, **5** (1990) 1535-1547
2. Park, J.H., Lee, K.Y.: Composite Modeling for Adaptive Short-term Load Forecasting. IEEE Trans. Power Systems, **6** (1991) 450-457
3. Wang, X., Zhu, S.: Ship Power Load Forecasting Using Support Vector Machine. Proceedings of CSEE, **24** (2004) 36-39
4. Fuhaid, A.S., Sayed, M.A., Mahmoud, M.S.: Cascaded Artificial Neural Networks for Short-term Load Forecasting. IEEE Trans. Power Systems, **12** (1997) 1524-1529
5. Pawlak, Z.: Rough Sets. International Journal of Computer and Information Science, **11** (1982) 341-356
6. Xiao, J., Zhang, T.: New Rough Set Approach to Knowledge Reduction in Decision Table. Proceedings of the Third International Conference on Machine Learning and Cybernetics (2004) 2208-2211
7. Xiao, J., Wang, X.: Highway Traffic Flow Model Using FCM-RBF Neural Network. In: F.L.Yin, J. Wang, C. Guo (eds.): Advances in Neural Networks. Lecture Notes in Computer Science, 3174. Springer-Verlag, Berlin Heidelberg New York (2004) 956–961
8. Wu, X., Song, Y., Wang, Y.: Estimation Model for Loads of Ship Power System based on Fuzzy SOFM Network. Ship Building of China, **44** (2003) 65-70

Contingency Screening of Power System Based on Rough Sets and Fuzzy ARTMAP

Youping Fan[1], Yunping Chen[1], Wansheng Sun[1], Dong Liu[1], and Yi Chai[2]

[1] Faculty of Electrical Engineering, Wuhan University, Wuhan, Hubei 430072, China
Fyoupingnxinrong@yahoo.com.cn
[2] Automation College, Chongqing University, Chongqing 400044, China
chaiyi@cqu.edu.cn

Abstract. The paper adopts rough sets theory and fuzzy ARTMAP method to explore the online adaptive contingency classification in power system transient stability control. On the basis of contingency vector space model, the rough sets theory is applied to generalize the information system comprised by contingency samples set, and compute the best reducing properties set. So dimension of contingency feature space is reduced greatly, and disturbance in contingency classification is decreased too, which improves the efficiency of classification. In addition, using the advantage of adaptive classification and incremental learning of Fuzzy ARTMAP neural network, the online adaptive classification of contingency is achieved.

1 Introduction

The realization of fast on-line contingency screening is necessary for on-line transient control and security control of the power system. In this way the contingency set with stabilizing failure or failure possibilities for further strategy analysis can be determined. As an information process tool with excellent performance, ANN is applicable in contingency screening for the purpose of on-line transient stability analysis [1]. ANN-based contingency screening essentially adopts ANN to make transient stability assessment (TSA) for the specialized system under expected contingency set [2], [3], [4], [5]. For TSA, output space can be simplified as stability and stability failure states. Input space varies depending on different examined object. Practically there are two ways to express input space. One is to express input feature variables as operating state and fault itself, the other is expressed as state variables or its combination reflecting system stability at a certain moment or within certain time slice after the occurrence of fault. Though not concerning about complicated transient stable physical course itself, the former method needs to obtain a large quantity of samples relying on different power nets, system states, faults and maintained states etc. Moreover, the practical manipulation gets complicated and the wanted input dimensions get bigger with the increase of power net scale. So it usually works on power net with simple scale or the local part of complicated power net. The latter one works deeply into the complicated physical course with transient stability and tries to use state variables or its combinations to completely express input pattern including system operating state, fault state etc. However, it is also unfit for the complicated large-scale power net for the increase of generators will result in the increase of input variable dimensions and then computation difficulty arises.

Aimed at the above problems, the paper proposes an online adaptive classification method based on rough set and Fuzzy ARTMAP for fast TSA in large power system. The organization is as follows. In Section 2, vector representation for contingency is described and rough sets theory is employed to restrict its attribute set. In Section 3, Fuzzy ARTMAP is applied to contingency classification and its algorithm is given. In Section 4, a simulation test result is given. Finally conclusions are drawn in Section 5.

2 Contingency Feature Space and Its Attribute Restrictions

2.1 Vector Space Model of Contingency

Rough set theory mainly investigates data architecture composed of object set and attribute set [6], which is generally called vector space model (VSM). Its form shows in Table 1. The object of the decision list represents features, symptoms and signs etc. Attributive is divided into condition attributive and decision attribute, where $U=\{X_1,X_2,\ldots,X_n\}$ represents object set, $C=\{F_1,F_2,\ldots,F_m\}$represents condition attributive set, D represents decision attributive; f_{ij} represents jth state attributive value of ith object. d_i represents decision attribute value of ith object.

Table 1. Vector Space Model

U	F_1	F_2	…	F_n	D
X_1	f_{11}	f_{12}	…	f_{1m}	d_1
X_2	f_{21}	f_{22}	…	f_{2m}	d_2
⋮	⋮	⋮		⋮	⋮
X_n	f_{n1}	f_{n2}	…	f_{nm}	d_n

In transient security analysis, decision Table 1 shows the following explanation. $X_i(i=1,2,\ldots,n)$ Indicates n states of the power system's multiple operating ways. $F_i(i=1,2,\ldots,m)$ indicates m features and signs of operating states. $d_i(i=1,2,\ldots,n)$ indicates assessment of ith operating state. Transient security analysis needs to settle how to select least feature set to reduce attributive dimension, release computation burden and reduce influence made by uncertain factors by guaranteeing the coincidental assessment of operating states. Attribute reduction method in rough sets theory can settle the problem well.

2.2 Contingency Feature Space Decreasing Dimensions

Viewed from Rough set theory[7], trait entry $T_1,T_2,\ldots, T_n$ constitutes attributive set $\{T_1,T_2,\ldots, T_n\}$. Information system is defined as $K = <U, Q, V, f>$, where $U= \{d_1, d_2,\ldots, d_L\}$ indicates universe composed of the entire training samples contingency, $Q = \{C, D\}$ indicates the attributive set, where $C = \{T_1,T_2,\ldots, T_n\}$ is called condition attributive, $D = \{dp\}$ indicates result attributive or decision attributive and the value selection for dp derives from contingency classification. $V = \bigcup_{q\in Q} V_q$ indicates attributive value set, f indicates information function. For $\forall d \in U$, $q \in Q$, there is $f(d,q) \in V_q$. The value range for every attributive is continuous value [0, 1]. Data universalization is made first for the convenience of roughness set process, and then interval [0, 1]is made discretization into M_i pieces.

Now confirm universalizing rate $p\,(0 < p \leq 1)$ and range value for attributive value set cardinal $m_i < M$, and then assume conception tree set H where element H_i responses with conception hierarchical of the corresponding attributive $T_{i,}$ sample contingency set U and condition attributive set C. The descriptive algorithm is given as follows:

Step 1 Let $d_m = p\times card\ (U)$; U' = U; C' = C

Step 2 Computing cardinals for every attributive value set $d_i = card\ (V_{Ti})$;

Step 3 If $card\ (U') > d_m$ and $d_i > m_i$, then go to the next step. Otherwise stop and output universalized information system.

Step 4 Select attribute $T_i \in C'$ which makes d_i /m_i biggest. If T_i is universal, it is raised to the upper class conception In $H_{i\,,}$ whose interval numbering is used to replace the value in V_{Ti}. Otherwise, $C' = C' - \{T_i\}$.

Step 5 Delete all repeated element group from U' and then turn to Step 2.

The above is the whole course of data universalization. Range of every attribute is changed as $V_{Ti} = \{1, 2, \ldots, M_i\}$. Information system K is transformed into universalized one: $K' = <U, Q, V', f>$.

For two attribute sets $R\in C$ and D, the positive range of R in $U/IND(D)$ or R positive range of D , defined as: $POS_R(D) = \bigcup_{X\in U/IND(D)} R\_(X)$. Where $U/IND(D)$ indicates the universe U's division on indiscernible relation IND (D) of attribute set D, $R\_(X)$ indicates R lower approximation of set X, expressed as $R\_(X) = \bigcup\{Y \in U/IND(R) : Y \subseteq X\}$.

Lower approximation $R\_(X)$ is the object set included in set X determined by $R\_(X)$ defined knowledge. Due to the only decision attribute dp in D, $R\_(X) = \bigcup\{Y \in U/IND(R) : Y \subseteq X\}$ can be transformed as: $POS_R(D) = \bigcup_{X\in U/dp} R\_(X)$. Where U/dp is all the equivalence family of dp or all the classification of sample contingency set U.

Attribute set D's reliance degree on attribute set R is defined as : $\gamma_R(D) = card(POS_R(D))/card(U)$. In it, $card(\bullet)$ indicates set cardinals or the contingency training samples set and $card(U)$ is a constant.

Importance of attribute reflects contribution of the different attributes' reliance degree on condition attribute and decision attribute in condition attribute. Assuming attribute set $R \subseteq C$, condition attribute $a\in R$, importance degree of attribute a in R for $U / IND(D)$ is defined as: $G_{R,D}(a) = \gamma_R(D) - \gamma_{R-\{a\}}(D)$.

Core is the basis of computation attribute statute set. In the rough set theory, core reflects the intersection set of all restriction attribute sets of attribute set C relating to attribute set D. For attribute $r\in C$, attribute r is called unommitted parts of D, if and only if $POS_C(D) = POS_{C-\{r\}}(D)$. All unommitted attribute set in C is D core of C, written as: $CORE_D(C)$. Core can be computed through discerning matrix. Define discerning matrix *M* (*C*) relating to attribute set *C* as $M(C)=(m_{i,j})_{n\times n}$.

$$(m_{i,j})=\begin{cases}\Phi & d_i,d_j\in D \quad \text{sameequivalentkind}\\ \{T\in C: f(d_i,T)\neq f(d_j,T)\}, d_i,d_j\in D & \text{differentequivalentkind}\end{cases} \tag{1}$$

As there are only one-attribute element *ds* in *D*expressing classification numbering of training sample contingency, the above equation can be simplified as

$$(m_{i,j})=\begin{cases}\Phi & f(d_i,dp)=f(d_j,dp)\\ \{T\in C: f(d_i,T)\neq f(d_j,T)\} & f(d_i,S)\neq f(d_j,dp)\end{cases} \tag{2}$$

Computing according to discerning matrix

$$CORE_D(C)=\{T\in C: m_{i,j}=\{T\},1\leq j<i\leq n\} \tag{3}$$

That is, elements in core are the sets constituted by elements in the unit only including single element in discerning matrix.

So far, the optimal restriction set in condition attribute set *C* can be computed. Computing steps are as follows:

Step 1 Build discerning matrix *M* (*C*) about condition attribute set *C* in the information system *K*. Computing relating core $CORE_D(C)$ of attribute set *C*.

Step 2 Let *R* = *C, E* = $CORE_D(C)$; $R'=R-E$.

Step 3 For every attribute $T_i\in R'$, computing the importance $G_{R',D}(T_i)$ of attribute according to Equation (7) and then put them in order according to the size. Computing attribute set *D*'s reliance degree $\gamma_E(D), \gamma_R(D)$ on *E* and *R*.

Step 4 If $\gamma_E(D)=\gamma_R(D)$ or $R'=\Phi$, then turn to Step 6.

Step 5 select corresponding attributes T_j of $G_{R',D}(T_j)_{max}$. Be careful if the importance of many attributes maybe equals to the largest importance, then select the attribute with the least attribute value, that is, *card* (V_j) is the smallest. If $E=E\cup\{T_j\}$, $R'=R'-\{T_j\}$, compute *D*'s reliance degree on *E* again: $\gamma_E(D)$ and turn to Step 4.

Step 6 Let *n'* = *card* (*E*), $\Gamma=\gamma_E(D)$, the following is a feedback course, expressed as follows in the form of false code. *For* (*int i* = *0; i* < *n'; i* ++) *if* ($T_i\notin CORE_D(C)$) $E=E-\{T_i\}$, Delete the attribution not belonging to the related core from *E* one by one and compute $\gamma_E(D)$; If ($\gamma_E(D)\neq\Gamma$), $E=E\cup\{T_j\}$ and reliance degree has changed, then recover the attribute.

Step 7 Put out the optimal attribute statute set *E, card* (*E*) = *m*, $m\leq n$.

So far, the optimal attribute statute set *E* of the contingency is available.

3 Contingency Adaptive Classification Based on Neural Network

Fuzzy ARTMAP [8] can process off-line and instructive learning. Fuzzy ARTMAP is composed of two Fuzzy ART modules connected together, which are called ART$_a$ and

ART$_b$ separately. ART$_a$ receives m dimensions of input pattern vector a. On the stage of off-line supervised training, input of ART$_b$ is expectation output b of input pattern a, representing its correct classification. While adopting the net in file classification, instructing input of ART$_b$ is the exact classification of sample files, representing as vector: $b=(b_1,b_2,\ldots,b_n)$. $b_i=1(i=k)$; Else $b_i=0(i\neq k)$. k is classification belonging to the sample file a, n is the total classification number of the sample file. Therefore, b is directly loaded on mapping field. Simplified Fuzzy ARTMAP is given in Fig 1.

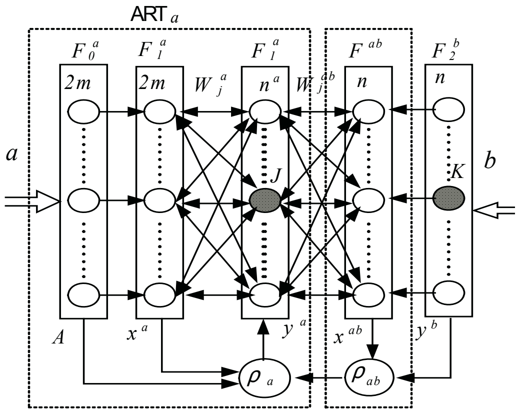


Fig. 1. Simplified Fuzzy ARTMAP Architecture

Supervised training algorithm of Fuzzy ARTMAP is as follows:

Step 1 Initializing net. Suppose x^a,y^a,y^b,x^{ab} to be zero vector, weight of W_j^{ab}, W_j^a are both supposed to be 1 and warning parameter ρ_a is supposed to be basic value $0<\rho_a<1$.

Step 2 Inputting sample (a^i,b^i), a^i is transformed as A by means of F_0^a. output y^a,y^b of F_2^a, F_2^b is as follows: $y_j^a=\left|A\wedge W_j^a\right|/\left(\alpha+\left|W_j^a\right|\right)$, $y_j^b=b_j$.

Where α is selecting parameter, generally chosen as smaller positive, $\wedge$ is fuzzy and, defined as $(p\wedge q)_i\equiv min(p_i,q_i)$ which reflects matching degree between input vector A and long period of remembering weight. By competeting, the winner unit of F_2^a produces output. Assume the winner unit to be J, then activate mapping field. F^{ab} level output of mapping field is given as:

$$x_{ab}=\begin{cases} y^b\wedge W_J^{ab} & \text{Jth node of F and F are both activated} \\ W_J^{ab} & \text{Jth node is activated while F isn t} \\ y^b & \text{F isnt activated while F is} \\ 0 & \text{F and F are both not activated.} \end{cases} \tag{4}$$

If y^b and W_J^{ab} is not matching, ART$_a$ starts to match tracing course to search for a better kind node.

Step 3 Matching tracing. If $\left|x_{ab}\right| < \rho_{ab}\left|y^b\right|$, then the predicting classification of ART$_a$ is out of match with actual classification. Now the warning parameter ρ_a of ART$_a$ must be modified as: $\rho_a = \left|A \wedge W_J^a\right| / \left|A\right| + \delta$. Where δ is a number slightly larger than zero. Output x^a of Level F_1^a must satisfy the equation: $\left|x^a\right| = \left|A \wedge W_J^a\right| < \rho_a\left|A\right|$.

Then the original resonance condition of ART$_a$ is destroyed. ART$_a$ starts to choose other node K of F_2^a, which must satisfy the following equation: $\left|x^a\right| \geq \rho_a\left|A\right|$ and $\left|x^{ab}\right| \geq \rho_{ab}\left|y^b\right| = \rho_{ab}$.

If satisfied, then go on with slow learning. Update its connecting weight vector as: $W_{K(new)}^a = \beta\left(A \wedge W_{K(old)}^a\right) + (1-\beta)W_{K(old)}^a$ and $W_{K(new)}^{ab} = \beta\left(y^b \wedge W_{K(old)}^{ab}\right) + (1-\beta)W_{K(old)}^{ab}$. In it, $0 \leq \beta \leq 1$, which is called learning speed rate.

If all the F_2^a nodes are not satisfied, a new F_2^a node N is produced. Then initialize its connecting weight (fast learning) as: $W_{N(init)}^a = A$. In the mapping field, supposing the new node corresponding with *Sth* kind node of F_2^b, the connecting weight vector between the node and mapping field is $W_{NS}^{ab} = 1$, else $W_{Nk}^{ab} = 0$ ($k \neq S$).

When nets work on a state of on-line classification, its working way is the same with common Fuzzy ART without providing instructing input for the net any more. If all the pattern classification produced by new files under supervised training doesn't match, Fuzzy ARTMAP produces new kind node from level F_2^a to store the new pattern.

4 Contingency Feature Space and Its Attribute Restrictions

The system is realized by adopting VC++6.0 based on Windows2000. Next it will be confirmed by analyzing a certain computing case in a simulating experiment.

Training samples: Due to the working state of Central China Power Net mainly depending on the four factors: nets topology structure, output of Gezhou Dam Power Station, power reception of East Hubei Province and exchanging power on connecting lines, the working state can be divided into several classes: a certain typical working state of Central China written as working state A, working state B when the power reception of East Hubei Province falls down with the decrease of output power of Gezhou Dam power station, working state C when output power of Gezhou Dam remains unchanged, and the power reception of East Hubei Province falls down with the decrease of output power of Geheyan power station, and working state D when output power of Gezhou Dam power station and the power reception of East Hubei Province remain unchanged, while power production and power consummation of Henan Province power net are inclined to less. Under the above 4 different working states, 150 samples totally 600 samples of system's stable cases for all 500kv output power lines are fetched on different line positions where three-phase short-out faults happen. Off-line computation shows that the working sequence from big to small is ADCB referring to the endangering degree of the system's transient stability.

Measuring samples: Under working state A and a certain working state with less power production and consummation in Henan Power Net written as E, the stable cases of the system with three-phase short-out faults on different positions of all 500kv output power lines are called measuring samples, whose positions don't coincide with training samples. 128 samples are fetched under the former working state, which is the same with the one used in training samples. 144 samples are fetched under the working state E, which is a new working way between A and B, employed to judge ANN's ability to recognize new working state.

Domain value selection: The net's stable index distributing figure responsing for the training samples is combined with the stable pattern of training samples to determine the field value of unstable field, fuzzy field, stable field separately and then depend on them to determine the stable pattern of measuring samples. **Fig. 2** shows the case of determining net field value. It's seen that unstable field, fuzzy field and stable field can be defined as $1 \le y_i < 1.25$; $1.25 \le y_i < 1.7$, $1.7 \le y_i \le 2.0$.

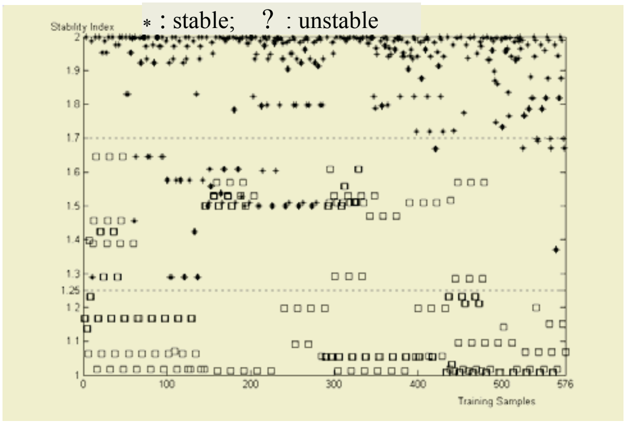


Fig. 2. Setting classification threshold under different operation conditions

Measuring result: Obviously, the net makes good effect on predicted contingency set under the known working state and possesses a certain deductive ability for predicted contingency set under the unknown small working state, which shows that the function of the net depends on the covering range of the training samples on a large degree.

5 Conclusions

The paper adopts rough set based data restriction to effectively settle generation of the system of coordinates for contingency feature space and the problem of high dimensions, theoretically giving the method for reducing redundancy attribute, which is helpful for increasing accuracy for contingency judgment and reducing the computing scale. It takes obvious advantages on the ability of adaptive clustering. Hence, the research on classification algorithm based on adaptive nerve net shows a good prospect by being applied in contingency automatically on-line screening and makes great significance for the information management and knowledge discovery of power system transient stability analysis.

Acknowledgements

This work is Supported by the National Natural Science Foundation of China (50477018) and the Post-doctoral Science Foundation of China (2004036135).

References

1. Edwards, A. R., Chan, K. W., Dunn, R.: Transient Stability Screening Using Artificial Neural Networks within A Dynamic Security Assessment System. IEE Proc. Gener. Transm. Distrib., **143** (1996) 159-166
2. Aggarwal, R., Song, Y.: Artificial Neural Networks in Power Systems. Part 1: General Introduction to Neural Computing. Power Engineering Journal, **140** (1997) 129-134
3. Aggarwal, R., Song, Y.: Artificial Neural Networks in Power Systems. Part 2: Types of Artificial Neural Networks. Power Engineering Journal, **141** (1998) 41-47
4. Aggarwal, R., Song Y.: Artificial Neural Networks in Power Systems. Part 3: Examples of applications in power systems. Power Engineering Journal, **142** (1998) 279-287
5. Tso, S.K., Gu, X.P., Zeng, Q.Y., Lo, K.L.: An ANN-based Multilevel Classification Approach Using Decomposed Input Space for Transient Stability Assessment. Electric Power System Research, **46** (1998) 259-266
6. Pawlak, Z.: Rough Sets, Theoretical Aspects of Reasoning about Data. Dordrecht: Kluwer Academic Publishers, (1991)
7. Stefanowski, J., Vanderpooten, D.: A General Tow-Stage Approach to Inducing Rules from Examples. Rough Sets, Fuzzy Sets and Knowledge Discovery. Berlin: Springer-Verlag, (1994) 317-325
8. Carpenter, G.A., Grossberg, S., Markuzon, N., Reynolds, J.H., Rosen, D.B.: Fuzzy ARTMAP: A Neural Network Architecture for Incremental Supervised Learning of Analog Multidimensional Maps. IEEE Trans. Neural Networks, **3** (1992) 259-266

Intelligent Neuro-fuzzy Based Predictive Control of a Continuous Stirred Tank Reactor

Mahdi Jalili-Kharaajoo and Farzad Habibipour Roudsari

Young Researchers Club, Islamic Aazd University, Tehran, Iran
mahdijalili@ece.ut.ac.ir

Abstract. In this paper, a predictive control strategy based on neuro-fuzzy (NF) model of the plant is applied to Continuous Stirred Tank Reactor (CSTR). An optimizer algorithm based on evolutionary programming technique (EP) uses the identifier-predicted outputs and determines input sequence in a time window. Using the proposed neuro-fuzzy predictive controller, the performance of Ph tracking problem in a CSTR process is investigated.

1 Introduction

Model based predictive control (MBPC) [1],[2] is now widely used in industry due to its ability to handle difficult control problems which involve multivariable process interactions, constraints in the system variables, time delays, etc. The classical MBPC algorithms use linear models of the process to predict the output of the process over a certain horizon, and to evaluate a future sequence of control signals in order to minimize a certain cost function that takes account of the future output prediction errors over a reference trajectory, as well as control efforts. Although industrial processes especially continuous and batch processes in chemical and petrochemical plants usually contain complex nonlinearities, most of the MPC algorithms are based on a linear model of the process and such predictive control algorithms may not give rise to satisfactory control performance [3],[4]. In recent years, the use of neuro-fuzzy networks for nonlinear system identification has proved to be extremely successful [5]-[9]. The aim of this paper is to develop a nonlinear control technique to provide high-quality control in the presence of nonlinearities, as well as a better understanding of the design process when using these emerging technologies, i.e., neuro-fuzzy control algorithm. In this paper, we will use an EP algorithm [10],[11] to minimize the cost function and obtain the control input. The paper analyzes a neuro-fuzzy based nonlinear predictive controller for a Continuous Stirred Tank Reactor (CSTR), which is a highly nonlinear process [12].

2 Neuro-fuzzy Identification of the Plant

The structure of intelligent adaptive predictive control is shown in Fig. 1. Prediction system is formed by neuro-fuzzy identifier (NFI) to generate the anticipated plant output for a future time window, $N_1 \le t \le N_2$. The fuzzy rules and membership functions of this identifier can be trained off-line by the actual measured data of CSTR process. The future control variable for this prediction stage is determined in an optimization algorithm for the time interval of $N_1 \le t \le N_u$, such that $N_u \le N_2$, minimizing the following cost function:

$$J = \sum_{k=N_1}^{N_2} \left\| \hat{y}(t+k) - r(t+k) \right\|_R^2 + \sum_{k=N_1}^{N_u} \left\| \Delta u(t+k) \right\| + \left\| y(t) - r(t) \right\|_R^2 \tag{1}$$

where $\hat{y}(t+k)$ is predicted plant output vector which is determined by NFI for time horizon of $N_1 \le k \le N_2$, $r(t+k)$ is desired set-point vector, $\Delta u(t+k)$ is predicted input variation vector in time range of $N_1 \le k \le N_u$, $y(t)$ and $r(t)$ are present plant output and set-point vectors, respectively. The optimization block finds the sequence of inputs to minimize cost function in (1) for future time, but only the first value of this sequence is applied to the plant. We use the neuro-fuzzy network proposed in [9], a four-layer network, for the objective of predictive control. The first layer of identifier network represents the input stage that provides the crisp inputs for the fuzzification step. The second layer performs fuzzification. The weights and biases respectively represent the widths and means of input membership functions. Using exponential activation function, the outputs of the second layer neurons are the fuzzified system inputs. Weighting matrix of the third layer input represents antecedent parts of rules, and is called the *Premise Matrix*. Each row of the premise matrix presents a fuzzy rule such as:

$$R_{\mathrm{Pr}emise} : IF\ x_1\ is\ T_{x_1}^1\ AND ...\ AND\ x_m\ is\ T_{x_m}^1\ THEN...$$

where $T_{x_i}^j$ is the *j*th linguistic term of the *i*th input.

The neuron output is determined by the *min* composition to provide the firing strength of rules. The fourth layer consists of separate sections for every system output. Each section represents consequent parts of rules for an output, such as:

$$R_{Consequent} : ...\ THEN...\ y_i\ \ is\ \ T_{y_i}^1 ...$$

where $T_{y_i}^j$ is the *j*th linguistic term of the *i*th output. Layer 5 makes the output membership functions. Combination of the fifth and sixth layers provides defuzzification method. Identification process may not perform desirably if it does not include the input/output interaction. For this purpose, series-parallel configuration [13] is chosen. This identification structure considers the past output states in conjunction with the present inputs to determine the present output. The identifier with augmented inputs is represented by

$$\hat{y}(k+1) = \hat{f}\big(y(k),...,y(k-i);u(k),...,u(k-j)\big) \tag{2}$$

such that $\hat{y}(k)$ is the estimated output at time step *k*, $\hat{f}$ is identifier function, $u(k)$ and $y(k)$ are plant input and output vectors, respectively, at time step *k*.

Adaptation of neuro-fuzzy identifier to a CSTR system is essential to extract an identifier that truly models the system. For CSTR data [17] the proposed neuro-fuzzy identification is performed that the results are depicted in Fig. 2. As it can be seen the system is identified excellent and the identification error is not noticeable.

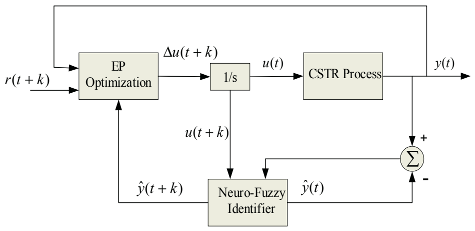


Fig. 1. Neuro-fuzzy predictive control scheme.

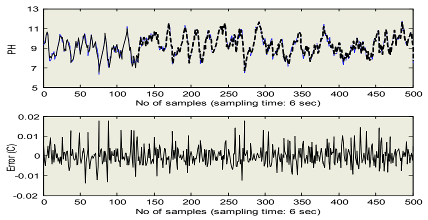


Fig. 2. Model validation: actual output and the output of the neural network model for test data3

3 Control Input Optimization

The search engine-based on evolutionary programming (EP) is used to determine the optimized control variables for a finite future time interval. The EP population consists of the individuals to present the deviation of control inputs, which is:

$$U_n = \{\Delta U_{1,n}, \Delta U_{2,n}, \ldots, \Delta U_{n_p,n}\} \tag{3}$$

such that U_n is the n^{th} generation of population, and n_p is the population size. The i^{th} individual is written by

$$\Delta U_{i,n} = [\Delta u_1^{\,i,n}, \ldots, \Delta u_m^{\,i,n}],\, for\;\; i = 1,2,\ldots,n_p \tag{4}$$

where m is number of inputs. The $\Delta u_j^{\,i,n}$ is the j^{th} vector of the i^{th} generation as in the following

$$\Delta u_j^{i,n} = [\Delta u_j^{i,n}(1) \ldots \Delta u_j^{i,n}(n_u)]^T,\, for\; j = 1,\ldots,m \tag{5}$$

such that n_u is the number of steps in the discrete-time horizon for the power unite input estimation that is defined by

$$n_u = N_u - N_1 \tag{6}$$

where N_1 is the start time of prediction horizon and N_u is the end time of the input prediction. The individuals of input deviation vector belongs to:

$$\Delta u_j^{i,n}(.) \in [\Delta u_{j,\min}, \Delta u_{j,\max}] \tag{7}$$

The cost function of the i^{th} individual in the population is defined by

$$f_{i,n} = \sum_{k=1}^{n_y} \left\| r(t+k) - \hat{y}_{i,n}(t+k) \right\|_R^2 + \sum_{k=1}^{n_u} \left\| \Delta U_{i,n}(k) \right\|_Q^2 + \left\| r(t) - \hat{y}(t) \right\|_R^2 \tag{8}$$

where $r(t+k)$ is the desired reference set-point at sample time of $t+k$, and $\hat{y}_{i,n}(t+k)$ is the discrete predicted plant output vector which is determined by applying $\Delta U_{i,n}(k)$ into the locally-linear fuzzy identifier for time horizon of $n_y = N_2 - N_1$. The $\Delta U_{i,n}(k)$ in (8) is the k^{th} of the i^{th} individual in the n^{th} generation. The input deviation vectors is determined in a smaller time window of n_u as in (6) such that $n_u \le n_y$. The inputs of the identifier stay constants after $t + n_u$. The maximum, minimum, sum and average of the individual fitness in the n^th generation should be calculated for further statistical process by

$$f_{\max}\big|_n = \{ f_{i,n} \big| f_{i,n} \ge f_{j,n} \ \forall f_{j,n}, j = 1,\dots,n_p \} \ \ , \ \ f_{\min}\big|_n = \{ f_{i,n} \big| f_{i,n} \le f_{j,n} \ \forall f_{j,n}, j = 1,\dots,n_p \} \tag{9}$$

$$f_{sum}\big|_n = \sum_{i=1}^{n_p} f_{i.n} \quad , \quad f_{avg}\big|_n = \frac{f_{sum}\big|_n}{n_p} \tag{10}$$

In mutation, each element of the parent individual as in (5) provides a new element by adding a random number such as

$$\Delta u_j^{i+n_p,n}(k) = \Delta u_j^{i,n}(k) + N(\mu, \sigma_{i,j}^2(n)) \quad for \quad i = 1,2,\dots,n_p \quad j = 1,2,\dots,m \quad k = 1,2,\dots,n_u \tag{11}$$

such that $N(\mu, \sigma_{i,j}^2(n))$ is Gaussian random variable with mean $\mu = 0$ and variance of $\sigma_{i,j}^2(n)$. The variance of the random variable in (11) is chosen to be

$$\sigma_{i,j}^2(n) = \beta(n)(\Delta u_{j,\max} - \Delta u_{j,\min}) \frac{f_{i,n}}{f_{\max}\big|_n} \tag{12}$$

where $\beta(n)$ is the mutation scale of the population such that $0 < \beta(n) \le 1$. After mutation, the fitness of offspring individuals are evaluated and assigned to them.

The generated new individuals and old individuals produce a new combine population whit size of $2n_p$. The i^th individual $\Delta U_{i,n}$ competes with j^{th} individual $\Delta U_{j,n}$, such that $j = 1,2,\dots,p$. The number of individuals to compete whit is a fixed number p. The p individuals are selected randomly whit uniform distribution. The result of this competition is a binary number $v_{ij,n} \in \{0,1\}$ to represent *lose* or *win*, and is determined by

$$v_{ij,n} = \begin{cases} 1 & if \lambda_{j,n} < \dfrac{f_{j,n}}{f_{j,n} + f_{i,n}} \\ 0 & otherwise \end{cases} \tag{13}$$

such that $\lambda_{j,n} \in [0,1]$ is a randomly selected number with uniform distribution, and $f_{j,n}$ is the fitness of the j^{th} selected individual. The value of $v_{ij,n}$ will be set to 1 if according to (13) the fitness of the i^{th} individual is relatively smaller than the fitness of the j^{th} individual. To select the survived individual, a weight value is assigned to each individual by

$$w_{i,n} = \sum_{k=1}^{p} v_{ij,n} \qquad for \quad i = 1,2,\ldots,2n_p \tag{14}$$

The n_p individuals whit the highest competition weight $w_{i,n}$ are selected to form the *(n+1)*th generation. This newly formed generation participates in the next iteration. To determine the convergence of the process, the difference of maximum and minimum fitness of the population is checked against a desired small number $\varepsilon > 0$ as in

$$f_{\max}\big|_n - f_{\min}\big|_n \le \varepsilon \tag{15}$$

If this convergence condition is met, the mutation scale with the lowest fitness is selected as sequence of n_u input vectors for the future time horizon. An adaptive mutation scale provides a change of mutation probability according to the minimum fitness value of n_p individuals in the *(n+1)*th generation. The mutation scale for the next generation is determined by

$$\beta(n+1) = \begin{cases} \beta(n) - \beta_{step} & if\ f_{\min}\big|_n = f_{\min}\big|_{n+1} \\ \beta(n) & if\ f_{\min}\big|_n < f_{\min}\big|_{n+1} \end{cases} \tag{16}$$

where n is generation number, β_{step} is the predefined possible step change of the mutation scale in each iteration.

4 Simulation Results

The objective of the control strategy is to govern the CSTR dynamics to force the system Ph to track a certain set-points. In this system, the input is the coolant flow rate and the output is the Ph of the process [12]. The identifier is trained and initialized before the control action starts. After the initial training, identifier is engaged in the closed loop of the predictive control as in Fig. 1. The performance of the proposed controller is shown in Fig. 3. Evidently, the PH values of the plant could track the set-point values excellent. It is remarkable that to improve the transient response, one may consider a larger prediction time, or a larger population size in EP.

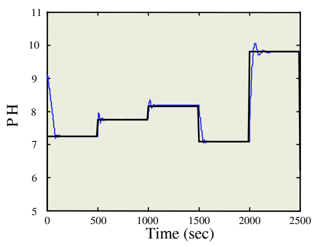


Fig. 3. Results of neuro-fuzzy predictive control of CSTR plant (PH tracking).

5 Conclusions

In this paper, a neuro-fuzzy based predictive control was applied to CSTR system. This system is a highly nonlinear process; therefore, a nonlinear predictive method, e.g., neuro-fuzzy predictive control, can be a better match to govern the system dynamics. An optimizer algorithm based on EP used the identifier-predicted outputs and determined input sequence in a time window. Simulation results of PH tracking problem in a CSTR demonstrated the effectiveness of the proposed approach.

References

1. Camacho, E.F.: Model Predictive Control, Springer Verlag (1998)
2. Garcia, C.E., Prett, D.M., and Morari, M.: Model Predictive Control: Theory and Practice-a Survey, Automatica*,* **25** (1989) 335-348
3. Badgwell, A.B., Qin, S.J.: Review of Nonlinear Model Predictive Control Applications. Nonlinear Predictive Control Theory and Practice, Kouvaritakis, B, Cannon, M (Eds.), IEE Control Series (2001) 3-32
4. Parker, R.S., Gatzke E.P., Mahadevan, R., Meadows, E.S., and Doyle, F.J.: Nonlinear Model Predictive Control: Issues and Applications, Nonlinear Predictive Control Theory and Practice, Kouvaritakis, B, Cannon, M (Eds.), IEE Control Serie*s* (2001) 34-57
5. Babuska, R., Botto, M.A., Costa, J.S.D., and Verbruggen, H.B.: Neural and Fuzzy Modeling on Nonlinear Predictive Control. a Comparison Study, Computational Engineering in Systems Science (1996)
6. Arahal, M.R., Berenguel, M., and Camacho, E.F.: Neural Identification Applied to Predictive Control of a Solar Plant. Con. Eng. Prac , **6** (1998) 333-344
7. Lennox, B., and Montague, G. Neural Network Control of a Gasoline Engine with Rapid Sampling. Nonlinear Predictive Control Theory and Practice, Kouvaritakis, B, Cannon, M (Eds.), IEE Control Series (2001) 245-255
8. Petrovic, I., Rac, Z., and Peric, N.: Neural Network Based Predictive Control of Electrical Drives with Elastic Transmission and Backlash, Proc. EPE2001, Graz, Austria (2001)
9. Ghezelayagh, H. and Lee, K.Y.: Application of Neuro-Fuzzy Identification in the Predictive Control of Power Plant. in preprints of 15th IFAC World Congress, Barcelona, Spain, June (2002)
10. Fogel, L.J.: The Future of Evolutionary Programming. Proc. 24th Asilomar Conference on Signals, Systems and Computers. Pacific Grove, CA (1991)
11. Lai, L.L.: Intelligent System Application in Power Engineering: Evolutionary Programming and Neural Networks. John Wiley & Sons Inc., New York, USA (1998)
12. Wu, Q., Wang, Y.J., Zhu, Q.M. and Warwick, K.: Neurofuzzy Model Based l_∞ Predictive Control of Nonlinear CSTR System. in Proc. IEEE Conference on Control Application, Glasgow, Scotland, UK (2002) 59-64
13. Narandra, K.S., Parthasarathy, K.: Identification and Control of Dynamical Systems Using Neural Networks. IEEE Trans. on Neural Networks, **1** (1990) 4-27
14. Goldberg, D.E.: Genetic Algorithm in Search, Optimization, and Machine Learning. Reading, MA, Addison-Wesley (1989)
15. Mason, A.J.: Partition Coefficients, Static Deception and Deceptive Problems for Non-Binary Alphabets. Proceedings of the 4th International Conference on Genetic Algorithms, (1991) 210-214
16. Dimeo, R. and Lee, K.Y.: Boiler-turbine Control System Design Using a Genetic Algorithm. IEEE Trans. Energy Conversion, **10** (1005) 752-759
17. ftp://ftp.esat.kuleuven.ac.be/pub/SISTA/espinosa/datasets/cstr.dat

Adaptive Neuro-fuzzy SVC for Multimachine Hybrid Power System Stability Improvement with a Long of Double Circuit Transmission Lines

Chamni Jaipradidtham

Department of Electrical Engineering, Faculty of Engineering,
Kasem Bundit University, Bangkok, Thailand
j_chamni@hotmail.com

Abstract. This paper presents the development of Neuro-Fuzzy Static Var Compen-sator (SVC)for the purpose of improving power system stability with a long of double circuit transmission lines system. The proposed control method, Neuro-Fuzzy Logic Static Var Compensator based on proportional integral information for the purpose of power system stability improvement. The test system is a two area multimachine hybrid power system consisting of 2 synchronous generators and 2 induction generat-ors connected to an infinite bus through a double circuit transmission line with SVC located at the midpoint. The SVC is installed at the transmission line that is linked between two power system areas. The input signal of fuzzy controller is the power flowing in the transmission line connected to the SVC bus. The simulation results demonstrate the good response of this proposed control. Comparison results of stabi-lity control between the conventional method of power system stabilizer (PSS)and the PSS with SVC controlled by fuzzy of a long distance power transmission system.

1 Introduction

The power system control technique has been developed with the advancement of power electronics technologies that introduce new degrees of freedom into the operation of power system. The SVC has been widely used in power system for both voltage regulation and dynamic stability enhancement. In a modern integrated power network, transient and dynamic stability is of increasing importance for secure operation of po-wer system. The principle of SVC damping-control schemes has been investigated based on damping torque analysis. The recently developed flexible ac transmission system (FACTS)devices with proper control strategy can significantly improve the transient stability margin. Amongst the available FACTS devices, the interline power flow controller (IPFC)is the most versatile one[3]and can be used to enhance system stability, fuzzy logic based PSS(FLPSS)shows great potential in increasing the dam-ping of generator oscillations, especially when made adaptive neuro-fuzzy inference system(ANFIS) based PSS is developed [2].

The ANFIS PSS uses a zero-order Sugeno-type fuzzy logic controller whose membership functions and consequences are also tuned online by the back-propagation method. Simulation results from previous study in show that fuzzy logic static var compensator (FLSVC)can greatly improve power system stability with a long transmission line system by considering two important aspects[4]. Simulation results for a one machine infinite bus system and two multimachine systems are presented to show the effectiveness of the proposed method.

2 System Model

The proposed controller, system model used in this paper is show in Fig.1(a). To study the new control strategy for the interline power-flow controller (IPFC). The multimachine power system presented in references[3]is considered for transient stability simula-tions. The power system and its detailed circuit model are show in Fig.1(a). Synchro-nous machine is equipped with an automatic voltage regulator (AVR) and a governor as shown in Fig.(c)[4]. SVC is installed at the midpoint of transmission line and the block diagram of SVC. U_{SVC} is the supplementary control signal generated by FLSVC or fuzzy logic static var compensator based on proportional integral information. U_{PSS} is stabilizing signal of excitation control system and U_{PSS} is set to zero. The supplementary control signal U_{SVC} is added to the voltage control loop of SVC[4].

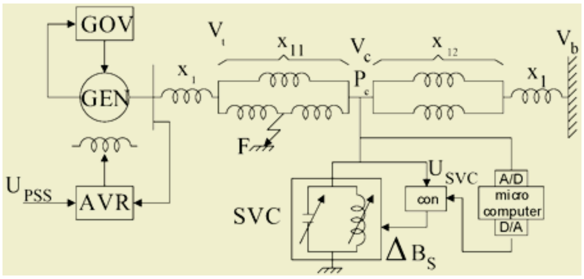


Fig. 1a. System model.

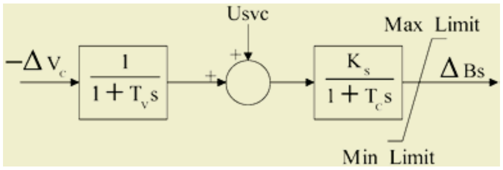


Fig. 1b. Block diagram of SVC.

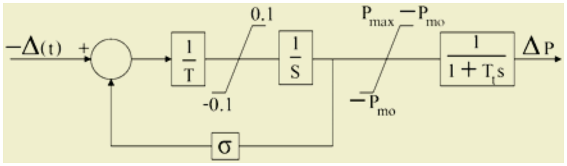


Fig. 1c. Block diagram of GOV.

The value of U_{SVC}(k) is determined at each sampling time T through the following steps:

Step 1: The values of Zs(k) and Za(k) are measured at every sampling time by the signal conditioning.

Step 2: Compute the value of vertical axis of phase plane by multiplying scaling factor α_1 and Za(k).

Step 3: The generator state is given by the point P(k) where

$$P(k) = (Z_s(k),\ \alpha_1 Z_a(k)) \tag{1}$$

Step 4: Calculate the value of radius *Dr (k)* and angle *θ (k)*using

$$Dr(k) = \sqrt{Z_s(k)^2 + (\alpha_1 Z_a(k))^2} \tag{2}$$

and

$$\theta(k) = \sin^{-1}\alpha_1 Z_a(k)/D_r(k) \tag{3}$$

Step 5: Compute the value of grade of membership in N region. $\mu_N(k)$ using membership function as show in Fig.3. The symbol α_2 is overlapping angle between deceleration control (sector A in Fig. 2 and N region in Fig.3)and acceleration control (sector B in Fig. 2 and P region in Fig.3).

Step 6 : Compute the value of grade of membership in G region. $\mu_G(k)$ using membership function. The symbol α_3 is distance factor.

Step 7 : Compute the value of supplementary control signal U_{SVC} (k) using

$$\begin{aligned} U_{SVC}(k) &= (\mu_N(k) - \mu_P(k))\ \mu_G(k)\ U_{max} \\ &= (2\mu_N(k) - 1)\ \mu_G(k)\ U_{max} \end{aligned} \tag{4}$$

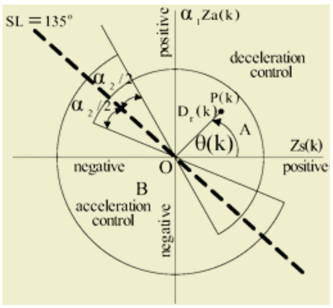


Fig. 2. Phase plane.

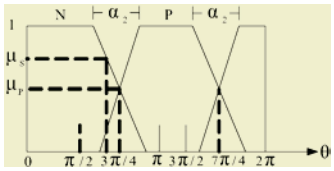


Fig. 3. Membership function of *θ (k).*

3 ANFIS PSS Design

The parameters of the identifier are updated based on the error between the estimated generator speed deviation (Δω)[2], while the parameters of the ANFIS PSS are tuned by backpropagating the error signal between (Δω) and its desired value $\Delta\omega_d$. A zero-order Sugeno-type fuzzy controller with 49 rules is used for the ANFIS PSS. The rules are built using the generator accelerating power and speed deviation (Δω) as control variables, as show in Table I. The linguistic terms used for the membership function are large positive (LP), medium positive (MP), small positive (SP), zero(ZE), small negative(SN), medium negative(MN), and large negative(LN).

Table 1. Rule Matrix.

		P_{acc} (≈ΔP_e)						
		LP	MP	SP	ZE	SN	MN	LN
Δω	LP	1.0	1.0	1.0	1.0	0.66	0.33	0.0
	MP	1.0	1.0	0.66	0.66	0.33	0.0	-0.33
	SP	1.0	0.66	0.33	0.33	0.0	-0.33	-0.66
	ZE	0.66	0.66	0.33	0.0	-0.33	-0.66	-0.66
	SN	0.66	0.33	0.0	-0.33	-0.33	-0.66	-1.0
	MN	0.33	0.0	-0.33	-0.66	-0.66	-1.0	-1.0
	LN	0.0	-0.33	-0.66	-1.0	-1.0	-1.0	-1.0

The fuzzy logic based controller can be made adaptive by being represented in a feedforward neural network form, as show in Fig. 4. In this figure, K_1, K_2 are the input scaling factors, and K_3 is the output scaling factor[2]. This network consists of four layers, with each layer representing a specific part in the ANFIS controller.

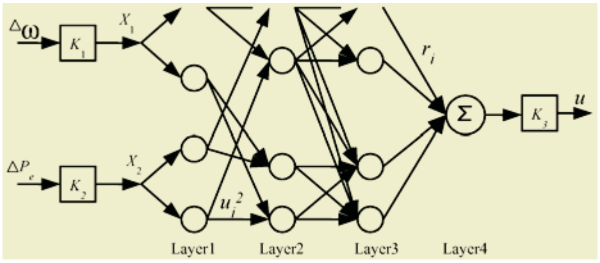


Fig. 4. Neural network model of the ANFIS.

In Fig.4, the output scaling factor K_3 is determined by the output limits of the PSS. The scaling factors $K_1$and K_2 are chosen such that the variables X_1 and X_2 are the normalized values of Δω and ΔP_e, respectively, as observed by simulation.

4 Simulation Results

In this study, to test the performance of the proposed stabilizer, simulation studies were performed on one system, is used for test purposes : a two-area, 4 machine is SG1,SG2, IG1 and IG2 represent aggregate synchronous and induction generators,10 bus system is B1-B10, L1-L19 is load, we consider the following fault cases: short circuit in the middle of one of the lines connecting buses B1 and B2, short circuit between B1 and B10, as shown in Fig.5. The FLSVC uses optimal fuzzy control parameters in[1]. The values of $\alpha_1 = 0.012$, $\alpha_2 = 90°$, $\alpha_3 = 0.23$, the performance of the proposed control method. FLSVCPI, has been investigated in two severe cases as compared to FLSVC, (1) the system under large disturbance at which the receiving end reactance X_j is 1.0 times of its designed value, and (2)the system under large disturbance at which the receiving end reactance X_j is changed to 1.448 times of its designed value.

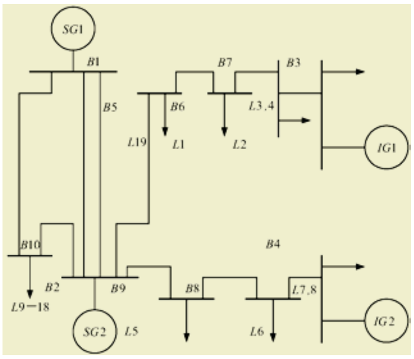


Fig. 5. Multimachine test system.

5 Conclusions

This paper presents the development of control method of SVC for improving power system stability from previous study. Fig. 6-8 show the new neuro-fuzzy logic based control scheme adapts itself to generate suitable variation of the control signal depending on the operating condition of the power system and a superior performance in comparison to the linear PI controllers is used for phase plane type fuzzy logic static var compensator(FPSVC)and PSS. ANFIS PSS will work under different operating conditions since the information about these conditions is obtained on line. It can also be adapted to different systems by adjusting the scaling factors of the ANFIS. Simulation results show the effectiveness of the proposed stabilizer in damping controllers. FLSVCPI can improve system damping significantly as compared to FLSVC.

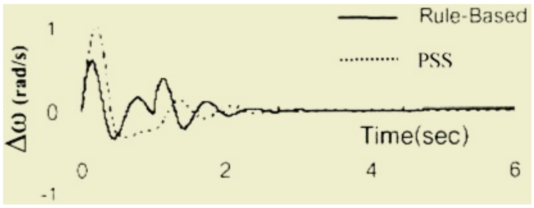


Fig. 6. Angular speed deviation responses :in case I, PSS 1 is SG1 at $X_j = 1$.

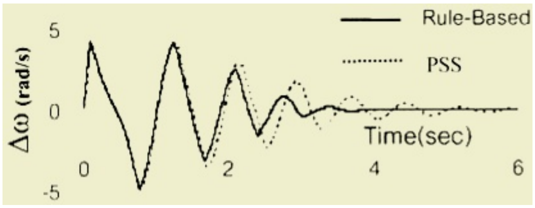


Fig. 7. Angular speed deviation responses : in case II, PSS 2 is SG1 at $X_j = 1$.

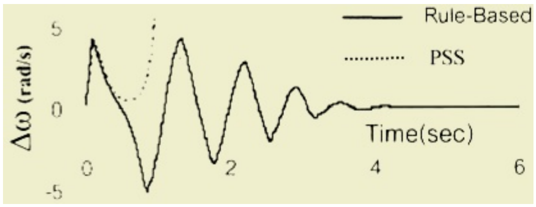


Fig. 8. Angular speed deviation responses : in case II,PSS 2 is SG1- SG2 at X_j = 1.18.

References

1. Hongesombut, K., and Leelajindakrairerk, M.: Genetic Algorithm Aided Fuzzy Controller Design of SVC for Improving Power System Stability. Ladkrabang Engineering Journal, **16** (1999) 31-36
2. You, R., Eghbali, H.J., and Nehrir, M.H.: An Online Adaptive Neuro-Fuzzy Power System Stabilizer for Multimachine Systems, IEEE TP, **18** (2003) 128-135
3. Mishra, S., Dash, P.K., Hota, P.K., Tripathy, M.: Genetically Optimized Neuro-fuzzy IPFC For Damping Modal Oscillations,IEEE TPS,**17** (2002) 1140- 1147
4. Hongesombut, K., and Leelajindakrairerk, M.: FLSVC Based on PI Information for Power System Stability Improvement With A Long Transmission Line System. 22st Electrical Engineering Conference (1999) 208-211

Application of BP Network-Based Multi-sensor Fusion Techniques in Measurement of the Unburned Carbon in Fly Ash*

Gaowei Yan[1], Gang Xie[1], Keming Xie[1], Zehua Chen[1], and Hongbing Wang[2]

[1] College of Information Engineering, Taiyuan University of Technology, Taiyuan, Shanxi 030024 China
firstygw@yahoo.com.cn
[2] Taiyuan Haitong Automation Ltd., Kexiang building, Changzhi Road, Taiyuan, Shanxi 030006, China

Abstract. Coal type has great influence on the measurement of the unburned carbon in fly ash in thermal power plant. Based on analyzing the factors mainly affect the measurement accuracy, microwave technique and capacitance technology combined with the specific resistivity of fly ash are adopt to construct the measuring system based on the multi-sensor fusion. Then the fuzzy technology is applied to decrease the temperature influence on the measuring result of ash resistivity, and BP neural network is utilized to fuse the multi-sensor information effectively. Experiment results shown that the proposed method can effectively decrease the coal type effect on the unburned carbon measurement, however at the same time the accuracy for the same coal type is increased.

1 Introduction

The combustion efficiency of the boiler is an important consideration in the pulverized-coal-fired power plant. The fly ash unburned carbon (UBC) level directly reflects the efficiency of the boiler combustion. The higher the UBC level, the higher the cost of coal, the higher the discharge of NO_x, and the higher the environmental pollution as well. The microwave-based technique [1] are widely applied at present, but there are many factors affect the stability of the result, such as the sample density, particle size and the content of different elements in the fly ash. Especially, the error is larger in case of coal type changing, and it can not reflect the change trend of the UBC. For this kind of non-linear and seriously cross-influencing system, it is necessary to adopt multi-sensor detection techniques and multi-sensor information fusion technology to obtain all-sided information of all the measured parameters. The factors that affect the unburned carbon content of fly ash is analyzed in this paper, multi-sensor is adopted to gain multi-dimension information of the fly ash; multi-sensor information fusion principle is utilized to design the structure, and the algorithm of the measuring system. The BP neural network is used to fuse the multi-sensor data, which makes the measuring system keep changing with the environment and medium parameter variation, so as to improve the measuring precision of the system effectively.

* This work is Supported by Chinese National Natural Science Foundations Project (60374029), Shanxi Province University Technology Research Project (2003204), Shanxi Province Youth Science Foundations Project(20041015), Shanxi Province Tackle Projects in Science and Technology (001043).

2 Microwave Measure Principle of UBC and Influence Factors

In the microwave electromagnetic fields, the unburned carbon particles absorb microwave energy. The energy absorbed is directly related to the UBC level. The higher the UBC level the more energy absorbed. So fly ash is a kind of dielectric with loss characteristics. Its complex dielectric constant can be expressed as $\varepsilon^*=\varepsilon'+\varepsilon''$. The real part ε' is dielectric constant. The imaginary part ε'' is loss factor, which shows the absorption of the microwave. Loss factor ε'' is related to the microwave frequency and complex propagation factor γ, where $\gamma=\alpha+\beta$. α and β are respectively attenuation constant and phase constant, which representing the attenuation on the unit distance of the microwave and quantity of phase shifts. So UBC level can be obtained by measuring signal attenuation A and phase change Φ of the microwave with certain frequency after it going through the fly ash, Formula (1) is used to calculate the UBC level,

$$UBC\%=c_0+c_1(A)+c_2(\Phi) \tag{1}$$

where c_0,c_1,c_2 are fitting constants.

In practice, this method often leads to very large error, the measuring error is beyond acceptable range, especially when the coal type changes. The main factors that influence measuring precision are: the density of sample, particle size of the fly ash, temperature, humidity and the content of different elements in fly ash. Because the ingredient of the coal are very complicated, except silicon and unburned carbon, the fly ash after burning includes many kinds of metal oxides, such as SiO_2, Fe_2O_3, Al_2O_3, CaO, MgO, etc [2]. The concentrations of metal oxides are dependent on the type of coal. Among these metal oxides, Fe_2O_3, Al_2O_3, M_gO, etc, have great influence on the absorption and phase variation of microwave. This will cause a serious influence on the measuring result.

3 Multi-sensor Fusion Measurement System

According to the analysis above, the measurement of UBC through microwave technology is a system with notable nonlinearity and serious cross-influencing variables. To gain the all-sided information of the medium, the sensors relative to parameters and environment have to be added to the system [3]. At the same time, it is necessary to apply the multi-sensor information fusion technology to efficiently fuse the gained information. For the advantages of parallel processing, fault tolerating and learning, Neural Network could simulate complex and nonlinear mapping, thus meeting the requirements of data processing through multi-sensor information fusion. Accordingly, BP network is applied to realize the multi-sensor information fusion in this paper.

3.1 Sensor Selection in Multi-sensor Fusion System

The formula (1) is obtained by linear regression, however, it can not reflect the complex nonlinear relation among attenuation const α, phase const β and complex dielectric constant ε^*. When the elements and proportion cannot be predefined, the measurement of attenuation and phase change of microwave can only reflect the effect on characteristic of dielectric from all of contents in the fly ash sample, and not only different UBC level lead to the variation of power and phase. After many experiments, we found that different kinds of fly ash with same UBC level had much different responses in high and low frequencies. Capacitance time-domain measurement is an effective method for low frequency permittivity measurement. So, we introduce the measurements of ash relational dielectric constant using capacitance method, combining it with microwave measurement to fuse the data. This method can much reduce the effect caused by the variation of coal type to the UBC level.

Specific resistivity is another important parameter in representation of fly ash [4], which has close relation with carbon content and contents of metal oxides in the fly ash. It is nearly in the range of 10^4~$10^{12}\Omega\cdot$cm according to different conditions. Because it is not a single value function with temperature, it is necessary to be converted into a value, which has no relation with the temperature. Fuzzy logic is applied to map the range of 10^4~$10^{12}\Omega\cdot$cm into fuzzy universe of –6~6, the temperature changing range 0~150°C is mapped into fuzzy universe of –4~4. The division of the fuzzy universe is shown in Fig.1. Sugeno fuzzy relation model is constructed according to the specific resistivity-temperature curve and experimental data, then specific resistivity of ash is mapped into 0~1 by fuzzy reasoning to take part in the neural network computing.

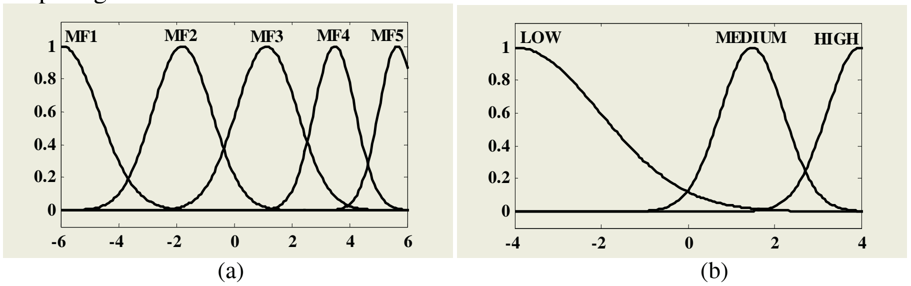


Fig. 1. (a) Fuzzy universe definition of fly ash specific resistivity. (b) Fuzzy universe definition of temperature.

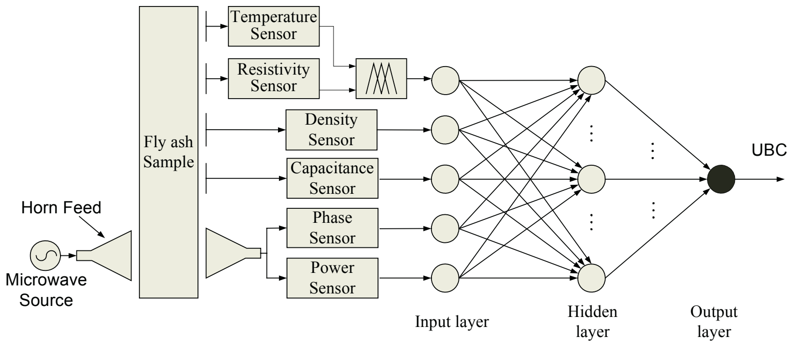


Fig. 2. A schematic diagram of the multi-sensor fusion measuring system based BP.

Because the density and particle size of fly ash are related, we configured density sensor to eliminate the influence of density and granularity to the measured result in measurement system. Fig.2 shows these sensors configuration and systematic structure of the multi-sensor fusion measurement system.

3.2 BP Neural Network Structure

The multi-sensor fusion system based on BP network presented in this paper has pattern classification and function approximation abilities. First, it distinguishes different fly ash types according to sensor information, then approximate to the nonlinear measurement result. We selected three-layer BP neural network, the hidden layer is Sigmoid function, and the output layer is linear function. The neuron number of input layer is 5, which corresponds with ash resistivity, density, capacitance, power and phase sensor respectively. The neuron number of output layer is 1, which corresponds with UBC level. The neural network is trained and tested by the sampled data of fly ash. The 25 neuron of hidden layer is final determined by experiment.

4 Experiment and Results

For the different kinds of coal, the fusion result of one kind of fly ash from single power plant can not satisfy different requirement. We have collected 60 kinds of ash samples came from 20 different districts and each includes 3 working conditions. Then 30 new samples are gained by mixing 60 samples randomly. Ground them separately, they are filtrated in 5 different densities. This finally resulted in 450 kinds of ash sample. At the same time, we chose one ash sample with high carbon level, filtrated the particle with more carbon and ground it, then we got 20 kinds of ash sample by mixing the sample with pure ash. These would be used to test the result of the same type sample.

Momentum law is applied to train the BP neural network, 300 ash samples are chosen from 450 ash samples for training and the rest for testing. Then we compute and compared the result got by other 3 ways: power attenuation, Formula (1) and multi-sensor fusion of BP network. Fig.3 (a) shows the linear regression of the testing result of the 150 ash samples by the way of power attenuation, which has the standard error 2.35. Fig.3 (b) shows the testing results by Formula (1), and the standard error is 1.67. Fig.3 (c) shows the testing results get from the BP neural network by fusing the data from power, phase and density sensor, and the standard error is 1.18. Fig.3 (d) shows the linear regression of the fusion result of the 5 kinds of sensor mentioned by this paper, and the standard error is only 0.48. It can be seen from the result that BP network based multi-sensor fusion system can figure the character of fly ash from different dimension. It is effective in decreasing the influence caused by coal type variation and other factors, and it is also effective in improving the precision of measurement of unburned carbon in fly ash.

After training and tests by BP network, we have measured 20 ash samples of the same kind fly ash, and contrasted them with Formula (1) method. In the Fig.4 (a), the result indicates that the measuring precision for same kind fly ash improves notably. The ash example measuring results with the different ash type is showed in Fig.4 (b). From the measuring data, it can be seen that the measuring error by BP network multi-sensor fusion system is much smaller than the method of power attenuation combined with phase variation, and it can meet the requirement of practical applications.

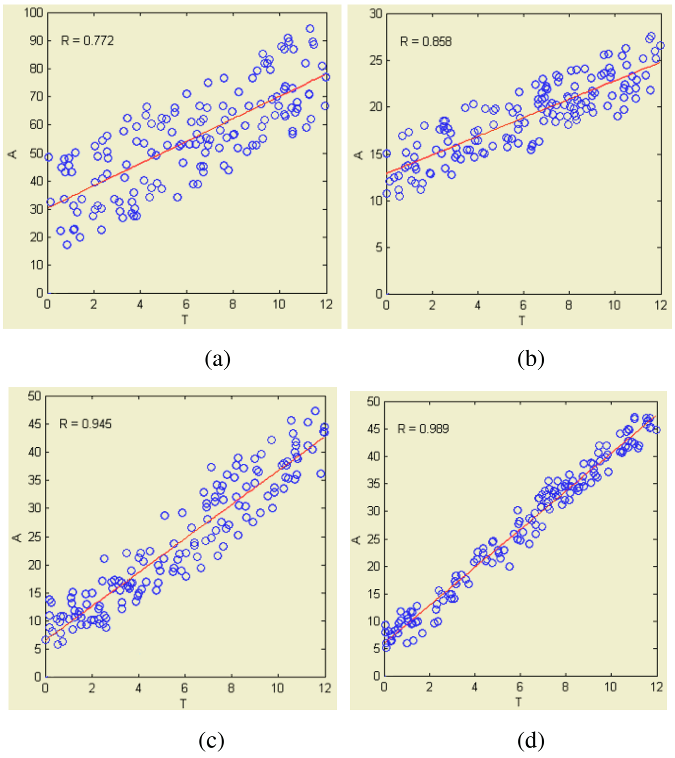


Fig. 3. Linear regression of testing results for: (a) Power attenuation method; (b) Formula (1) method; (c) BP network fusion result of 3 sensors; (d) BP network fusion result of 5 sensors. The solid line is the ideal function. "○" is the points of sample.

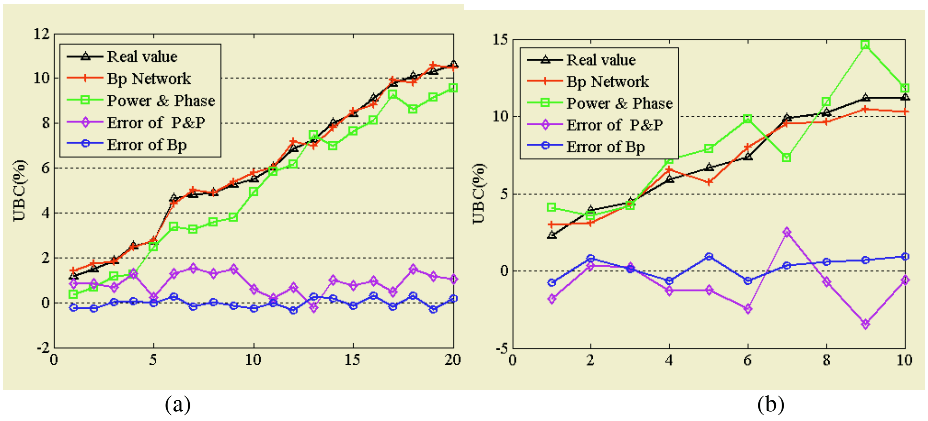


Fig. 4. The contrast of the measuring result between BP network fusion and Formula (1) for: (a) same kind fly ash; (b) different kind fly ash. "Δ" real value; "+" results of BP network fusion; "□" results of formula (1); "◊" error of Formula (1); "○" error of BP network fusion.

5 Conclusions

This paper deeply analyses a kind of non-linear systems with serious cross-influencing variables: unburned carbon content of the fly ash measuring system. The proposed system takes several kinds of multi-sensor to examine the fly ash to get the expression of the multidimensional information, and then construct BP network-based multi-sensor information fusion system which realize measurement of non-linear parameters, Experiment results shown that the proposed method can effectively decrease the coal type effect on the unburned carbon measurement, however the accuracy for the same coal type is increased at this time.

References

1. Evans, T.G., Yip, V., Cutmore, N.G.: Microwave Technique for On-Line Determination of Unburned Carbon in Fly Ash. IEEE Asia-Pacific Microwave Conference, Adelaide (1992)
2. Styszko, K., Grochowiaka, Henryk Jankowski: Characterization of the Coal Fly Ash for the Purpose of Improvement of Industrial On-Line Measurement of Unburned Carbon Content, Fuel **83** (2004) 1847-1853
3. Durrant Whyte, H.F.: Elements of Sensor Fusion. IEE Colloquium on Published (1991)
4. Bickelhaupt, R.E.: Volume Resistivity-Fly Ash Composition Relationship, Sci. Technol., **9** (1975) 337

Classification of Nuclear Receptor Subfamilies with RBF Kernel in Support Vector Machine[⋆]

Jun Cai^{1,2} and Yanda Li^{1,2}

^{1} Institute of Bioinformatics
Tsinghua University Beijing 10084, China
^{2} Department of Automation, Tsinghua University Beijing 10084, China
caijun99@mails.tsinghua.edu.cn

Abstract. Nuclear receptors (NRs) are ligand-inducible transcription factors that regulate diverse functions as a superfamily of crucial medical significance. Because of their involvement in many physiological and pathological processes, the development of methods to infer the different NR subfamilies has become an important goal in biomedical research. In this paper we introduce a sequence-based computational approach-Support Vector Machine to classify the 19 subfamilies of NRs. We use 4-tuple residue composition instead of dipeptide composition to encode the NR sequences. The overall predictive accuracy about 96% has been achieved in a five fold cross-validation.

1 Introduction

Nuclear receptors (NRs) are ligand-inducible transcription factors that regulate diverse functions, such as homeostasis, differentiation, embryonic development and organ physiology [1]. They are implicated in many important diseases like cancer, diabetes, and osteoporosis, and, therefore, are targets for pharmaceutical industries with similar importance as the G protein-coupled receptors (GPCRs), ion channels, or kinases [2]. These nuclear hormone receptors form an evolution-related superfamily of crucial medical significance. NR superfamily has been subdivided into different families and subfamilies because of their binding with different ligand types [3]. Inferring the diversity of different NR functions has become an important goal in biomedical research. Whereas, not many experimentally determined structure data are available for NR. It is of great interest to research biologists and pharmaceutical companies to develop an accurate sequence-based prediction of NR function and superfamily diversity instead of concerning about 3-D structure [4]. Recently, Bhasin and Raphava have classified NRs superfamily into four families with amino acid composition and dipeptide composition using support vector machines [5]. Classes of thyroid hormone like, HNF4-like, estrogen like and fushi tarazu-F1 like were considered in their work. However, there are many important subfamilies derived from a family. For example, thyroid hormone like

[⋆] This work was funded by the National Natural Science Grant in China (No.60171038 and No.60234020) and the National Basic Research Priorities Program of the Ministry of Science and Technology (No.2001CCA0). J.Cai also thanks Tsinghua University Ph.D. Grant for the support.

family consists of many subfamilies such as thyroid hormone, retinoic acid, vitamin D3-like and so on. These diversiform subfamilies regulate various biological functions in cellular process.In the current study, we try to apply sequence-based Support Vector Machine (SVM) method to approach the problem of classifying NR proteins at subfamily level, as an extension of NR classification at family level. The results show that good performance of classification of 19 NR subfamilies is obtained with a cross-validation test.

2 Materials and Methods

2.1 Sequence Data

The data for subfamilies classification of NRs was taken from nucleaRDB database (Jul 2004 release)[6]. All putative/orphan sequences and fragments were excluded and redundancy was reduced so that pair-wise sequence identity is relative low. About 400 protein sequences compose our initial data set. According to pharmacological knowledge, these NRs belong to various subfamily components. Any subfamily that contained less than 6 proteins was dropped for further consideration and we divided these NRs into 19 subfamilies. A simplified view of these NR subfamilies is presented in Table 1.

2.2 Support Vector Machine

SVM is a popular machine learning algorithm based on recent advances in statistical learning theory [7, 8]. This algorithm first maps data into a high-dimensional feature

Table 1. View of the NR subfamilies.

Subfamily Class	Description	Number of NRs
1	Knirps like	7
2	DAX like	12
3	Thyroid hormone	14
4	Retinoic acid	17
5	Peroxisome proliferator activated	22
6	REV-ERB	16
7	RAR-related orphan receptor	15
8	Ecdysone-like	30
9	Vitamin D3-like	21
10	Hepatocyte nuclear factor 4	13
11	Retinoic acid X	36
12	Orphan nuclear receptors TR2, TR4	7
13	Tailless-like	15
14	COUP-TF-like	16
15	Estrogen	52
16	Estrogen-related	7
17	Glucocorticoid-like	47
18	Nerve Growth factor IB-like	12
19	Fushi tarazu-F1 like	29

space, and then establishes a hyperplane as the decision-making surface, which maximizes the boundary between two classes. The actual mapping is achieved through a kernel function, making it easy to implement and fast to compute. Four popular kernel functions are:

$$\text{linear kernel: } K(u,v) = u^T v \tag{1}$$

$$\text{polynomial kernel: } K(u,v) = (\gamma u^T v + r)^d, \gamma > 0 \tag{2}$$

$$\text{RBF kernel: } K(u,v) = exp(-\gamma \|u - v\|^2), \gamma > 0 \tag{3}$$

$$\text{sigmoid kernel: } K(u,v) = tanh(\gamma u^T v + r), \gamma > 0 \tag{4}$$

where u,v are data vectors in SVM. In principle, SVM is a two-class classifier. With the recent improvements, the SVM can directly cope with multi-class classification problem now. The software used to implement SVM was libSVM, which can be downloaded from http://www.csie.ntu.edu.tw/˜cjlin/libsvm. We selected ‘one-against-the rest’ multi-class strategy in libSVM to deal with our problem instead of ‘one-againse-one’ strategy.

2.3 Data Representation

Instead of amino acid composition or protein’s dipeptide composition [9, 10], we used simplified 4-tuple residues composition to encode the amino acid sequences. The sequence alphabet was first reduced from 20 amino acids to six categories of biochemical similarity: [I,V,L,M], [F,Y,W], [H,K,R], [D,E], [Q,N,T,P] and [A,C,G,S] [11]. After this reduction, there were 6^4=1296 possible substrings of length 4. We extracted and counted the occurrences of these substrings from a NR sequence string in a sliding window fashion. For a given protein sequence, the 4-tuple residues composition is simply an integral vector of length 1296, in which each bit indicates the counts the corresponding length-4 substring occurs in the protein. This simplified 4-tuple residues composition method was previously used to predict protein-protein interactions from protein sequence [12].

2.4 Scaling Data

Scaling data vectors before applying SVM is very important. The main advantage of scaling data is to avoid attributes in greater numeric ranges dominate those in smaller numeric ranges. Another advantage is to avoid numerical difficulties during the calculation. Because kernel values usually depend on the inner products of feature vectors, e.g. the linear kernel and the polynomial kernel, large attribute values might cause numerical problems. In this work, we linearly scaling each attribute in data vector to the range [-1, +1].

2.5 Selecting Kernel Functions

The linear kernel is a special case of RBF. The linear kernel with a penalty parameter has the same performance as the RBF kernel with some proper parameters. Similarly, the sigmoid kernel behaves like RBF for certain parameters. In addition, the number of hyperparameters influences the complexity of model selection. The polynomial kernel has more hyperparameters than the RBF kernel. So in our current study, we used RBF kernel function as a reasonable choice.

2.6 Performance Assessment of Classification

Cross-validation within the original data set was utilized to provide a nearly unbiased estimate of the prediction error rate [13]. The performance of classifying the subfamilies of NRs was evaluated using 5-fold cross-validation. The dataset of NRs was divided into five subsets of approximately equal size. Sequentially one subset was tested using the classifier trained on the remaining 4 subsets. Thus, each NR instance was predicted once so the cross-validation accuracy was the percentage of data which are correctly classified. Summary statistics assessing the performance of classification were calculated, including sensitivity, specificity and accuracy of classification. More formally, these definitions are:

$$\text{sensitivity} = \frac{\text{true positives}}{\text{true positives + false negatives}} \tag{5}$$

$$\text{specificity} = \frac{\text{true positives}}{\text{true positives + false positives}} \tag{6}$$

$$\text{accuracy} = \frac{\text{true positives + true negatives}}{\text{true positives + false negatives + true negatives + false positives}} \tag{7}$$

3 Results

3.1 Parameters Tuning

There are two parameters while using RBF kernels: kernel parameter γ and penalty parameter C. We used the ‘grid-search’ strategy embedded in libSVM on choice of parameters γ and C for SVM model. The best result was achieved when C = 8 and γ = 0.0001.

3.2 Classification of NR Subfamilies

The results of NR classification are demonstrated in term of sensitivity and specificity for each subfamily, as shown in Table 2. The prediction for the largest two class, estrogen and glucocorticoid-like subfamilies (Class 15 and Class 17) reaches about 99% sensitivity and 98% specificity. The sensitivity and specificity of most subfamily class are over 91%. The performance of classification is good and balanced except for the first class of knirps-like subfamily with low sensitivity of 60%.

3.3 Comparisons with Different Methods

In order to check the performance of our method, we tried to classify the subfamilies via widely used nearest neighbor(NN) and decision tree algorithms and compared our simplified 4-tuple residues composition with protein’s dipeptide composition via SVM.

Table 2. Sensitivities and specificities of classification for each subfamily.

Subfamily Class	Sensitivity(%)	Specificity(%)
1	60	100
2	100	85.7
3	92.9	100
4	100	94.4
5	100	95.7
6	100	93.8
7	100	100
8	100	96.8
9	100	95.5
10	77.0	100
11	91.7	94.3
12	85.7	85.7
13	100	93.8
14	93.8	100
15	100	98.1
16	100	100
17	97.9	97.9
18	91.7	84.6
19	96.6	100

We measured ‘nearest’ by Euclidean distance in NN and used C4.5 strategy in decision tree [14]. Overall classification accuracies of NR subfamilies in comparison are summarized in Table 3. The accuracy of C4.5 decision tree or NN is about 10% or 6% lower than that of our methods. The use of simplified 4-tuple residues composition in SVM achieved better performance than using protein’s dipeptide composition.

4 Conclusion

In this paper, we introduced SVM method for recognizing the 19 subfamily of NRs. Simplified 4-tuple residue composition was used to encode the NR sequences. The rate of correct identification obtained in five fold cross-validation reaches about 96%. Comparisons with other methods imply that our method do a good performance and we can

Table 3. Performance comparisons with different algorithms.

Method	Accuracy(%)
Our method	96.1
SVM with PDC[a]	93.3
NN[b]	89.7
Decision Tree	86.3

[a] protein’s dipeptide composition
[b] the nearest neighbor algorithm

predict the type of NRs to a considerably accurate extent. It is anticipated that the establishment of such method will speed up the pace of identifying subfamilies of orphan NRs and facilitate drug discovery for diseases.

References

1. Mangelsdorf, D.J., Thummel, C., Beato, M., Herrlich, P., Schutz, G., Umesono, K., Blumberg, B., Kastner, P., Mark, M., Chambon, P: The nuclear receptor superfamily: the second decade. Cell, **83** (1995) 835-839
2. Hopkins, A.L., Groom, C.R.: The druggable genome. Nature Rev Drug Discov, **1** (2002) 727-730
3. Robinson-Rechavi, M., Garcia, H.E., Laudet, V.: The nuclear receptor superfamily. J Cell Sci, **116** (2003) 585-586
4. Robinson-Rechavi, M., Laude, V.: Bioinformatics of nuclear receptors. Methods Enzymol, **364** (2003) 95-118
5. Bhasin, M., Raghava, G.P.: Classification of nuclear receptors based on amino acid composition and dipeptide composition. J Biol Chem, **279** (2004) 23262-23266
6. Horn, F., Vriend, G., Cohen, F.E.: Collecting and harvesting biological data: the GPCRDB and NucleaRDB information systems. Nucleic Acids Res, **29** (2001) 346-349
7. Vapnik, V.N.: The Nature of Statistical Learning Theory. Springer-Verlag, Berlin Heidelberg New York (1995)
8. Vapnik, V.N.: Statistical Learning Theory. Wiley, New-York (1998)
9. Chou, K.C.: A novel approach to predicting protein structural classes in a (20-1)-d amino acid composition space. Proteins Struct. Funct. Genet., **21** (1995) 319–344
10. Van Heel, M.: A new family of powerful multivariate statistical sequence analysis techniques. J Mol Biol, **220** (1991) 877-887
11. Taylor, W.R., Jones, D.T.: Deriving an amino acid distance matrix. J Theor Biol, **164** (1993) 65-83
12. Gomez, S.M., Noble, W.S., Rzhetsky, A.: Learning to predict protein-protein interactions from protein sequences. Bioinformatics, **19** (2003) 1875-1881
13. Bengio, Y., Grandvalet, Y.: No unbiased estimator of the variance of k-fold cross-validation. J Mach Learn Res, **5** (2004) 1089-1105
14. Ross, Q.: C4.5: Programs for Machine Learning. San Mateo, CA (1993)

Prediction of Contact Maps in Proteins Based on Recurrent Neural Network with Bias Units

Guixia Liu, Chunguang Zhou, Yuanxian Zhu, and Wengang Zhou

College of Computer Science and Technology, Jilin University,
Changchun, Jilin 130012, China
lgx1034@163.com

Abstract. Prediction of inter_residue contact maps may be seen as a strategic step toward the solution of fundamental open problems in structural genomics. Predicting the contact map of a protein of unknown structure can give significant clues about the structure of and folding mechanism of that protein. In this paper, we focus on prediction of contact maps in proteins based on recurrent neural network with bias units and have gotten a better prediction results.

1 Introduction

A fundamental unsolved problem in computational molecular biology is the prediction of the three dimensional structure of a protein from its sequence of amino acids. However, full molecular modeling to find the structure is at present intractable, and so intermediate steps such as inter-residues contact prediction have been developed rapidly. A variety of approaches to contact prediction have been taken such as finding correlated interchanges in multiple sequence alignments [1] likelihood matrix methods[2]; and neural networks methods[3]. Different methods focusing on distances or contacts among residues in the protein have been proposed for predicting the structure starting from the sequence, contacts among residues constrain protein folding and characterize different protein structures. Therefore, the prediction of residue contacts in proteins is an interesting problem whose solution may be useful in protein-folding recognition and de novo design. Structural similarity between a pair of proteins can be detected by inspecting their contact maps without searching for all their possible orientation. Secondary structure can easily be detected from contact maps, the structure of protein can be recovered using contact maps [4].

There are a variety of measures of residues contact used in the literature. Some use the distance between the C_α-C_α atoms [5], while others prefer to use the distance between the C_β-C_β. Contact maps are two dimensional, binary representations of protein structures. For a protein with N residues, the contact map for each pair of amino acids k and l ($1 \leq k, l \leq N$), will have a value $C(k,l)=1$, if a suitably defined distance $d(k,l)<d_{thr}$ where d_{thr} is a user-defined threshold distance between the amino acids, and $C(k,l)=0$ otherwise. We consider two residues to be in contact if the distance between their C_α atoms is less than 8Å.

2 Background Material

2.1 Database

In this work, we use a large set of non-homologous proteins of solved 3D structure. The set which consists of proteins with an identity value <25%is extracted from the Protein Data Bank (PDB) using PDB_select list [6] of October 2004, it is the most recent 25% threshold list and containing 2485 proteins. This set is first reduced by excluding those proteins which has non-standard amino acid residues, and then the set is further reduced by removing those proteins whose backbone is interrupted. We use the DSSP program [7] on all the PDB files to extract 3D coordinates and to assign secondary structures, then we remove also sequences on which DSSP crashes. Finally the set contains 2095 proteins. Furthermore, 105 proteins randomly selected are given in Table 1. The sub-dataset is divided into 5 classes: $L<100$, $100\leq L\leq 199$, $200\leq L\leq 299$, $300\leq L\leq 399$, $L\geq 400$, according to their residue length. The contacts between residues which are less than five residues separation are not included while we train or test the networks.

Table 1. The database of proteins used to train and test the recurrent neural network

L<100	100≤-L≤-199	200≤L≤299	300≤ L≤399	L>400
1EKTA	1PSRA	1O3SA	1ST4A	1KP0A
1MHMB	1L1PA	1EL6A	1T5JA	1KS8A
1KIKA	2PVBA	1HAVA	1FZEF	1PMOC
1URQA	1I4VA	1FUJA	1UF4A	1FSU_
11OAIA	1LUQB	1BU2A	1B3OA	1H6VC
1JNIA	1N32I	1SE1A	1SIG_	1GLLO
1PCFA	1UUZB	1XO1A	1BOB_	1N4WA
1IGL_	1H31B	1P1XA	1EFPA	1QATA
1OS6A	1HDKA	1OL1B	1D3VA	1M1NB
1PL5A	1K3KA	1JKUA	1T2DA	1AYL_
1C01A	1VKBA	1OJRA	1JIKA	1UWKA
1E44A	1LM8V	1R52B	1I9YA	4AAHA
1WKT_	1V2BB	1HM7A	1R5YA	1M6BB
1CY5A	1BYSA	1I78B	1NOYA	1EQRA
1FBR_	1N1FA	1NH1A	3SIL_	1WD9A
1TTG_	1I12A	1O58A	1MUWA	1SLY_
1IM3D	1SK3A	1A3GA	1T9GC	1SU8A
1PA4A	1QOLA	1SM4A	1QOPB	1BJT_
1HS7A	1KD6A	1GC1G	1I24A	1H2WA
1LRIA	1PVMB	1U4GA	1QHDA	1JB0B
2EZL_	1M4UA	1GLV_	1JFBA	1HE8A

Protein length(L) is the residue number of the covalent structure.

2.2 Features

In our work, we capture two features of the amino acids: predicted secondary structure and hydrophobicity. The predicted secondary structures for each protein are obtained by using PSIPRED, we use a 3 bit binary to denote the 6 possible secondary structure pairs, because a amino acid residue has three possible secondary structures:α-helix, β-sheet and coil. Hydrophobicity is a measure of nonpolarity of the side chains. As the nonpolarity (hydrophobicity) of the side chain increases, it avoids being in contact with water and buried within the protein nonpolar core. This is seen as the essential driving force in protein folding. This quantity is used to encode residue specific information to the network. Since the hydrophobicity of a residue affects the non-covalent bonding between its surroundings, it can be a contributing factor to contact decision of that residue with others. ROSEF hydrophobicity scale is used since it is one of frequently used scale [8].

3 Recurrent Neural Network with Bias Units and Input Encoding

In our work, we use a recurrent neural network with bias units to deal with prediction of contact maps in proteins. The architecture of the network is depicted in Figure 1. This topology consists of three layers of neurons: one output neuron representing the contact propensity, one hidden layer containing ten neurons and one input layer with different number of neurons depending on the amount of information encoded, we use 40 neurons for 5 residue pairwise, 4 neurons for residue classification according to hydrophobicity, polar, acidic and basic, 3 neurons for secondary structure information. This topology also has ten conjunction units and two bias units. Here we use Gaussian function as the activation transfer function of the network:

$$f(x) = e^{\frac{-x^2}{2}} \tag{1}$$

therefore, while the input pattern is $x=(x_1,x_2,\cdots x_n)^T$, the output of our net can be computed as:

$$L(x) = \sum_{j=1}^{p} V_j \bullet b_j - \gamma \tag{2}$$

$$c(x)=f(L(x)) \tag{3}$$

Where j∈ (1,2,···,p), p is the number of hidden neurons, γ is the weight between the bias unit and the output layer. V_j is the connecting weights between the hidden layer and the output layer, and b_j is the output of the hidden layer, can be computed as below:

$$S_j^t(x) = \sum_{i=1}^{n} W_{ij} \bullet x_i + \sum_{k=1}^{p} U_{kj} \bullet b_j^{(t-1)} - \theta_j \tag{4}$$

$$b_j^t(x) = f(S_j^t(x)) \tag{5}$$

Where i∈ (1,2,···,n), n is the number of input neurons, j∈(1,2,···,p), W_{ij} is the connecting weights between the input layer and the hidden layer, U_{kj} is the connecting weights between the conjunction units and the hidden layer, θ_j is the weight between bias unit and the hidden layer, b^t_j and $b^{(t-1)}_j$ are the output of the jth hidden neuron with its input and this time and last time, separately.

A major characteristic of the neural network is that they have ten conjunction units which are used to take into account the influence of neighbors pairing. Another important characteristic is that they use new input encoding. We implement the neural networks taking both parallel pairings and anti-parallel pairings into consideration. Here we use a 3-residue-long input window. It is said that we should encode the couples formed by the residues both in positions {i–1, j–1},{i, j},{i+1, j+1} (parallel pairing) and in positions {i–1, j+1},{i, j},{i+1, j–1} (antiparallel pairing). We denote each possible couples by a 8 bits binary (shown in table 1), so the five possible combinations ({i–1, j–1},{i, j},{i+1, j+1},{i–1, j+1},{i+1, j–1}) of the couples require 40 (8*5) input neurons. We need 4 bits binary (input neurons) to encode the 10 possible pair cases. We add the secondary structure information as input to the network. It requires 3 bits binary for 6 possible cases. Then we use 1 bit binary to encode whether the residue separation between residue i and residue j is below 5, by doing this, the network can easily predict the alpha helix accurately.

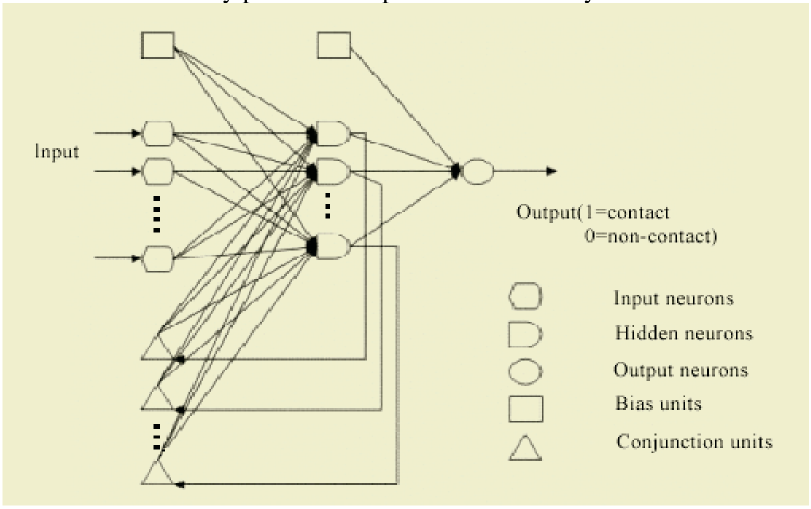


Fig. 1. The architecture of the neural network

4 Results and Discussions

We are actually interested in the network capability of predicting residue contacts. Therefore accuracy *(*A) of the network is defined as the ratio of the correctly predicted contacts by the network to the actual number of contacts in a protein and calculated according to:

$$A=N^*_C/N_C \tag{6}$$

Where N_c^* is the number of correctly predicted contacting residues by the network; Nc is the actual number of contact within the protein.

A random predictor makes Nc number of guess in order to predict the contacting pairs, assuming that there are Np number of residue pairs in which Nc of them are contacting. Therefore, its performance *(*Ar) is calculated by the following formula:

$$A_r=N_c/N_p \tag{7}$$

Where N_P=(L-4)(L-3)/2,L is the protein length.

In order to calculate the improvement over a random predictor, accuracy A of the network is divided to performance of the random predictor A. Improvement over a random predictor is denoted by R and calculated according to the formula:

$$R=A/Ar \tag{8}$$

The performance of the network is tested on five sets. All performance results are shown in table 2.

Table 2. Accuracy of predicted contact maps

RESIDUE LENGTH	L < 100	100 ≤ L ≤ 199	200 ≤ L ≤ 299	300 ≤ L ≤ 399	L > 400
A	0.11	0.09	0.087	0.071	0.065
R	3.75	3.54	4.87	5.32	7.93
ALL PROTEINS		A=0.08		R=4.65	

5 Conclusions

In this paper, we propose a recurrent neural network with bias units for predicting contact maps within 105 proteins and have gotten a better result. In future, we will predict contact maps in non-homologous proteins with novel methods and make some research on structure comparison.

Acknowledgements

This work is supported by the National Natural Science Foundation of China under Grant No. 60433020, the Key Science-Technology Project of the National Education Ministry of China under Grant No. 02090, and the Key Laboratory of Symbol Computation and Knowledge Engineering of the Ministry of Education.

References

1. Göbel, U., Sander, C., Scheider, R., Valencia, A.: Correlated Mutations and Residue Contacts in Proteins. Proteins, **18** (1994) 309-317
2. Singer, M.S., Vriend, G., Bywater, R.P.: Prediction of Protein Residue Contacts with a PDB-derived Likelihood Matrix. Protein Eng, **15** (2002) 721-725
3. Fariselli, P., Olmea, O., Valencia, A., Cassadio, R.: Prediction of Contact Maps with Neural Networks and Correlated Mutations. Protein Eng, **14** (2001) 835-843
4. Vendruscolo, M., Kussell, E., Domany, E.: Recover of Protein Structure from Contact Maps. Structure Fold. Des., **2** (1997) 295-306
5. Mirny, L., Domany, E.: Protein Fold Recognition and Dynamics in the Space of Contact Maps. Proteins, **26** (1996) 319-410
6. Hobohm, U., Scharf, M., Schneider, R., Sander, C.: Selection of Representative Data Sets. Prot.Sci, **1** (1992) 409-417
7. Kabsch, W., Sander, C.: Dictionary of Protein Secondary Structure: Pattern Recognition of Hydrogen-bonded and Geometrical Features. Biopolymers, **22** (1983) 2577-2637
8. Rose, G.D., Geselowitz, A. R., Lesser, G.J., Lee, R.H., Zehfus, M.H.: Hydrophobicity of Amino Acid Residue in Globular Protein. Science*,* **229** (1985) 834-838

A SVR-Based Multiple Modeling Algorithm for Antibiotic Fermentation Process Using FCM

Yaofeng Xue[1] and Jingqi Yuan[1,2]

[1] Department of Automation, Shanghai Jiao Tong University, Shanghai 200030, China
xueyf@sjtu.edu.cn
[2] State Key Laboratory of Bioreactor Engineering,
East China University of Science and Technology, Shanghai 200237, China

Abstract. A multiple modeling algorithm for antibiotic fermentation process based on fuzzy c-means (FCM) and support vector regression (SVR) is proposed. By analyzing the features of antibiotic fermentation, the mechanism of multiple modeling of the bioprocess is presented. Using FCM clustering method, the bioprocess is classified into several work states and sub-models. Then, taking advantage of the generalization properties of SVR, the multiple model of bioprocess is established and the proposed algorithm is described. Experimental data of industrial penicillin production is used to validate the model.

1 Introduction

Antibiotic fermentation is a time-variant and nonlinear process. It is difficult to model the process accurately and directly because of complex and not well understood metabolic pathways. To analyze, predict and control the bioprocess, the empirical model can be built by regression methods using the on-line and off-line measurement data. Studies on the neural networks have been carried out to model and control bioprocesses [1], [2]. However, to the widely existed fluctuation in antibiotic fermentation, these modeling methods have poor performance.

According to the features of antibiotic fermentation and the statistical analysis of the process records, the bioprocess is divided into several work states and sub-models using FCM clustering method in this paper. Correspondingly, the fuzzy memberships of input variables with regard to sub-models are obtained. Then, based on the powerful generalization capability of SVR, the multiple model of antibiotic fermentation is established. With this model, the product concentration and the total product of penicillin fed-batch cultivation are predicted. The root mean square error (RMSE) of the prediction is found less than 5%.

2 Multiple Modeling for Antibiotic Fermentation Process

The process performance of antibiotic fermentation fluctuates in industrial plants. This may result from many factors, such as the difference in quality of raw materials, feeding rate of substrates and other culture conditions. With the modern measurement technology, lots of process variables of antibiotic fermentation can be obtained. By analyzing the industrial data statistically with empirical knowledge, the bioprocess can be classified into several work states S_i ($i=1,2,\cdots,c$) and sub-models M_i ($i=1,2,\cdots,c$) accordingly. Suppose a bioprocess sampling data set $X=\{x_1,x_2,\cdots,x_n\}\subseteq R^n$ and $x_k\in X$ ($k=1,2,\cdots,n$). The fuzzy membership d_{ik} ($i=1,2,\cdots,c$, $k=1,2,\cdots,n$) of x_k with regard to sub-model M_i is obtained by using FCM clustering method. Training with data in the database of every work state, the SVR-based sub-models are built. The output value z_{ik} of sub-model M_i about x_k can be achieved as:

$$z_{ik}=\psi_i(x_k),\ \ x_k\in X,\ \ i=1,2,\cdots,c \tag{1}$$

where $\psi_i(\bullet)$ is the estimation function of input-output relationship of sub-model M_i.

The output variable Y of the multiple model of bioprocess can be calculated as:

$$Y=\sum_{k=1}^{n}\sum_{i=1}^{c}d_{ik}z_{ik}=\sum_{k=1}^{n}\sum_{i=1}^{c}d_{ik}\psi_i(x_k),\ \ x_k\in X \tag{2}$$

Therefore, the multiple model for antibiotic fermentation process is built and Fig. 1 described the mechanism.

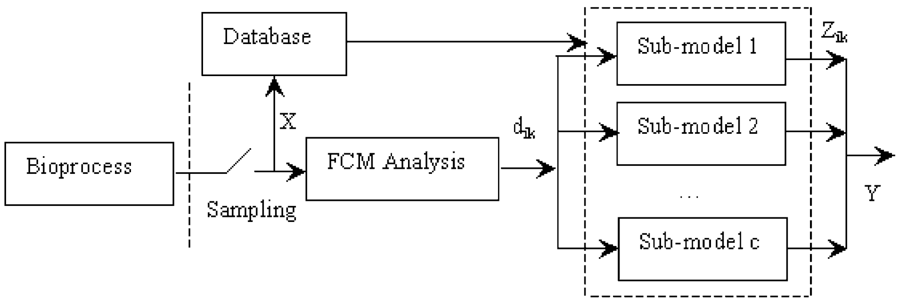


Fig. 1 Mechanism of multiple modeling for fermentation process

2.1 FCM Clustering

Fuzzy clustering method is a powerful data analysis tool, which can partition a data set into several fuzzy clusters. Among the fuzzy clustering methods, the fuzzy c-means (FCM) approach is widely applied in many fields [3], [4].

Suppose a data set $X=\{x_1,x_2,\cdots,x_n\}\subseteq R^n$ and X can be partitioned into c ($2\le c\le n$) clusters by FCM method. Using the fuzzy membership functions of X with regard to i th ($i=1,2,\cdots,c$) cluster, X can be transformed into a fuzzy data set S . $V=\{v_1,v_2,\cdots,v_c\}\subseteq R^n$ is the clustering center matrix. The FCM membership matrix $D=(d_{ik})_{c\times n}$ and d_{ik} ($i=1,2,\cdots,c$, $k=1,2,\cdots,n$) denotes the fuzzy membership of $s_k\in S$ ($k=1,2,\cdots,n$) subjected to i th ($i=1,2,\cdots,c$) cluster. Moreover, d_{ik} is constrained by $d_{ik}\in[0,1]$, $\sum_{i=1}^{c}d_{ik}=1$ and $0<\sum_{k=1}^{n}d_{ik}<n$. The objective function of

FCM clustering method is described as:

$$J(D,V)=\sum_{i=1}^{c}\sum_{k=1}^{n}(d_{ik})^{p}\|v_i-s_k\|^2 \tag{3}$$

where $\|v_i-s_k\|$ is the distance between v_i and s_k, $p\in R$ and $p\geq 1$. The FCM method uses the iterative approach to minimize the value of $J(D,V)$.

2.2 Modeling Using SVR

SVR is a statistical learning method based on support vector machine [5], [6], [7]. This learning algorithm follows the structure risk minimization (SRM) principle and its generalization error is minimized. SVR has a greater potential to generalize the input-output relationship by training data. In the following, a detailed description of SVR is represented.

Consider a training data set $\mathrm{T}=\{(x_1,y_1),\cdots,(x_n,y_n)\}$. x_k ($x_k\in\mathrm{R}^{\mathrm{n}}$, $k=1,2,\cdots,n$) and y_k ($y_k\in\mathrm{R}$, $k=1,2,\cdots,n$) are the input and output variables, respectively. The nonlinear estimation function is given as following:

$$\psi(x)\ =<w,\ \Phi(x)>+b \tag{4}$$

where $w\in\mathrm{R}^{\mathrm{n}}$ is the weight vector, $b\in\mathrm{R}$ is the bias, $\Phi(x)$ is the mapping function of feature space and $<w,\Phi(x)>$ is the dot product function of feature space.

Using the kernel functions, SVR can transform the nonlinear input-output relationship in lower-dimensional space into a linear relationship in the high-dimensional feature space. After the proper ε, C_s and kernel function are selected, the optimal problem of SVR with ε-Insensitive loss function (ε-SVR) is presented as:

$$\begin{aligned}&\min_{\alpha,\alpha^*\in R^{2l}}\ \frac{1}{2}\sum_{k,j=1}^{n}(\alpha_k^*-\alpha_k)(\alpha_j^*-\alpha_j)K(x,x')+\varepsilon\sum_{k=1}^{n}(\alpha_k^*+\alpha_k)-\sum_{k=1}^{n}y_k(\alpha_k^*-\alpha_k)\\ &s.t.\ \ \sum_{k=1}^{n}(\alpha_k-\alpha_k^*)=0,\ 0\leq\alpha_k,\ \alpha_k^*\leq\frac{C_s}{n},\ \mathrm{k}=1,2,\cdots,n\end{aligned} \tag{5}$$

The optimal solution of ε-SVR is $\bar{\alpha}=(\bar{\alpha}_1,\bar{\alpha}_1^*,\cdots,\bar{\alpha}_n,\bar{\alpha}_n^*)^T$ and the nonlinear estimation function is achieved as:

$$\psi(x)=\sum_{k=1}^{n}(\bar{\alpha}_k^*-\bar{\alpha}_k)K(x_k,x)+\bar{b} \tag{6}$$

where $\bar{b}$ is determined by $\bar{\alpha}_j$ or $\bar{\alpha}_j^*$ in the open interval $(0,\frac{C_s}{n})$.

2.3 FCM and SVR Based Multiple Modeling Algorithm

Based on the principles of FCM and SVR, the multiple modeling algorithm for antibiotic fermentation process is represented as the followings:

Step 1: Suppose a training data set $T=\{(x_1,y_1),\cdots,(x_n,y_n)\}$ in the database of fermentation process. With the statistical analysis and empirical knowledge, T is partitioned into subset T_i ($i=1,2,\cdots,c$; $2\leq c\leq n$) according to different work states.

Step 2: Select proper ε ($\varepsilon \in R$, $\varepsilon \geq 0$), C_s ($C_s \in R$, $C_s \geq 0$) and kernel function $K(x, x')$ of SVR.

Step 3: After training T_i, the optimal solution $\bar{\alpha} = (\bar{\alpha}_1, \bar{\alpha}_1^*, \cdots, \bar{\alpha}_n, \bar{\alpha}_n^*)^T$ of SVR-based sub-model M_i ($i = 1, 2, \cdots, c$) is achieved. The nonlinear regression function of sub-model M_i is established as Eq. (6).

Step 4: Consider a current sampling data set $X = \{x_1, x_2, \cdots, x_n\} \subseteq R^n$ of fermentation process. Using fuzzy membership functions of x_k ($x_k \in X$, $k = 1, \cdots, n$) with regard to i th ($i = 1, 2, \cdots, c$) cluster, X can be transformed into a fuzzy matrix S ($s_k \in S$, $0 \leq s_k \leq 1$).

Step 5: Construct the initial FCM membership matrix $D^{(0)}$.

Step 6: Calculate the clustering center matrix $V^{(l)}$:

$$v_i^{(l)} = \sum_{k=1}^{n} (d_{ik}^{(l)})^p s_k / \sum_{k=1}^{n} (d_{ik}^{(l)})^p \tag{7}$$

where l ($l = 0, 1, 2, \cdots$) is the iteration time and $i = 1, 2, \cdots, c$.

Step 7: Compute the next FCM membership matrix $D^{(l+1)}$:

$$d_{ik}^{(l+1)} = [\sum_{k=1}^{c} (\| s_k - v_i \| / \| s_k - v_j \|)^{\frac{1}{p-1}}]^{-1} \tag{8}$$

Step 8: Choose $\delta > 0$. If Eq. (9) is satisfied as:

$$\| D^{(l+1)} - D^{(l)} \| \leq \delta \tag{9}$$

then stop the iterative computation, $D^{(l+1)}$ and $V^{(l+1)}$ are the optimal matrixes and go to step 9. Otherwise, let $l = l + 1$ and go to step 6.

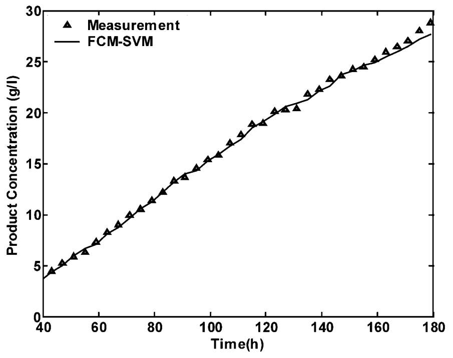
Step 9: Input the regression function of sub-model M_i ($i = 1, 2, \cdots, c$) with x_k ($x_k \in X$, $k = 1, \cdots, n$), the output z_{ik} of M_i is obtained as Eq. (1).

Step 10: Calculate Eq. (2) and achieve the output Y of multiple model corresponding to input X .

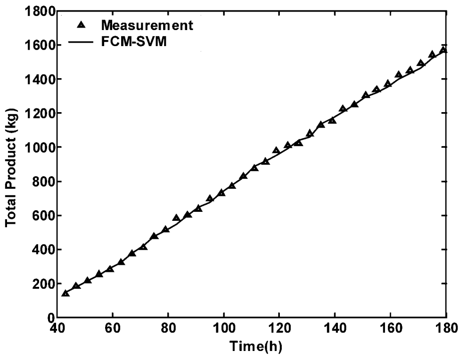
3 Simulations

In the following simulations, the penicillin production data in a Chinese pharmaceutical factory is used to verify the feasibility of the proposed algorithm.

The algorithm is used to predict the product concentration and the total product of penicillin fed-batch cultivation 8 h ahead. The sampling time interval of penicillin fermentation process is 4 h. The penicillin concentration, the total product of penicillin, the total reducible sugar consumption and the total precursor consumption are composed of the input variables of the multiple model. According to the statistical analysis of historical batch data, the bioprocess is divided into 3 sub-models and trained with 800 input-output data pairs. Then another 560 data are selected as the testing data set. Fig. 2 (a) and (b) compare the measured values (black triangle) and the predicted values (black line) using FCM and SVR based modeling algorithm of the product concentration and the total product of a penicillin fermentation batch, respectively.



(a) Product concentration



(b) Total product

Fig. 2. Comparison between measured and predicted values using the algorithm for a batch.

The simulations results are listed in table 1. The simulation results show that the proposed algorithm is feasible and accurate for predictions of penicillin fed-batch cultivation process.

Table 1. RMSE of prediction values using the proposed algorithm (%) .

Batch No.	1	2	3	4	5	6	7	Mean
RMSE of the total product	3.33	3.15	3.73	2.86	2.05	5.11	2.51	3.25
RMSE of the product concentration	4.50	3.40	4.30	2.61	3.16	4.4	2.62	3.57

4 Conclusions

In this paper, a multiple modeling algorithm using FCM and SVR has been introduced. By the statistical analysis of the industrial data of fed-batch antibiotic fermentation, the bioprocess is divided into several work states and sub-models accordingly. Then, the multiple modeling mechanism of bioprocess is presented. By integrating the advantages of FCM and SVR, a multiple modeling algorithm is proposed. With the industrial penicillin production data, the product concentration and the total product of penicillin fed-batch cultivation are predicted using the proposed algorithm. The simulation results show that the feasibility and validity of the proposed multiple modeling algorithm.

Acknowledgements

This work was supported by Nature Science Foundation of China (No. 60174024) and Alexander von Humboldt-Stiftung.

References

1. Xiong, Z., Zhang, J.: Modeling and Optimal Control of Fed-Batch Processes Using Control Affine Feedforward Neural Networks. Proceedings of the American Control Conference, **6** (2002) 5025-5030
2. Hodge, D., Simon, L., Karim, M.N.: Data Driven Approaches to Modeling and Analysis of Bioprocesses: Some Industrial Examples. Proceedings of the American Control Conference, **3** (2003) 2062-2076
3. Bezdek, J.C.: Pattern Recognition with Fuzzy objective function algorithm. Plenum Press, New work (1981)
4. Cannon, R.L., Dave, J., Bezdek, J.C.: Efficient implementation of the fuzzy c means clustering algorithms. IEEE Trans. Pattern Anal. Machine Intell., **8** (1986) 248-255
5. Vapnik, V.: Statistical Learning Theory. Wiley, New York (1998)
6. Vapnik, V.: The nature of statistical learning theory. Springer Verlag, New York (1995)
7. Burges, C.J.C.: A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery 2, **2** (1998) 121-167

Non-parametric Statistical Tests for Informative Gene Selection*

Jinwen Ma, Fuhai Li, and Jianfeng Liu

Department of Information Science, School of Mathematical Sciences
and LMAM, Peking University, Beijing 100871, China
jwma@math.pku.edu.cn

Abstract. This paper presents two non-parametric statistical test methods, called Kolmogorov-Smirnov (KS) and U statistic test methods, respectively, for informative gene selection of a tumor from microarray data, with help of the theory of false discovery rate. To test the effectiveness of these non-parametric statistical test methods, we use the support vector machine (SVM) to construct a tumor diagnosis system (i.e., a binary classifier) based on the identified informative genes on the colon and leukemia data. It is shown by the experiments that the constructed tumor diagnosis system with both the KS and U statistic test methods can reach a good prediction accuracy on both the colon and leukemia data sets.

1 Introduction

With the rapid development of DNA microarray technology, we can now get rapid, large-scale screening for patterns of gene expression. These microarray data corresponding to certain biological feature are generally represented by a gene expression matrix $W = (w_{ij})_{n \times m}$. In the matrix, a row represents a gene, while a column represents a sample. The numerical value w_{ij} denotes the expression level of a specific gene in a particular sample. Many microarray data sets are now available on the web.

For medical diagnosis and treatment, it is important to select or discover informative genes of a tumor from microarray data. Essentially, the informative genes can not only provide valuable information for discovering the crucial reasons of the tumor as well as the treatment methods, but also support to construct an efficient tumor diagnosis system from their expression levels directly without any influence of the other irrelevant genes. In fact, these have been already many methods for informative gene selection. However, most of the existing methods are based on ranking the important genes according to a certain criterion which requires that the data follows a normal distribution (e.g., [1]-[4]). But the normality assumption is often violated in real data sets [5]. In order to avoid the normality assumption, a rank sum test method (as a typical non-parametric statistical method) has been suggested to select informative genes with a considerably improved performance of tumor diagnosis on the colon and leukemia data [5].

In this paper, we further study this problem via two other non-parametric statistical tests, called Kolmogorov-Smirnov (KS) test and U statistic test, respectively, for informative gene selection with help of the theory of false discovery rate [6]-[8]. To test the

* This work was supported by the Natural Science Foundation of China for Project 60471054.

effectiveness of these non-parametric statistical test methods, we use the support vector machine (SVM) to construct a tumor diagnosis system (i.e., a binary classifier) based on the identified informative genes on the colon and leukemia data. Our experiments show that the constructed tumor diagnosis system with both the KS and U test methods through SVM can reach a good prediction accuracy on both the colon and leukemia data sets.

2 The KS and U Statistic Tests

In this section, we introduce the KS and U statistic tests as the bases for informative gene selection. We consider a data set of two classes, i.e., $\mathcal{S} = \{x_{11}, \cdots, x_{1n}; x_{21}, \cdots, x_{2m}\}$ in which $x_{11}, \cdots, x_{1n}$ come from one population being subject to the probability distribution $F_1(x)$, while $x_{21}, \cdots, x_{2m}$ come from another population being subject to the probability distribution $F_2(x)$. We need to make a non-parametric statistical test of hypothesis $H_0 : F_1(x) = F_2(x)$ without any information on these two probability distributions.

2.1 The KS Test

The KS test is a typical non-parametric statistical test based on the ranks of the observations. Actually, we first rank all the observations in $\mathcal{S}$ in an ascending order. Then, each observation has a ranking number, being called its rank in statistics. We now define a discrete probability distribution $F = \{f_1, f_2, \cdots, f_{n+m}\}$ according to the ranks $\{k_1, k_2, \cdots, k_n\}$ of the observations $\{x_{11}, \cdots, x_{1n}\}$ of the first class as follows (assuming that $k_1 \leq k_2 \leq \cdots \leq k_n$).

$$f_j = \begin{cases} 0, \text{ if } j < k_1; \\ \frac{i}{n}, \text{ if } j = k_i, i = 1, 2, \cdots, n; \\ \frac{i}{n}, \text{ if } k_i < j < k_{i+1}, i = 1, 2, \cdots, n-1; \\ 1, \text{ if } j > k_n, \end{cases} \tag{1}$$

for $j = 1, 2, \cdots, n + m$. In the same way, we can define $G = \{g_1, g_2, \cdots, g_{n+m}\}$ according to the ranks of the observations $\{x_{21}, \cdots, x_{2n}\}$ of the second class. Finally, we construct the KS statistic D_{nm} as follows.

$$D_{nm} = max\{|f_i - g_i| : i = 1, 2, \cdots, n + m.\}. \tag{2}$$

For a significance level $\alpha > 0$, we can get the threshold value $V(\alpha)$ of D_{nm} from a general KS test table. If $D_{nm} > V(\alpha)$, we reject H_0; otherwise, we accept H_0.

2.2 The U Statistics Test

Supposing that $n \leq m$ and R_1 is the sum of ranks of the observations of the first class, i.e., $R_1 = \sum_{i=1}^{n} k_i$, we have the following (Mann-Whitney) U statistic:

$$U = nm + \frac{n(n+1)}{2} - R_1. \tag{3}$$

On the other hand, if $n > m$, we let R_1 be the sum of ranks of the observations of the second class and the U statistic is defined by

$$U = nm + \frac{m(m+1)}{2} - R_1. \tag{4}$$

When H_0 holds, the U statistic tends to be subject to a normal distribution with the mean $\mu_U = nm/2$ and the standard variance $\sigma_U = \sqrt{nm(n+m+1)/12}$ as the number of observations in each class becomes large. That is,

$$Z = \frac{U - \mu_U}{\sigma_U} = \frac{U - nm/2}{\sqrt{nm(n+m+1)/12}} \sim N(0,1). \tag{5}$$

Thus, we can make the statistical test on H_0 from Z. For a significance level $\alpha > 0$, we can get the threshold value $V(\alpha)$ from the standard normal distribution function. In the same way, if $Z > V(\alpha)$, we reject H_0; otherwise, we accept H_0.

3 Informative Gene Selection and Tumor Diagnosis System via SVM

We now consider the informative gene selection based on these two non-parametric statistical tests. For informative gene selection, we can make a statistical test on each gene with its expressions on the two classes of samples(tumor and normal tissues or two kinds of tumor tissues). Clearly, if a gene is informative to the tumor, the probability distributions on the two classes should be quite different; otherwise they should be the same. On the other hand, we generally know nothing about the structures of these distributions. In these situations, it is reasonable to apply the KS or U statistic test to the informative gene selection via a microarray data set. That is, on each gene, when H_0 is rejected by the test of hypothesis, we consider this gene is informative; otherwise, we consider it is not informative.

Generally, there are thousands of genes in a microarray data and thus the informative gene selection is a large multiple-hypothesis testing problem. In this case, we must control the false discovery rate (FDR), i.e., the ratio of the number of falsely discovered (or selected) informative genes over that of all discovered informative genes [6]-[7]. Only when the FDR is controlled in a certain degree, we are sure that the informative gene selection is reliable. In order to do so, Storey and Tibshirani [8] proposed a q-value method which can be used to select the informative genes with the FDR being controlled directly. Actually, we first make the KS or U statistic test for each gene from the microarray data independently. Then, we can calculate the p-values of these statistical tests according to the statistics, say $p_1 \leq p_2 \leq \cdots, \leq p_n$ in an ascending order. By the q-value estimation algorithm given in [8] (which is now available on the web site: http://faculty.washington.edu/ jstorey/qvalue.), we can get their corresponding q-values, say $q_1 \leq q_2 \leq \cdots, \leq q_n$. Finally, if we want to control the FDR by $\alpha > 0$, we need only to select the informative genes by the selection criterion that $q_i \leq \alpha$. We will use this q-value method via the KS and U statistic tests to select the informative genes from a microarray data set.

To test the effectiveness of our non-parametric statistical test method for informative gene selection, we build a tumor diagnosis system (i.e., a binary classifier) using the support vector machine (SVM) [9]. Actually, SVM has been proved to be the most effective machine learning algorithm for processing large scale gene expression profiles. It has been derived from the optimal classification problem in the sample space with a finite number of samples. There are many softwares of SVM available on the web and we will use the software of SVM in Matlab. For comparison, we also try the following 3 kinds of support vector machines: (1). Radial basis function SVM (RBF kernel); (2). 3-poly SVM (cubic polynomial kernel); and (3). Linear SVM (no kernel).

4 Experiment Results

We test the effectiveness of our non-parametric statistical test methods for informative gene selection through SVM for tumor diagnosis using two real data sets as follows.

The colon cancer data set. It contains the expression profiles of $2,000$ genes in 22 normal tissues and 40 colon tumor tissues (retrieved from http://micro-array.princeton.edu /oncology/database.html). In our experiment, we use the train set (22 normal and 22 tumorous tissues) and test set (18 tumorous tissues) provided at the web site.

The leukemia cancer data set. It consists of $7,129$ genes in 47 acute lymphoblastic leukemia (ALL) and 25 acute myeloid leukemia (AML) samples (retrieved from http://www-genome.wi.mit.edu/cgi-bin/cancer/datasets.cgi). In our experiments, we use the train set (27 ALL, 11 AML) and the test set (20 ALL, 14 AML) provided at the web site.

We use MATLAB toolbox OSU_SVM 3.0 (which can be obtained from the web site: `http://www.ece.osu.edu/~maj/osu_svm/`.) to implement the three kinds of SVMs. In the radial basis function and 3-poly SVMs, there are two parameters γ and C. In our experiments, we generally select $\gamma = 0.02$ and $C = 0.05$ on the colon dataset, and $\gamma = 0.002$ and $C = 10$ on the leukemia dataset. Sometimes, they are slightly adjusted to get the best performance of the SVMs. For the KS and U statistic test methods, we try three FDR α: 0.03, 0.05, and 0.07, respectively. The informative gene selection returns different numbers of informative genes on both the colon and leukemia data sets with slightly different prediction accuracies on the test sets. The results of the two non-parametric statistical test methods on the colon and leukemia data sets are given in Table 1-4, respectively.

From Tables 1& 2, we find that on the colon data set, the SVM tumor diagnosis system can reach an optimum prediction accuracy 1 by both the KS and U statistic test

Table 1. The result of the KS test method on the colon data set. Here and in the following tables, each number in the second to the fourth rows represents the prediction accuracy of the SVM on the test set.

α (# informative genes)	0.03 (130)	0.05 (187)	0.07 (216)
RBF SVM	0.9444	1.0000	1.0000
3-poly SVM	0.9444	0.9444	0.9444
Linear SVM	0.9444	0.9444	0.9444

Table 2. The result of the U test method on the colon data set.

α (# informative genes)	0.03 (130)	0.05 (187)	0.07 (216)
RBF SVM	0.9444	1.0000	1.0000
3-poly SVM	0.9444	0.9444	0.9444
Linear SVM	0.9444	0.9444	0.9444

Table 3. The result of the KS test method on the leukemia data set.

α (# informative genes)	0.03 (775)	0.05 (946)	0.07 (1108)
RBF SVM	0.9706	0.9706	0.9706
3-poly SVM	0.9706	0.9706	0.9706
Linear SVM	0.9706	0.9706	0.9706

methods. From Tables 3& 4, we further find that on the leukemia data set, the prediction accuracy of the SVM tumor diagnosis system by using the informative genes of the KS test method is always 0.9706, where only one prediction error happens in our test experiments. As for the U statistic test method, the optimum prediction accuracy of the SVM tumor diagnosis system is even 1. Therefore, the SVM tumor diagnosis system with both the KS and U statistic test methods for informative gene selection can reach a good prediction accuracy on both the colon and leukemia data sets.

We also make the experiments on these two microarray data sets by the rank sum test method proposed in [5] with help of false discovery rate theory. It is found by the experiments that the rank sum method is as good as the U statistic test method and better than the original one to select the informative genes directly by the test of hypothesis. It is also found by the experiments that these non-parametric statistical test methods considerably outperforms the typical ranking methods [1]-[4] through the same SVM software.

5 Conclusions

We have investigated the informative gene selection problem from a microarray data set via non-parametric statistical test with help of the theory of false discovery rate theory. We apply the Kolmogorov-Smirnov (KS) and U statistic tests and the q-value algorithm to the informative gene selection of a tumor and use the support vector machine (SVM) to construct a tumor diagnosis system with the identified informative genes on the colon and leukemia data. Our experiments show that the constructed tumor diagnosis system with both the KS and U statistic test methods can lead to a good prediction accuracy on both the colon and leukemia data sets.

Table 4. The result of the U statistic test method on the leukemia data set.

α (# informative genes)	0.03 (1042)	0.05 (1255)	0.07 (1416)
RBF SVM	0.9706	0.9706	0.9706
3-poly SVM	0.9706	0.9706	1.0000
Linear SVM	0.9706	0.9706	0.9706

References

1. Alon, U., Barkai, N., Notterman, D.A., et al.: Broad Patterns of Gene Expression Revealed by Clustering Analysis of Tumor and Normal Colon Tissues Probed by Oligonucleotide Arrays. it Proc Natl Acad Sci USA, **96** (1999) 6745-6750
2. Golub, T.R., Slonim, D.K. Tamayo, P., et al.: Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring. it Science, **286** (1999) 531-537
3. Furey, T.S., Cristianini, N., Duffy, N., Bednarski, D.W., Schummer, M., and Haussler, D.: Support Vector Machine Classification and Validation of Cancer Tissue Samples Using Microarray Expression Data. it Bioinformatics, **16** (2000) 906-914
4. Ben-Dor, A., Friedman, N., and Yakhini, Z.: Scoring Genes for Relevance. Agilent Technical Report, no. AGL-2000-13(2000)
5. Deng, L., Ma, J., and Pei, J.: Rank Sum Method for Related Gene Selection and Its Application to Tumor Diagnosis . Chinese Science Bulletin, **49** (2004) 1652-1657
6. Benjamini, Y., and Hochberg, Y.: Controlling the False Discovery Rate: a Practical and Powerful Approach to Multiple Testing . J. R. Statist. Soc. B, **57** (1995) 289-300
7. Storey, J.D.: A Direct Approach to False Discovery Rates. J. R. Statist. Soc. B, **64** (2002) 479-498
8. Storey, J.D., and Tibshirani, R.: Statistical Significance for Genomewide Studies. Proc Natl Acad Sci USA, **100** (2003) 9440-9445
9. Vapnik, V.: The Nature of Statistical Learning Theory. Springer,New York (1995)

An Information Criterion for Informative Gene Selection[⋆]

Fei Ge and Jinwen Ma[⋆⋆]

Department of Information Science
School of Mathematical Sciences and LMAM
Peking University, Beijing 100871, China

Abstract. It is important in bioinformatics research and applications to select or discover informative genes of a tumor from microarray data. However, most of the existing methods are based on models which assume that the gene expressions are normal distributed, which is often violated in practice. In this paper, we propose an information criterion for informative gene selection by ranking the genes with the Kullback-Leiber discrimination information of two probability distributions of the expression levels on the tumor and normal (or another type of tumor) samples. We use support vector machine (SVM) to construct the tumor diagnosis system using certain top informative genes. The experiments on two well-known data sets (colon data and leukemia data) show that the information criterion can make the tumor diagnosis system reach 94.4% and 100% correctness rate of diagnosis on these two datasets, respectively.

1 Introduction

With the development of DNA microarray technology, we can now quickly obtain large-scale gene expression profiles, i.e., the microarray data, which provide important and detailed evidences to health state of human tissues for disease analysis and diagnosis. Moreover, as gene studies are shifting from DNA sequencing to function analysis, the microarry data will play a more important role since they can help us to discover and understand the biological characteristics from a group of genes.

The microarray data can be represented by a matrix $\mathbf{A} = (a_{ij})_{N\times k}$, where the i-th row corresponds to gene i, the j-th column corresponds to sample j, and a_{ij} denotes the mRNA expression level of gene i in sample j. Generally, it is a large matrix with thousands of rows according to such a number of genes in a microarray chip. As for tumor diagnosis, each sample is labelled to be of a certain tumor or not and the tumor diagnosis system can be trained with the supervised learning on these data. However, the computing complexity due to the high dimension of the data has made it hard to train the learning system. Moreover, not all these genes are relevant to the tumor and the irrelevant genes will contribute nothing to the learning system but noise. In order to achieve a high diagnosis accuracy, we should first select the informative genes that are discriminative among the tumor and normal phenotypes. Meanwhile, the informative genes provide clues to medical or biological studies.

[⋆] This work was supported by the Natural Science Foundation of China for Project 60471054.
[⋆⋆] The corresponding author, Email:jwma@math.pku.edu.cn.

The problem of informative gene selection has been studied extensively in the last five years. Golub et al.[1] proposed a kind of discrimination measurement on the genes via a simple statistic: $(\mu_1 - \mu_2)/(\sigma_1 + \sigma_2)$, similar to the t-statistic from expression levels in two different classes. The t-statistic method and its variations are the most popular in gene selection[2][3][4]. The statistic value is considered as a score for each gene. Then all the genes are ranked according to their scores, and the group of top genes are candidates for informative genes. The most serious problem for t-statistic method is that it assumes the expression levels of each gene follow a normal distribution. However, this is not always true in practice[5].

Many other scores have been proposed, such as NToM score[6] and BSS/WSS score[7]. They even don't consider the probability distributions of two-class gene expressions. According to the dimension reduction or filtering theory, some other methods are also proposed for informative gene selection, e.g., [8] [9] [10]. However, these methods are not only lack of theoretic foundation on informative gene selection, but also difficult to deal with, since the dimension of the data, i.e., the number of genes, is so large.

In this paper, we propose an information criterion for informative gene selection by measuring the discriminate power of a gene with the Kullback-Leiber discrimination information between the probability distributions of the expression level on the tumor and normal (or another type of tumor) samples, which doesn't need the normality assumption on the expression levels. We then construct a tumor diagnosis system by the support vector machine trained on the data set of certain top informative genes. In our experiments, the information criterion can make the tumor diagnosis system reach 94.4% correctness rate of diagnosis on colon dataset and 100% correctness rate of diagnosis on leukemia dataset, respectively.

In the sequel, we propose our information criterion for informative gene selection in Section 2. In Section 3, the experiments are conducted to demonstrate the information criterion, being compared with the t-statistic method. A brief conclusion is made in Section 4.

2 The Information Criterion

From the point of view of probability theory, the expression level of an informative gene should subject to different probability distributions on the tumor and normal tissues, respectively. Moreover, as this gene becomes more discriminative to the tumor, the two probability distributions should be more different. Otherwise, the expression level of an irrelevant gene should subject to the same probability distribution on both the tumor and normal tissues. That is, the two probability distributions become the same in this case. Based on this fact, we can use Kullback-Leiber discrimination information (also known as Kullback-Leiber divergence) of these two probability distributions to evaluate the discriminative power of the gene to the tumor.

If $p(x)$ and $q(x)$ are the probability distributions of the expression level on tumor and normal tissues, respectively, the Kullback-Leiber discrimination information is computed by $K(p||q) = \int p(x) \log(p(x)/q(x))dx$. Since there are only a finite number of data available, we can only use the empirical distributions instead of $p(x)$ and $q(x)$. For clarity, we rewrite the gene expression data by the following matrix:

$$\left(\begin{array}{ccc|ccc} a_{11} & \cdots & a_{1m} & b_{11} & \cdots & b_{1n} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{N1} & \cdots & a_{Nm} & b_{N1} & \cdots & b_{Nn} \end{array}\right), \tag{1}$$

where a_{ij} is the expression level of gene i for the j-th tumor sample, while each b_{ij} is the expression level of gene i for the j-th normal sample.

For a set of i.i.d. sample data $x_1, x_2, \cdots, x_N$, the empirical distribution can be estimated by

$$\hat{p}_N(x) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{h}K\left(\frac{x-x_i}{h}\right), \tag{2}$$

where $K(x)$ is called the *kernel function*, h is the *bandwidth* of Parzen window (related with N). In fact, Parzen[11] proved that under certain regular conditions, $\hat{p}_N(x)$ is an asymptotically unbiased estimator of the actual probability density function. In our experiments we use Gaussian kernel function $K(x) = \frac{1}{\sqrt{2\pi}}\mathrm{e}^{-x^2}$ and choose h by $\hat{h}_{NS} = (\frac{4}{3N})^{1/5}S$, where S denotes the standard deviation of the sample data.

According to Eq.(2), from the data $a_{i1}, a_{i2}, \cdots, a_{im}$, we can get $\hat{p}_i(x)$, the empirical distribution of the expression level of gene i on the tumor sample data, and from the data $b_{i1}, b_{i2}, \cdots, b_{in}$ we can get $\hat{q}_i(x)$, the empirical distribution for the expression of gene i on the normal sample data. For symmetry, we define

$$K_i = K(\hat{p}_i||\hat{q}_i) + K(\hat{q}_i||\hat{p}_i) = \int_{-\infty}^{+\infty}(\hat{p}_i(x) - \hat{q}_i(x))\log\frac{\hat{p}_i(x)}{\hat{q}_i(x)}dx \tag{3}$$

as the discriminative power of gene i to the tumor. Then, all genes can be ranked in the descending order of this criterion. We can select a certain number of genes ranked first as the informative genes and discard the rest ones.

After we select a group of informative genes, we can construct the tumor diagnosis system by a binary (or bipolar) supervised classifier trained with the expression levels of these informative genes. Since SVM owns a better generalization ability on a small sample set[12], we use it as our tumor diagnosis system, with radial basis function as the kernel function.

For comparison, we also give the t-statistic method for informative gene selection. Actually, the t-statistic method ranks genes in the descending order of the absolute value of t-statistic calculated from data samples in two classes as follows:

$$t_i = (\bar{a}_i - \bar{b}_i)/(\sqrt{\frac{s_{a,i}^2}{m} + \frac{s_{b,i}^2}{n}}), \tag{4}$$

where $\bar{a}_i$ and $s_{a,i}^2$ are the mean and variance of gene i's expression level on the tumor samples, respectively, while $\bar{b}_i$ and $s_{b,i}^2$ are the mean and variance of gene i's expression level on the normal samples, respectively. Traditionally, t-statistic is used to test whether two normal distributions have the same mean.

3 Experiment Results

We test the effectiveness of the information criterion for informative gene selection through the SVM for tumor diagnosis using the two real data sets as follows:

The colon cancer dataset[1] contains the expression profiles of 2000 genes in 22 normal tissues and 40 tumor tissues[2]. In our experiments, we randomly select 44 samples as the training set, and use the other 18 samples as test data.

The leukemia dataset[2] consists of expression profiles of 7129 genes from 47 acute lymphoblastic leukemia (ALL) and 25 acute myeloid leukemia (AML) samples[1]. Specifically, the training dataset contains 38 samples (27 ALL and 11 AML), while the test dataset contains 34 samples (20 ALL, 14 AML).

We calculate all K_i and then rank the genes with these values. As shown above, genes with larger K_i will have stronger discriminate powers. In the experiments, we select the top k genes with the highest ranks, and train the SVM with the expression levels of these k genes. We test the performance with k gradually increasing from some initial value k^0.

We use MATLAB toolbox OSU SVM 2.0 (which can be obtained from `http://eewww.eng.ohio-state.edu/~maj/osu_svm/`) to implement the SVM with the RBF kernel functions. In this situation, there are only two parameters γ and C to be determined. Actually, the selection of γ and C affects the performance of the SVM. In the experiments, we take a grid search procedure from 16×16 pairs of γ and C, and choose the optimal values by cross validation. Then, the SVM is trained for the tumor diagnosis. The correctness rates of tumor prediction on the two datasets with k from 1 to 300 are sketched in Figs 1 & 2, respectively.

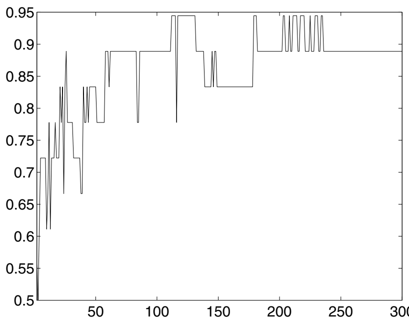


Fig. 1. The correctness rate of tumor prediction on colon cancer data

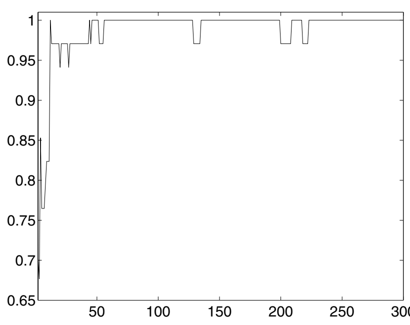


Fig. 2. The correctness rate of tumor prediection on leukemia data

From the curves in Figs 1 & 2, we can observe that as the number k of selected informative genes increases from the beginning, the prediction accuracy tends to increase, too. When the highest correctness rate of tumor prediction is reached, there exists certain interval in which each k can maintain a good prediction accuracy, although the correctness rate may fluctuate slightly. But if k further increases, the correctness rate begins to decrease. This means that the information criterion is significant on the selection of informative genes of a tumor. It can be also found that when the number of informative genes is properly selected, the information criterion can make the tumor

[1] retrieved from `http://microarray.princeton.edu/oncology/database.html`

[2] retrieved from `http://www-genome.wi.mit.edu/cgi-bin/cancer/datasets.cgi`

diagnosis system reach 94.4% correctness rate of diagnosis on the colon dataset and 100% correctness rate of diagnosis on the leukemia dataset, respectively.

On the other hand, the experiment results show that the number of informative genes should be carefully selected and is essential to the performance of SVM for tumor prediction. With the more or less genes selected, the SVM would result in a drop of prediction accuracy. However, there exist a number of weakly related genes that are not very sensitive to the performance of the SVM for tumor prediction. It can be found by the experiments that the optimal number of informative genes of colon cancer is about 125, while that of leukemia is about 75.

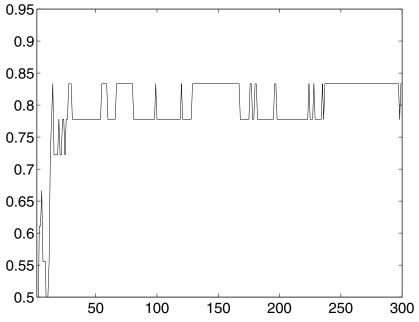


Fig. 3. The correctness rate of tumor prediction on colon cancer data

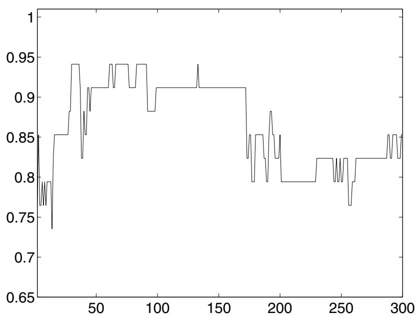


Fig. 4. The correctness rate of tumor prediection on leukemia data

For comparison, we replace the gene selection criterion with the t-statistic and the results on the two datasets are shown in Figs 3 & 4. Clearly, our information criterion is superior to the t-statistic method on the tumor diagnosis.

4 Conclusions

We have proposed an information criterion for informative gene selection of a tumor according to the Kullback-Leiber discriminative information between the two probability distributions of the expression levels on the tumor and normal tissues. By experiments on real data sets through the SVM for tumor diagnosis, we show that the information criterion is significant and even better than the t-statistic method. Moreover, the experiments also show that the information criterion can make the tumor diagnosis system reach 94.4% correctness rate of diagnosis on colon dataset and 100% correctness rate of diagnosis on leukemia dataset, respectively.

References

1. Golub, T.R., Slonim, D.K., Tamayo, P., et al.: Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring. Science, **286** (1999) 531-537

2. Alon, U., Barkai, N., Notterman, D.A., et al.: Broad Patterns of Gene Expression Revealed by Clustering Analysis of Tumor and Normal Colon Tissues Probed by Oligonucleotide Arrays. Proceedings of the National Academy of Sciences, **96** (1999) 6745–6750
3. Furey, T.S., Cristianini, N., Duffy, N., et al.: Support Vector Machine Classification and Validation of Cancer Tissue Samples Using Microarray Expression Data. Bioinformatics, **16** (2000) 906-914
4. Nguyen, D., and Rocke, D.: Tumor Classification by Partial Least Squares Using Microarray Gene Expression Data. Bioinformatics, **18** (2002) 39-C50.
5. Deng, L., Ma, J., and Pei, J.: Rank Sum Method for Related Gene Selection and Its Application to Tumor Diagnosis. Chinese Science Bulletin, **15** (2004)1652-1657
6. Ben-Dor, A., Friedman, N., and Yakhini, Z.: Scoring Genes for Relevance. Agilent Technical Report, no. AGL-2000-13 (2000)
7. Dudoit, S., Fridlyand, J., Speed, T.: Comparison of Discrimination Methods for The Classification of Tumor Using Gene Expression Data. Journal of American Statistical Association, **97** (2002)77–87
8. Liu, J., Iba, H., and Ishizuka, M.: Selecting Informative Genes with Parallel Genetic Algorithms in Tissue Classification. Genome Informatics, **12** (2001) 14-23
9. Xiong, M., Li, W., Zhao, J., et al.: Feature (Gene) Selection in Gene Expression-based Tumor Classification. Mol Genet Metab, **73** (2001) 239-47
10. Hellem Bø , T., and Jonassen, I.: New Feature Subset Selection Procedures for Classification of Expression Profiles. Genome Biology, 3(4): research0017.1-C0017.11, (2002)
11. Parzen, E.: On the Estimation of a Probability Density Function and Mode. Annals of Mathematical Statistics, **33** (1962)1064–1076
12. Vapnik, V.: The Nature of Statistical Learning Theory. Springer,New York (1995)

OPTOC-Based Clustering Analysis of Gene Expression Profiles in Spectral Space

Shuanhu Wu[1], Alan Wee Chung Liew[2], and Hong Yan[3]

[1] School of Computer Science & Technology Yantai University,
Yantai , Shandong 264005, China
wushuanhu@163.com

[2] Department of Computer Science and Engineering
Chinese University of Hong Kong, Hong Kong, China
wcliew@cse.cuhk.edu.hk

[3] Department of Computer Engineering and Information Technology
City University of Hong Kong, Tat Chee Avenue, Kowloon, Hong Kong, China
h.yan@cityu.edu.hk

Abstract. In this paper, a new feature extracting method and clustering scheme in spectral space for gene expression data was proposed. We model each member of same cluster as the sum of cluster's representative term and experimental artifacts term. More compact clusters and hence better clustering results can be obtained through extracting essential features or reducing experimental artifacts. In term of the periodicity of gene expression profile data, features extracting is performed in DCT domain by soft-thresholding de-noising method. Clustering process is based on OPTOC competitive learning strategy. The results for clustering real gene expression profiles show that our method is better than directly clustering in the original space.

1 Introduction

The gene expression profiles obtained from cDNA microarrays can be used for analysis of complex processes involving biological development [1], hormonal and cellular oscillations [2], enzymatic pathways [3] and genetic regulatory pathways [4]. Gene expression profile analysis requires the extraction of meaningful patterns inherent in the data in order to discover functional gene classes. Currently the most popular approach to gene expression data analysis is data clustering, since genes of similar function tend to display similar expression patterns.

Gene expression data have been studied by many researchers based on the self-organizing maps [5], k-nearest neighbor classifiers [6], hierarchical clustering [7], binary transition networks [8], naïve Bayes modeling [9], cluster affinity search technique [10], super-paramagnetic clustering [11], simulated annealing [12], et al. These algorithms all process gene expression profile data directly. Since gene expression data often contain heavy noise, direct analysis can produce misleading results.

In this paper, we study how to extract reliable features of temporal gene expression profiles and use them to obtain better clustering results. Since biological systems can simultaneously cycle at different frequencies [13], we chose the discrete cosine transformation (DCT) [14] to model gene expression profiles and extract its essential features. Our method provides a frequency spectral representation of the gene expression signal. In our method, the data are first cleansed using a de-noising technique [15], which results in more compact clusters. Then, we cluster the data in the DCT domain based on the Self-Splitting and Merging Competitive Learning (SSMCL) algorithm [18].

2 Spectral Feature Extraction

Clustering analysis has been proven to be a useful tool in discovering pattern classes in gene expression data because similar gene expression profiles can be an indicator that the genes participate in the same or related cellular processes. Let X be the data set in an N-dimensional space, i.e. $X = \{x_1,\ldots x_N\}$, where each data vector (time series) $x_k = \{x_{k1},\ldots,x_{kN}\}$ is a point in an N-dimensional space. Assume that dataset X is grouped into M clusters, $C_1,\ldots,C_M$, and the distribution of each cluster satisfies the Gaussian distribution. We can represent each member in the same cluster C_i with the following model:

$$x_k = r_k + \sigma z_k . \tag{1}$$

where x_k is a member of cluster C_i, r_k is the corresponding noiseless data, z_k is a white noise component, which is independently and identically distributed (i.i.d) and is denoted by $z_k \sim N(0,1)$, and σ is the noise standard deviation. We assume that the variation in r_k is the genuine signal, i.e., the variation is due to the underlying biological process, and not due to noise. Let v_i denote the centroid of the C_i cluster, where v_i is the arithmetic mean of all members in C_i. Since the noise z_k is zero mean, the centroid of the noisy data x_k is also the centroid of the noise-free data r_k. Then the compactness of the cluster can be measured by the average within-class sum-of- square error ($AWSS$):

$$\begin{aligned} AWSS &= \frac{1}{T_i}\sum_{k=1}^{T_i}(x_k - v_i)^T(x_k - v_i) \\ &= \frac{1}{T_i}\sum_{k=1}^{T_i}(r_k + \sigma z_k - v_i)^T(x_k + \sigma z_k - v_i) \\ &= \frac{1}{T_i}\sum_{k=1}^{T_i}\left[(r_k - v_i)^T(r_k - v_i) + 2\sigma(r_k - v_i)^T z_k + \sigma^2 z_k{}^T z_k\right] \\ &= \frac{1}{T_i}\sum_{k=1}^{T_i}(r_k - v_i)^T(r_k - v_i) + \sigma^2 N \end{aligned} \tag{2}$$

where we make use of the property that the noise and the signal are statistically independent and uncorrelated. The first term in Equation (2) is the AWSS for the noise-free data. From Equation (2), it is obvious that if we can decrease the noise term, i.e., reduce the noise power σ^2, the AWSS can be decreased, making the cluster more compact.

2.1 Feature Decomposition by DCT

Gene expression profile data from a time series experiments show cellular oscillations [2], and have a definite periodicity. The periodic patterns have also been studied in dynamic modeling of gene expressions using SVD based techniques. Neal et al. [20] have shown that the first two singular vectors corresponding to two largest singular values are approximately sinusoidal. Atul et al. [13] have treated gene expression as a discrete time-invariant signal and compared the similarities of gene expressions using signal processing metrics. We select discrete cosine transform (DCT) here for feature extraction [14]. The DCT provides a spectral representation of signals. For slowly varying signal, the DCT tends to provide a reasonable approximation of the Karhunen-Loeve transform which is statistically optimal for data de-correlation. Since the DCT is model independent, it does not require the estimation of the covariance matrix.

Given a time series, $x_0, x_1, \ldots, x_{N-1}$, , in an N dimensional space, its discrete cosine transform (DCT) is defined as

$$y_k = \alpha(k)\sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi(2n+1)k}{2N}\right]. \qquad (3)$$
$$k = 0,1,\ldots N-1$$

Where

$$\alpha(k) = \begin{cases} \sqrt{1/N}, k = 0 \\ \sqrt{2/N}, k \neq 0 \end{cases}.$$

Based on Equation (3), each gene expression profile can be decomposed into the combination of cosine components of different frequencies. Since the basis functions in the DCT is orthonormal and linear, the Gaussian noise in the time domain is transformed to Gaussian noise in the frequency domain. Fig. 1(a) and (b) is an illustration of the behavior of a simulated signal in DCT domain. We can see from Fig.1(b) that the Gaussian noise is largely separated from the main signal in the frequency domain although it overlaps with the signal in the time domain that is very useful for extracting the main features of gene expression data.

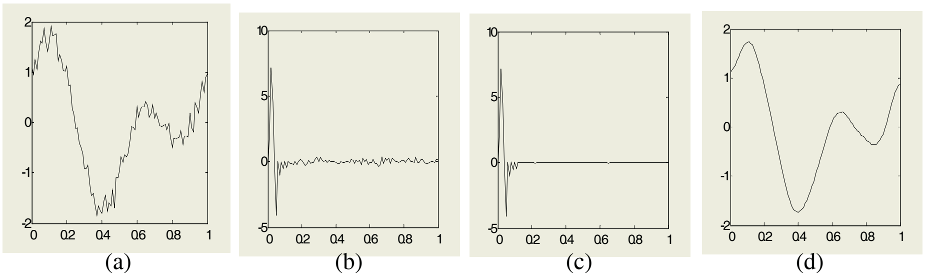


Fig. 1. Illustration of noise reduced based on soft-thresholding in the DCT domain. (a) The simulated signal, $y_i=\sin(4\pi t_i)+\cos(2\pi t_i)+0.15z_i$; $t_i=0,1,\ldots,100$; where z_i is Gaussian noise with $\sigma=0.15$. (b) The DCT coefficients of the simulated signal. (c) De-noised result in the frequency domain. (d) Restored signal in the time domain.

2.2 Feature Extraction and De-noising by Soft-Thresholding

By the DCT described above, we can see that the main signal and the noise term can be decomposed in different frequency components. If the noise can be reduced effectively without affecting the main signal, the main features of the signal will be enhanced naturally. We use a soft thresholding method for reducing noise. Soft thresholding [15], is a nonlinear operation and has been proven to be optimal for removing Gaussian noise. The soft-thresholding operator, T_δ, is defined as

$$T_\delta(y) = sign(y)(|y| - \delta)_+ = \begin{cases} y-\delta, & \text{for } y > \delta \\ 0, & \text{for } y \le \delta \\ y+\delta, & \text{for } y < -\delta \end{cases} \cdot \tag{4}$$

where δ is the threshold. For Gaussian noise with known noise level σ, we can set $\delta = 2\sigma$ for the DCT coefficients. For real gene expression profile data, since the level of the experimental error (noise) cannot be obtained a priori, we can estimate the noise level as a percentage of the total signal energy. Let $\{x_1,\dots x_N\}$ be a temporal gene expression profile, its DCT coefficients are $\{y_1,\dots,y_N\}$, then empirically the noise level can be estimated as follows:

$$\sigma = \sqrt{f_\sigma \sum_{i=1}^{N} y_i^2 / N} \cdot \tag{5}$$

where f_σ is weighting constant. In our experiment, we set f_σ at about 5%.

Since Gaussian noise has small amplitude in DCT domain compared with the main signal features and is distributed evenly in the DCT domain, we only perform soft-thresholding on the DCT coefficients that are without any significant main features. Our feature extraction process is defined as follows:

$$T_{f,\sigma}(y) = \begin{cases} y, & \text{if } |y| \ge 2.5\sigma \\ \text{sign}(y)(|y| - 2\sigma)_+, & \text{if } |y| < 2.5\sigma \end{cases} \cdot \tag{6}$$

Fig.1 shows the de-noising result for the simulated signal. Fig.1(b) is the spectrum of the simulated signal of Fig.1(a). Fig.1(c) is the de-noising result in the DCT domain using our method. Fig.1(d) is the restored signal. The example shows that our technique for noise reduction is very effective.

3 Clustering Algorithm Based on OPTOC Competitive Learning

Most conventional clustering algorithms have no guarantee that the clusters found do correspond to natural clusters in the dataset even if the correct number of clusters is given [6]. Self-Splitting Competitive Learning algorithm is a newly developed online learning algorithm based on the one-prototype-take-one-cluster (OPTOC) scheme [16]. In conventional clustering algorithm, if the number of prototypes is less than that of the number of natural clusters in the dataset, there must be at least one prototype that wins patterns from more than two clusters, and this behavior is called one-prototype-take-multiple-clusters (OPTMC). Fig.2(a) shows an example of learning based on the OPTMC paradigm, where P1 actually wins all three clusters and finally settles at the center of clusters S1, S2 and S3. In contrast, the one-prototype-take-one-cluster (OPTOC) idea allows one prototype to characterize only one natural cluster in the dataset, regardless of the number of clusters in the data. Fig.2(b) shows an example of learning based on the OPTOC paradigm, where P1 actually wins only the one of three clusters and finally settles at the center of cluster S3. The OPTOC based learning strategy has the following two main advantages: (1) It can find natural clusters, and (2) The final partition of the dataset is not sensitive to initialization.

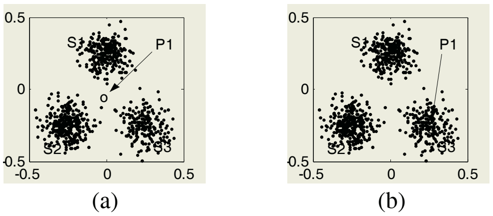


Fig. 2. Two learning methods: OPTMC versus OPTOC. (a) One prototype takes the center of three clusters (OPTMC). (b) One prototype takes one cluster (OPTOC) and ignores the other two clusters.

In [18], we proposed a new Self-Splitting and Merging Clustering algorithm (SSMCL) based on the OPTOC competitive learning paradigm. The OPTOC paradigm allows one prototype to characterize only one natural cluster in the dataset, regardless of the number of clusters in the data. The OPTOC behavior of a cluster prototype is achieved through the use of a dynamic neighborhood, which causes the prototype to eventually settle at the center of a natural cluster, while ignoring competitions from other clusters. In order to estimate the number of natural clusters in a dataset, we proposed an over-clustering and merging strategy. The over-clustering step minimizes the chance of missing any natural clusters in the data, while the merging step ensures that the final clusters are all visually distinct from each other.

If the number of clusters is known, the merging step can be omitted and the algorithm stopped when the required number of clusters is reached after the self-splitting step. For the experimental results described in the next section, we assumed that the correct number of clusters is known. This is to allow a fair comparison between clustering in the spectral domain and the original data domain.

Let $\{x_{i,j}|i=1,\ldots,M;\ j=1,\ldots,N\}$be the dataset to be processed in an N dimensional space. The procedure for our spectral feature based clustering algorithm is described as follows:

(a) Data normalization: Let $x_{i,j} = x_{i,j} - \sum_{k=1}^{N} x_{i,k} / N; i = 1,\ldots,M; j = 1,\ldots,N$;
(b) Perform the DCT for each normalized member in the dataset according to Equation (3);
(c) Estimate the noise level according to Equation (5);
(d) Extract the main features according to Equation (6);
(e) Perform clustering on the extracted features and label each member in the dataset based on the results;
(f) Group the dataset in the original space in terms of the clustering results in the feature space (the DCT domain) for visualization.

4 Experiments and Comparisons

In this section, we verify our clustering scheme by simulated and real gene expression profile data. We first use one simulated cluster with 100 samples generated from a representative signal to verify if our features extraction scheme can make the resulting cluster more compact. Then we verify if our spectral space clustering scheme can generate better clustering results for Yeast cell cycle data set [17] compare with the results obtained by directly clustering the same data set in the original data space.

4.1 Validating Cluster Compaction by De-noising

The simulated cluster was generated from $y_i=\sin(4\pi t_i+\theta_k)+\cos(2\pi t_i)+0.15z_i$; $t_i=0,1,\ldots,100$, $i=0,1,\ldots100$; where the phase shift θ_k was varied from -0.2 to 0.2, at a step of 0.004, to produce a total of 101 samples for the cluster. The phase shift models the genuine variations in the signals. To these samples, we added random Gaussian noise with $\sigma=0.15$. These constitute our noisy signals. We then applied the de-noising operation given by Equation (6) to the noisy samples. Fig.3 shows graphically the compactness of the simulated cluster for (a) noiseless data, (b) noisy data, and (c) de-noised data. The AWSS for the noiseless, noisy and de-noised data were 0.6898, 2.9902 and 1.2585, respectively. We note that the theoretical value for the AWSS term due to noise is $\sigma^2 N=2.2725$. The simulation result agrees closely with the theoretical calculation, i.e., 2.2725+0.6898 =2.9623, which is very close to the simulation value of 2.9902 for the noisy data. After de-noising, the AWSS shows a significant decrease in value, indicating that the cluster has become more compact.

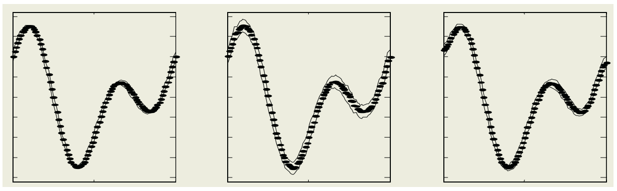


Fig. 3. Illustration of the compactness of a simulated cluster. The cluster is represented by the average profile pattern in the cluster (dot line). Two curves indicate the standard deviation of average expression. (a) Original noiseless data (AWSS=0.6898). (b) Noisy data (AWSS= 2.9902). (c) De-noised data (AWSS=1.2585).

4.2 Clustering Yeast Cell Cycle Data

The yeast cell cycle data set has established itself as a standard for the assessment of newly developed clustering algorithm. This data set contains 6601 genes with 17 time points for each gene taken at 10-min intervals covering nearly two yeast cell cycles. This data set is very attractive because a large number of genes in it are biologically characterized and have been assigned to different phase of the cell cycle.

The raw expression profiles are downloaded from http://genomics.stanford.edu. Firstly, we eliminate those genes whose expression levels were relatively low and did not show significant changes during the entire time course by a variation filter with criteria: (a) the value of expression profile at all 17 time points is equal to or greater than 100; (b) the ratio of the maximum and the minimum of each time-course expression profiles is at least equal to or greater than 2.5. 1368 gene expression profiles passed the variation filter and were normalized to be within 0 and 1.

The filtered gene expression profile data was clustered according to the procedure (a)~(f) described in section 3 by setting f_σ=5% for each gene expression profile. The number of cluster was set to 20 that was estimated for Yeast Cell Cycle gene expression profile data using the same data filtering method as above. Fig.4 is the clustering results in the spectral space. The curve in each subplot in Fig.4 indicates the representative patterns of each corresponding cluster. Fig.5 is the corresponding clustering results by mapping the clustering results in spectral space back into the original space. We also applied the same clustering algorithm directly to the original gene expression data. The clustering results are showed in Fig.6.

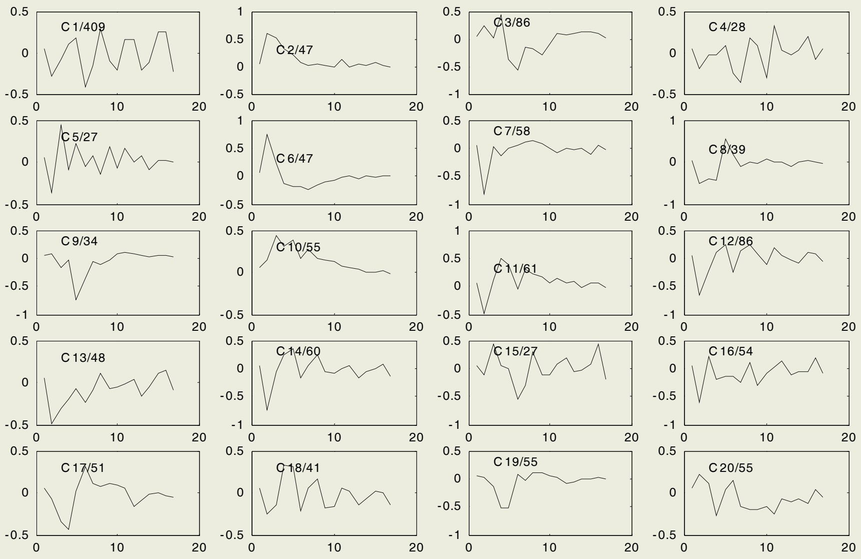


Fig. 4. The clustering results for the yeast cell cycle data in the DCT domain. The curve in each subplot indicates the representative patterns of corresponding cluster. C*m/n* denotes cluster #*m* containing *n* individual profiles.

From the results showed in Fig. 5 and Fig.6, we can see that the clustering results from the proposed method are better than the results obtained directly from the gene expression data. It is obvious that in Fig.6, clusters #1, #11 and #14 are visually similar and they should be clustered into one cluster. In contrast, the clusters obtained in the DCT domain have no apparent visual similarity between them and visually similar clusters (i.e., clusters #1, #11 and #14 in Figure 6) were successfully clustered into one cluster (cluster #1 in Fig.5). This shows that the proposed method here can produce more compact and accurate clusters.

5 Conclusions

Cluster analysis is an important tool in gene expression data analysis. Better clustering results depend not only on the choice of clustering algorithm but also on the input

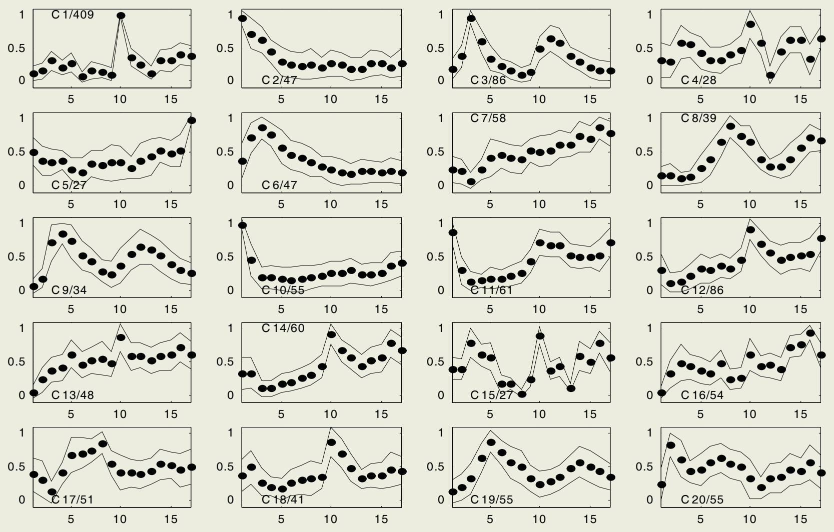


Fig. 5. The corresponding clusters for the yeast cell cycle data by mapping the clustering results in the spectral space back into original space. Each cluster is represented by the average profile pattern in the cluster (dot line). Two curves indicate the standard deviation of average expression.

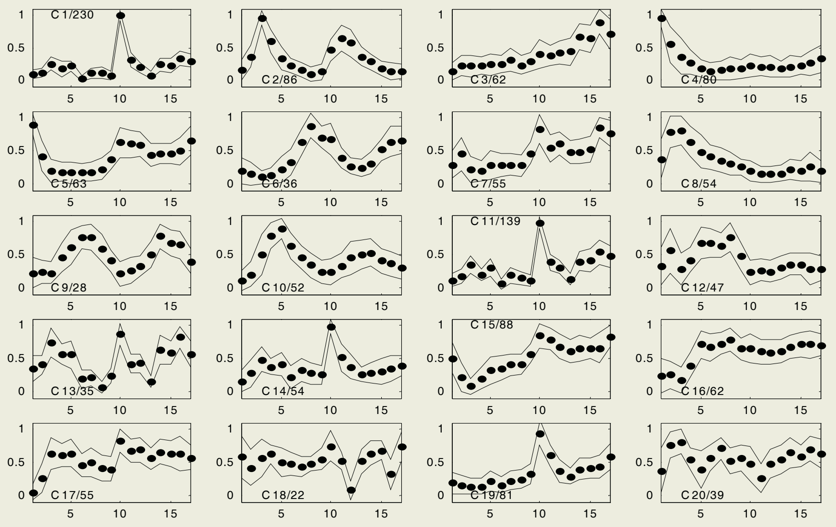


Fig. 6. The clustering results obtained by applying the same clustering algorithm directly to original gene expression data.

features. Generally, gene expression profile data are measured with various experimental errors, so directly clustering original data may not produce reliable results. We show theoretically by assuming that the noise is independent and uncorrelated with the signal, the compactness of the clusters, as measured by the average within-class sum-of-squared error, can be improved by noise reduction. In this paper, we exploit the strong temporal correlation between time point and the periodicity behavior of temporal gene expression profiles to produce better clustering results compared with that obtained directly from the original gene expression data. For spectral decomposition, the DCT transform is chosen. Our spectral domain clustering framework consists of two steps: (1) signal de-noising using soft-thresholding and (2) clustering in the spectral domain using the de-noised features. Since the DCT effectively separate the signal term and the noise term, noise reduction with soft-thresholding can be achieved without sacrificing signal quality. Using a newly developed clustering algorithm capable of searching for natural clusters in the data, the clustering results for real temporal gene expression profiles show that our method is better than a direct clustering of the original expression data.

References

1. Wen, X., Fuhrman, S., Michaels, G.S., Carr, D.B., Smith, S., Barker, J.L., Somogyi, R.: Large-scale Temporal Gene Expression Mapping of Central Nervous System Development. Proc. Natl. Acad. Sci. USA, **95** (1998) 334-339
2. Gilbert, D.A., Ferreria, G.M.: Problems Associated with the Study of Cellular Oscillations. Cell biol. Int, **24** (2000) 501-514
3. Arkin, A., Shen, P., Ross, J.: A Test Case of Correlation Metric Construction of a Reaction Pathway from Measurements. Science, **277** (1997) 1275-1279
4. Friddle, C.J., Koga, T., Rubin, E.M., Bristow, J.: Expression Profiling Reveals Distinct Sets of Genes Altered during Induction and Regression of Cardiac Hypertrophy. Proc. Natl. Acad. Sci. USA, **97** (2000) 6745-6750
5. Saban, M.R., Hellmich, H., Nguyen, N.B., Winston, J., Hammond, T.G., Saban, R.: Time Course of LPS-induced Gene Expression in a Mouse Model of Genitourinary Inflammation. Physiol Genom, **5** (2001) 147–160
6. Tavazoie, S., Hughes, J.D., Campbell, M.J., Cho, R.J., Church, G..M.: Systematic Determination of Genetic Network Aarchitecture. Nat Genet, **22** (1999) 281–285
7. Eisen, M.B., Spellman, P.T., Brown, P.O., Botstein D.: Cluster Analysis and Display of Genome-wide Expression Patterns. Proc. Natl Acad. Sci. USA, **95** (1998) 14863-14868
8. Liang, S., Fuhrman, S., Somogyi, R.: Reveal a General Reverse Engineering Algorithm for Inference of Genetic Network Architectures. Pac Symp Biocomput, **1** (1998) 18–29
9. Moler, E.J., Radisky, D.C., Mian, I.S.: Integrating Naive Bayes Models and External Knowledge to Examine Copper and Iron Homeostasis in S. cerevisiae. Physiol Genomics, **4** (2000) 127–135
10. Ben-Dor, A., Shamir, R., Yakhini, Z.: Clustering Gene Expression Patterns. J Comput Biol, **6** (1999) 281–297
11. Getz, G., Levine, E., Domany, E., Zhang, M.Q.: Super-paramagnetic Clustering of Yeast Gene Expression Profiles. Physica A, **279** (2000) 457–464
12. Lukashin, A.V., Rainer, F.: Analysis of Temporal Gene Expression Profiles: Clustering by Simulated Annealing and Determining the Optimal Number of Clusters. Bioinformatics, **17** (2001) 405-414
13. Atul, J.B., Ling, B., Ben, Y., Timothy, W.W., and Isaac, S.K.: Comparing the Similarity of Time-series Gene Expression using Signal Processing Metrics. Journal of Biomedical Informatics, **34** (2001) 396-405
14. Edward, R.D.: Random Processes for Image and Signal Processing. Bellingham. Washington: SPIE Optical Engineering Press; Institute of Electrical and Electronics Engineers, New York (1999)

15. Donoho, D.L.: De-noising by Soft-thresholding. IEEE Transactions on Information Theory, **41** (1995) 613-627
16. Zhang, Y.Z., Liu, Z.Q.: Self-Splitting Competitive Learning: A New On-line Clustering Paradigm. IEEE Transactions on Neural Networks, **13** (2002) 369-380
17. Neal, S.H., Madhusmita, M., Amos, M., Cieplak, M., Banavar, J.R., Fedoroff, N.V.: Fundamental Patterns Underlying Gene Expression Profiles: Simplicity from Complexity. Proc Natl Acad Sci USA, **97** (2000) 8409–8414
18. Wu, S., Liew, A.W.C., Yan, H.: Cluster Analysis of Gene Expression Data based on Self-Splitting and Merging Competitive Learning. IEEE Transactions on Information Technology in Biomedicine, **8** (2004) 5-14

Model the Relationship Between Gene Expression and TFBSs Using a Simplified Neural Network with Bayesian Variable Selection

Xiaobo Zhou[1,2], Kuang-Yu Liu[1,2], Guangqin Li[3], and Stephen Wong[1,2]

[1] HCNR-Center for Bioinformatics, Harvard Medical School, MA 02215, USA
zhou@crystal.harvard.edu
[2] Radiology Department, Brigham and Women's Hospital, Boston, MA 02115, USA
[3] Department of Neurology, The First Affiliated Hospital, Chongqing University of Medical Sciences, Chongqing 400014, China

Abstract. Although numerous computational methods consider the identification of individual transcription factor binding sites (TFBSs), very few focus on the interactions between these sites. In this study, we study the relationship between transcription factor binding sites and microarray gene expression data. A probit regression with one linear term plus nonlinear (it is actually a simplified neural network) is used to build a predictive model of outcome of interest (either gene expression ratios or up- and down-regulations) using these transcription factor binding sites. This issue is related to the more general problem of expression prediction in which we want to find small subsets of TFBSs to be used as predictors of possible co-expressed genes and those genes do share some DNA regulatory motifs. Given some maximum number of predictors to be used, a full search of all possible predictor sets is prohibitive. This paper considers Bayesian variable selection for prediction using the nonlinear probit model (or simplified neural network). We applied this nonlinear model with Bayesian motif selection on one gene expression data set. These TFs demonstrated intricate regulatory roles either as a family or as individual members and our analysis created plausible hypotheses for combinatorial interaction among TFBSs.

1 Introduction

The expression of a gene is controlled by many mechanisms. A key factor in these mechanisms is mRNA transcription regulation by various proteins, known as transcriptional factors (TFs), which bind to specific sites in the promoter region of a gene that activate or repress transcription. Genome sequences specify the gene transcription and translation activities that produce RNAs and proteins to support living cells, but how cells control global gene expression is far from transparent [10]. The transcriptional regulatory apparatus is organized in the form of arrays of transcription factor binding sites (TFBSs) or motifs on DNA. Identifying the components of this array, and the relationships among them is one of the challenging problems of contemporary biology. Transcriptional regulation in eukaryotic organisms requires cooperation of multiple transcription factors. To date, most computational methods focus on identifying single or multiple TFBSs rather than exploring their interdependence in regulation. Recently,

[3] developed a method called MOTIF REGRESSOR. This approach combines binding site identification using position weight matrices, and the linear regression approach for motif finding [2, 4]. All of those approaches [2–4] also first identifies potential TFBSs from groups of genes separately and then uses linear regression to model gene expression as a function of these sites. In this study, we will use transcriptional factor library TRANSFAC Professional [6], whose binding site sequence information has been verified experimentally, to find highly likely TFBS candidates, and study the relationship between gene expression and TFBS candidates, without introducing too many false positive candidates in the modeling process.

A simplified neural networks or nonlinear probit regression is used to build a predictive model of outcome of interest (either gene expression ratios or up- and down-regulations) using these transcription factor binding sites. This issue is related to the more general problem of expression prediction in which we want to find small subsets of TFBSs to be used as predictors of possible co-expressed genes and those genes do share some DNA regulatory motifs. Given a large number of predictors to be used, a full search of all possible predictor sets is combinatorial prohibitive. This paper considers Bayesian variable selection for prediction using a linear regression model. The nonlinear probit regressor is approximated as a linear plus a nonlinear combination of the TFBSs and a Gibbs sampler is employed to find the TFBSs with the strongest predictive power. We final applied this model to spinal cord injury (SCI) data analysis.

2 Material Preparation

As a first step of pre-preprocessing microarray data, we perform missing value estimation to the all gene expression data. A weighted K-nearest neighbor method based method select features of expression profiles similar to the gene of interest to impute missing values [8]. The list of genes is screened further for the ones with maximum fold changes. We first calculate the p-values of each gene by the ratio of the between-group mean sum of squares to the within-group (F test). Only those genes whose p-values are extremely small are retained. Next we choose two maximum concordant sets of the top 30 up-regulated and down-regulated genes. A combined discordant set of genes is obtained with the top 15 genes from each of the maximum concordant sets. As a result, there are three lists of genes for each microarray experiment data set.

Sequence database for up-regulated concordant genes combined discordant genes is generated for human and rat. Chromosomal location and sequence of these genes are collected from Homo sapiens Genome assembly Build 35 (NCBI35), and Rat Genome at http://www.hgsc.bcm.tmc.edu. Genome sequences with up to 800 bases upstream from the $5\prime$ end of the sense-strand $(+)$, are retrieved from Ensemble (http://www.ensembl.org) for each gene in the three lists. To search for potential transcription factor binding sites in the sequence database, MATCH, and PATCH from TRANSFAC Professional[6] are used to avoid false positives of potential regulatory binding sites. In this study [4], a dichotomized motif score is computed as

$$s_{g,h} = \begin{cases} 1, \text{ if gene } g \text{ has at least one copy of motif } h, \\ 0, \qquad\qquad\qquad \text{o.w.} \end{cases} \tag{1}$$

3 TFBSs Selection

3.1 Problem Formulation

Let $\boldsymbol{w}$ denote the gene expression profile, where $\boldsymbol{w}$ could be continues, binary, or ternary expression profiles. As a discrete vector variable, $\boldsymbol{w}$ represents the class of genes, e.g., 0 for down-regulated genes and 1 for up-regulated genes in binary format. We assume to have independent and identically distributed (i.i.d.) observations of random variable $\boldsymbol{w}$. For any given potential binding site set of size n and a list of genes $g = 1, \cdots, n$, we define a covariate motif score matrix

$$\boldsymbol{X} = \begin{bmatrix} \text{Motif}_1 & \text{Motif}_2 & \cdots & \text{Motif}_p \\ s_{1,1} & s_{1,2} & \cdots & s_{1,p} \\ s_{2,1} & s_{2,2} & \cdots & s_{2,p} \\ \vdots & \vdots & \ddots & \vdots \\ s_{n,1} & s_{n,2} & \cdots & s_{n,p} \end{bmatrix} \tag{2}$$

where the entries of this matrix are the sequence motif-matching score of motif h for gene g, $s_{g,h}$. Due to the small sample size, here we adopt a probit regression model composed of a linear term plus a nonlinear term. Then w_i and the TFBSs scores $\boldsymbol{x}_i \triangleq [x_{i,1}, x_{i,2}, ..., x_{i,p}]$ are related through [1]

$$P(w_i = 1 \mid \boldsymbol{x}_i) = \Phi\left(\sum_{j=1}^{p} a_j x_{ij} + \sum_{k=1}^{K} b_k \phi(\boldsymbol{x}_i, \boldsymbol{\mu}_k)\right), \tag{3}$$

$$\text{with} \;\; \phi(\boldsymbol{x}_i, \boldsymbol{\mu}_k) \triangleq \exp\{-\lambda_k \|\boldsymbol{x}_i - \boldsymbol{\mu}_k\|^2\}, \tag{4}$$

where $\phi(\cdot)$ is a radial basis function; $\|\cdot\|$ denotes a distance metric (usually Euclidean or Mahalanobis); $\boldsymbol{\beta} \triangleq [a_1, a_2, ..., a_p, b_1, ..., b_K]^T$ contains the parameters and Φ is the standard normal cumulative distribution function; $\boldsymbol{\mu}_k = [\mu_{k,1}, ..., \mu_{k,m}]$ contains the centers of the K clusters, whose values are obtained by using the fuzzy C-means clustering algorithm [9]. The parameters $\{\lambda_k\}_{k=1}^{K}$ are empirically set as 1.0. In this study, we fix $K = 2$. The motivation to setting K centers by using clustering is that we consider only consider up-regulate and down-regulate genes in this study. Define the following n independent latent variable $y_1, ..., y_n$:

$$y_i = \sum_{j=1}^{p} a_j x_{ij} + \sum_{k=1}^{K} b_k \phi(\boldsymbol{x}_i, \boldsymbol{\mu}_k) + e_i, \;\; i = 1, ..., n, \tag{5}$$

where $e_i \sim \mathcal{N}(0, 1)$. Denote $\boldsymbol{y} \triangleq [y_1, ..., y_n]^T$, $\boldsymbol{e} \triangleq [e_1, ..., e_p]^T$ and

$$\boldsymbol{D} \triangleq \begin{bmatrix} x_{1,1} & \cdots & x_{1,p} & \phi(\boldsymbol{x}_1, \boldsymbol{\mu}_1) & \cdots & \phi(\boldsymbol{x}_1, \boldsymbol{\mu}_K) \\ x_{2,1} & \cdots & x_{2,p} & \phi(\boldsymbol{x}_2, \boldsymbol{\mu}_1) & \cdots & \phi(\boldsymbol{x}_2, \boldsymbol{\mu}_K) \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ x_{n,1} & \cdots & x_{n,p} & \phi(\boldsymbol{x}_n, \boldsymbol{\mu}_1) & \cdots & \phi(\boldsymbol{x}_n, \boldsymbol{\mu}_K) \end{bmatrix}. \tag{6}$$

Then (5) can be expressed in a vector form as

$$\boldsymbol{y} = \boldsymbol{D}\boldsymbol{\beta} + \boldsymbol{e}, \tag{7}$$

where $\boldsymbol{e} \triangleq [e_1, ..., e_n] \sim \mathcal{N}(\boldsymbol{0}, \mathcal{I}_n)$. Define $\boldsymbol{\gamma} = [\gamma_1, ..., \gamma_{p+K}]$ as the $(p + K) \times 1$ indicator vector with the jth element γ_j such that $\gamma_j = 0$ if $\beta_j = 0$ (the variable is not selected) and $\gamma_j = 1$ if $\beta_j \neq 0$ (the variable is selected). Given $\boldsymbol{\gamma}$, let $\boldsymbol{\beta}_{\boldsymbol{\gamma}}$ consist of all nonzero elements of $\boldsymbol{\beta}$ and let $\boldsymbol{X}_{\boldsymbol{\gamma}}$ be the columns of $\boldsymbol{X}$ corresponding to those of $\boldsymbol{\gamma}$ that are equal to 1. In this study, for simplicity, we fix the nonlinear term and only consider TFBS selection. Hence $\boldsymbol{\gamma} = [\gamma_1, ..., \gamma_p, 1, 1]$, i.e., $\gamma_{p+1} = 1$ and $\gamma_{p+2} = 1$.

3.2 Bayesian TFBSs Selection

A Gibbs sampler is employed to estimate all the parameters. Given $\boldsymbol{\gamma}$, the prior distribution of $\boldsymbol{\beta}_{\boldsymbol{\gamma}}$ is $\boldsymbol{\beta}_{\boldsymbol{\gamma}} \sim \mathcal{N}(\boldsymbol{0}, c({\boldsymbol{X}_{\boldsymbol{\gamma}}}^T \boldsymbol{X}_{\boldsymbol{\gamma}})^{-1})$, where c is a constant (we set $c = 100$ [5] in this study). Since the detailed derivations of the posterior distributions of the parameters are similar to those in [5], here we simply summarize the procedure for Bayesian variable selection. Denote

$$S(\boldsymbol{\gamma}, \boldsymbol{y}) \triangleq \boldsymbol{y}^T \boldsymbol{y} - \frac{c}{c+1} \boldsymbol{y}^T \boldsymbol{X}_{\gamma} ({\boldsymbol{X}_{\gamma}}^T \boldsymbol{X}_{\gamma})^{-1} {\boldsymbol{X}_{\gamma}}^T \boldsymbol{y}. \tag{8}$$

Then the Gibbs sampling algorithm for estimating $\boldsymbol{\gamma}, \boldsymbol{\beta}, \boldsymbol{y}$ is as follows:

- Draw $\boldsymbol{\gamma}$ from $p(\boldsymbol{\gamma} \mid \boldsymbol{y})$, where $p(\boldsymbol{\gamma} \mid \boldsymbol{y}) \propto (1+c)^{-\frac{p_{\gamma}}{2}} \exp\left[-\frac{1}{2} S(\boldsymbol{\gamma}, \boldsymbol{y})\right] \prod_{j=1}^{p+2} \pi_j^{\gamma_j}$ $(1 - \pi_j)^{1-\gamma_j}$. Here $p_{\gamma} = \sum_{j=1}^{p+2} \gamma_j$ and $\pi_j = P(\gamma_j = 1)$ is a prior probability to select the jth TFBS. This parameter is often set as a small number due to small sample size. If π_j is chosen in a bigger value, then we found that often times $({\boldsymbol{X}_{\boldsymbol{\gamma}}}^T \boldsymbol{X}_{\boldsymbol{\gamma}})^{-1}$ does not exist. We usually sample each γ_j independently from

$$p(\gamma_j \mid \boldsymbol{y}, \gamma_{\neq j}) \propto (1+c)^{-\frac{p_{\gamma}}{2}} \exp\left[-\frac{1}{2} S(\boldsymbol{\gamma}, \boldsymbol{y})\right] \pi_j^{\gamma_j} (1 - \pi_j)^{1-\gamma_j}, \tag{9}$$

  where $\boldsymbol{\gamma}_{\neq j} \triangleq (\gamma_1, ..., \gamma_{j-1}, \gamma_{j+1}, ..., \gamma_p, 1, 1)$.
- Draw $\boldsymbol{\beta}$ from $p(\boldsymbol{\beta} \mid \boldsymbol{\gamma}, \boldsymbol{y}) \sim \mathcal{N}(\boldsymbol{V}_{\gamma} \boldsymbol{X}_{\gamma}{}^T \boldsymbol{y}, \boldsymbol{V}_{\gamma})$, where $\boldsymbol{V}_{\gamma} = \frac{c}{1+c} ({\boldsymbol{X}_{\gamma}}^T \boldsymbol{X}_{\gamma})^{-1}$.
- Draw $y_i,\ i = 1, ..., n$ from a truncated normal distribution as follows: $p(y_i \mid \boldsymbol{\beta}, w_i = 1) \propto \mathcal{N}(\boldsymbol{X}_i \boldsymbol{\beta}, 1) 1_{\{y_i > 0\}}$ and $p(y_i \mid \boldsymbol{\beta}, w_i = 0) \propto \mathcal{N}(\boldsymbol{X}_i \boldsymbol{\beta}, 1) 1_{\{y_i < 0\}}$.

In this study, 10000 Gibbs iterations are implemented with the first 2000 as the burn-in period. Then we obtain the Monte Carlo samples as $\{\boldsymbol{\gamma}^{(t)}, \boldsymbol{\beta}^{(t)}, \boldsymbol{y}^{(t)},\ t = 1, ..., T\}$, where $T = 10000$. Finally we count the number of times that each TFBS appears in $\{\boldsymbol{\gamma}^{(t)},\ t = 2001, ..., T\}$. The TFBSs with the highest appearance frequencies play the strongest role in predicting the gene expression profiles. Some implementation issues of this algorithm is similar to the Bayesian gene selection and prediction method in [11].

4 Results

Inflammatory responses often exacerbates secondary tissue damage following spinal cord injury, and counter treatments in reducing inflammation and in sparing secondary damage have been actively investigated. The microarray experiment of five anti-inflammatory compounds [7] using rat acute spinal contusion as injury model is aimed to understand better subsequent physiological events in secondary injury and to facilitate better treatments in the future. In particular, [7] have shown that a consistent and unique gene expression profiles is associated NS398, the selective cyclooxygenase-2 inhibitor. They also suggest that overall effect of these up-regulated genes can be interpreted as neuroprotective. Hence our modeling is concentrated on the effect of NS398 on gene expression levels.

Table 1. The top 10 appearance frequencies for Discordant genes. Condition 1: Equ. (1) is calculated for all possible motifs, and condition 2 for common motifs.

	Condition 1		Condition 2	
Index	Transcriptional-Factor	Frequency	Transcriptional-Factor	Frequency
1	NFKB	0.5421	Myc	0.6637
2	E2F1	0.4334	NFKB	0.5634
3	RUNX1	0.3637	LEF1	0.5457
4	Zic2	0.2832	TFAP2	0.4873
5	CREB	0.2758	COUP	0.4666
6	SMAD	0.2323	RAR	0.4452
7	MSX1	0.2167	SMAD	0.4417
8	RUNX1a	0.2056	Zic3	0.4205
9	MAZ	0.1871	MSX1	0.3821
10	Myc	0.1662	STAT4	0.3261

Upon finishing the estimation of missing values, we select genes whose expression profiles change significantly between the NS398 treatment and the vehicle (injury without treatment), using their associated p-values as selection criteria. These 2574 genes, from the total of 5376 genes, are then screened for maximum fold changes in terms of concordant up-regulated expressions (Concordant+), concordant down-regulated expressions (Concordant−), and maximum discordant expressions (Discordant). Here we only present the discordant case due to space limited. Motif selection results with the top 10 average appearance frequencies, i.e. the ones play the strongest roles in predicting the expression level, are given in Table 1 for Discordant genes in SCI data. The NFKB is almost most important TFBS chosen by using the proposed modelling. Another transcriptional factor SMAD is found in the both list. And also we found that among motifs selected more than 90% of the times, there are NFKB and SMAD family-wise motifs for Concordant+ genes, as well as PAX4 and TFAP2C for Concordant− genes. NFKB, nuclear factor of kappa light polypeptide gene enhancer, is a transcription regulator that is activated by various intra- and extra-cellular stimuli such as cytokines, oxidant-free radicals, ultraviolet irradiation, and bacterial or viral products.

These transcription factors demonstrated intricate regulatory roles either as a family or as individual members during transcription activation and repression. Our analysis using simplified neural network (or nonlinear probit model) Bayesian motif selection created plausible hypotheses for combinatorial interaction among transcription factor binding sites that were based on experimentally verified binding site sequence information. Hence our approach also offered additional advantage of avoiding false positive candidates in the modeling process.

References

1. Albert, J. and Chib, S.: Bayesian Analysis of Binary and Polychotomous Response Data. J. of the American Statistical Association, **88** (1993) 669-679
2. Bussemaker, H.J.: Regulatory Element Detection Using Correlation with Expression. Nat. Genet., **27** (2001) 167-71
3. Conlon, E.M., et. al.: Integrating Regulatory Motif Discovery and Genome-wide Expression Analysis. Proc. Natl. Acad. Sci. **100** (2003) 3339-44
4. Keles, S., et.al.: Regulatory Motif Finding by Logic Regression. Bioinformatics **20** (2004) 2799-811
5. Lee, K.E., et. al.: Gene Selection: a Bayesian Variable Selection Approach. Bioinformatics **19** (2003) 90-97
6. Matys, V., et. al.: TRANSFAC: Transcriptional Regulation, from Patterns to Profiles. Nucleic Acids Res. **31** (2003) 374-378
7. Pan, J. Z.: Screening Anti-inflammatory Compounds in Injured Spinal Cord with Microarrays: a Comparison of Bioinformatics Analysis Approaches. Physiol Genomics. **17** (2004) 201-214
8. Zhou, X., et. al.: Missing Value Estimation Based on Linear and Nonlinear Regression with Bayesian Gene Selection. Bioinformatics **19**(2003) 2302-2307
9. Zhou, X., et. al.: Gene Clustering Based on Cluster-wide Mutual Information. J. of Computational Biology **11** (2004) 151-165
10. Zhou, X., et. al.: A Bayesian Connective-based Approach to Constructing Probabilistic Gene Regulatory Networks. Bioinformatics **20** (2004) 2918-2927
11. Zhou, X., et. al.: Gene Prediction Using Multinomial Probit Regression with Bayesian Variable Selection. J. of Applied Signal Processing, **3** (2004) 115-124

Synchrony of Basic Neuronal Network Based on Event Related EEG

Xiaotong Wen, Xiaojie Zhao, and Li Yao

Department of Electronics, Beijing Normal University, Beijing 100088, China
zhaoxj86@hotmail.com

Abstract. The activities of neuronal network are realized through interaction of different nerve cell assemblies. EEG synchrony reflects the coordination among neuronal network. As usual, phase synchronization methods are used to detect the phase difference between narrowband signals from two different electrodes, and are applied to clinic such as the localization of epilepsy. In this paper, we combined a filter bank with a phase synchronization method and applied the method to event related EEG of cognitive task to obtain a dynamic spatial-temporal synchronization distribution in different rhythm. The results showed that the synchronal activities of neurons did match ERP component in the cognitive experiment. It gave explanations to interaction of the underlying neuronal networks and to information processing mechanism in cognitive process in some sense.

1 Introduction

More and more researchers have been devoted to the localization of brain functions and the coordination mechanism among the different neuronal structures. EEG signals are bioelectrical activities of numerous neurons and reflect the brain oscillations caused by synchronous depolarization of nerve cell assemblies. With the study of coordination mechanism of neuronal networks based on the synchronous activities of EEG, the patterns of information processing and transfer can be presented more clearly. They can be used to answer some open questions, such as how the nerve cell assemblies are organized for specific cognitive task, what the interaction of participating neuronal network is and how they coordinate [1]. By analyzing the electrical activities of neurons recorded via microelectrode in animal experiments, it was found that the integration of information among nerve cell assemblies could be implemented through the synchronous electrical activity [2].

In recent years, some phase synchrony measuring methods are introduced to explore the phenomena of neuronal coordinated actions. Tass et al introduced a method to detect phase synchrony from nonstationary signals and applied it to MEG signals [3]. Lachaux et al extended the method and made applications to the subdural EEG recordings of epileptic patient performing visual discrimination task [4]. Le Van Quyen et al gave the comparison of these methods and found that they were fundamentally equivalent for the study of neuro-electrical signals [5]. However, most of the methods mentioned above were carried out in a narrow frequency band (usually centering around the interesting frequency with a frequency band less than 4Hz) [6][7][8], and applied to the clinical pathology recordings. In this paper, we designed a filter bank for phase synchrony method to obtain a time-frequency distribution of phase synchrony index, which could offer high time-frequency resolution in the broad frequency band (about 30Hz). The method was applied to event related EEG of cognitive task, and the results gave the explanations to dynamic coordinate activities of neuronal network in some sense.

2 Time-Frequency Analysis of Phase Synchrony

2.1 Methods of Measuring Phase Synchronization

When phase synchrony in EEG signals is detected, the instantaneous phase should be extracted first. Considering the narrowband signal $s_{fc,w}(t)$ with bandwidth about 2Hz (from fc-w/2 to fc+w/2), we get analytical signal $s_a(t)$ of $s_{fc,w}(t)$ by equation (1):

$$s_a(t) = s_{f_c,w}(t) + i(\tilde{s}_{f_c,w}(t)) = A(t)e^{j\phi(t)} . \tag{1}$$

where $\tilde{s}_{f_c,w}(t)$ is the Hilbert transform of $s_a(t)$:

$$\tilde{s}_{f_c,w}(t) = \frac{1}{\pi}\int_{-\infty}^{+\infty} \frac{s_{f_c,w}(\tau\tau}{t-\tau} d\tau . \tag{2}$$

Then we can extract instantaneous phase value $\phi(t)$ of $s_{fc,w}(t)$ from $s_a(t)$ and calculate instantaneous phase difference between two different $s_{fc,w}(t)$ as below:

$$\phi_d(t) = \phi_x(t) - \phi_y(t) . \tag{3}$$

where x and y refer to electrode x and electrode y. In this way, the phase synchrony between x and y can be measured according to $\phi_d(t)$ of different trails.

In synchrony case, the distribution of $\phi_d(t)$ across trials should be unimodal on [-π, π], the more concentrated the distribution is, the more greater the probability of phase synchronization is. To quantify this, three methods are used.

Method 1. Calculating phase synchrony index based on Shannon entropy (ISE)
ISE is defined as [3]:

$$\gamma = (H_{\max} - H) / H_{\max} , H_{\max} = \ln N , \; H = \sum_{k=1}^{M} p_k \ln p_k . \tag{4}$$

where $H_{\max}$ is the maximum entropy, N is the number of samples (the trials), M is the number of bins and estimated as $M=e^{0.626+0.4\ln(N-1)}$, p_k means the probability of finding $\phi_d(t)$ in the bin k. Normalized in this way, γ=0 corresponds to no synchronization and γ=1 to perfect synchronization.

Method 2. Calculating phase synchrony index based on conditional probability (ICP)
ICP is defined as [3]:

$$\tilde{\lambda}(t) = \frac{1}{M}\sum_{l}^{M} | r_l(t) | , \; r_l(t) = \frac{1}{M_l} \sum_{\phi_x(t) \in Bin(l)} e^{i\phi_y(t)} . \tag{5}$$

where M is number of bins that the definition domain [-π, π] is divided into, l is the index of the bins. For each $r_l(t)$, $e^{i\phi_y(t)}$ are vectors on the unit circle. $|r_l(t)|$ shows the dependences between $\phi_x(t)$ in a certain bin and their corresponding $\phi_y(t)$. Obviously, high dependence leads to the result that the vector $r_l(t)$ approaches the unit circle on the complex plane. Therefore, the phase synchrony occurs almost definitely when $\tilde{\lambda}(t)$ approaches 1.

Method 3. Calculating phase locking value (PLV) as phase synchrony index PLV is calculated as [4]:

$$PLV(t) = |u(t)| \, , \; u(t) = \frac{1}{N}\sum_{n=1}^{N} e^{j\phi_d(t)} \; . \tag{6}$$

where N is the number of trials. A unimodal distribution of $\phi_d(t)$ across trials will surely make the vector $u(t)$ approach the unit circle on the complex plane. Namely, complete phase synchrony leads to $PLV(t)$=1, oppositely $PLV(t)$=0.

2.2 FIR Filter Bank Design

Because the rhythmic oscillations of event related EEG are often in a comparatively broad frequency band and the stimulation occurs in a short time period, the phase synchrony indices should be able to represent synchronization distribution over a broad frequency band with high time-frequency resolution. In order to measure phase synchrony in broad frequency band, a band-pass FIR filter bank was designed. Each filter (length=258ms, sampling frequency=500Hz) had the same bandwidth (2Hz) and the filter bank covered the whole interesting band with equal frequency intervals. In order to make the negative affect of band-pass filtering as small as possible, the filters were windowed by Blackman-Harris window. For each filter, phase synchrony index corresponding to frequency band of the filter was calculated. The phase synchrony distribution could be gotten finally by combining all indices over whole band. The frequency resolution of distribution mainly depended on the frequency interval. The less the interval was, the higher the resolution was.

3 Experiment Results

3.1 Simulation

The simulated signal sets were created from independent real EEG signal set X and set Y (each had 69 trials). For each trial $s_1(t)$ in set X and its corresponding trial $s_2(t)$ in set Y, we got two narrowband signals $s_1^{fc}(t)$ and $s_2^{fc}(t)$ by band-pass filter with center frequency fc. Using Bell function $Bell(t)$ (as illustrated in Fig. 1), we created simulated signals $c_1(t)$ and $c_2(t)$ by following this:

$$c_1(t) = s_1(t) \, , \; c_1^{fc}(t) = Bell(t)\cdot s_1^{fc}(t) + (1 - Bell(t))\cdot s_2^{fc}(t) \, , \tag{7}$$

$$c_2(t) = s_2(t) - s_2^{fc}(t) + rand\cdot c_1^{fc}(t) \, . \tag{8}$$

rand were different between 0 and 1 for each trial and different t. These random values made the amplitudes of the signals independent. So, any synchronization measured only related with phase, not amplitude. From equation (7~8), the phase information of $s_1(t)$ around frequency fc in the none-zero period (400ms-650ms) of $Bell(t)$ was extracted, and then replaced that of $s_2(t)$ in corresponding period. Using the method, two new simulated signal sets Xsimu and Ysimu were reconstructed along frequency range 1~30 Hz.

Applying the three methods of measuring phase synchrony to Xsimu and Ysimu, the time-frequency distribution results were shown in Fig. 2. Although all results displayed the same synchrony area and were compatible with the simulation conditions, the results of ISE, Fig. 2(a) separated synchrony area more clearly. Furthermore, ISE method was easy to implement in programming and ran fast. So in the later processing of real EEG data, we mainly applied ISE to calculate the phase synchrony indices.

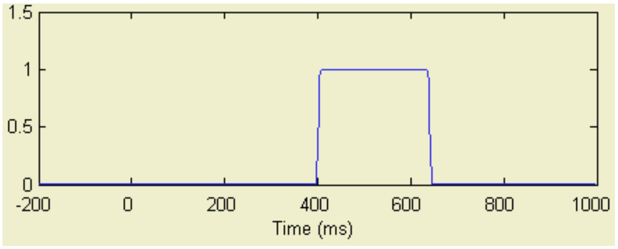


Fig. 1. Bell function with a non-zero period of 400~650 ms.

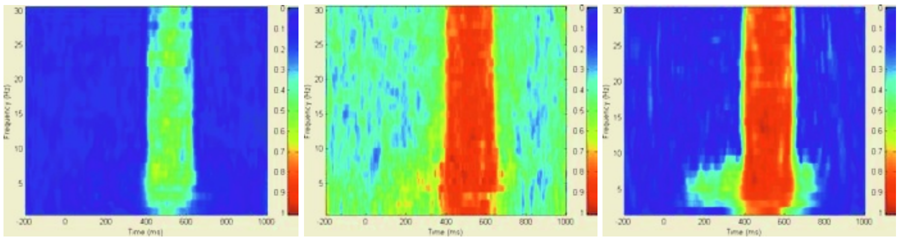


Fig. 2. Results of three methods of measuring phase synchrony. (a) ISE. (b) ICP. (c) PLV.

3.2 Event Related EEG

3.2.1 Cognitive Experiment

Eight healthy students (age 11~12) with right-handed and normal eyesight participated this experiment. Stimuli were 4 colorful Chinese characters (red, yellow, blue and green) in different colors. Subjects were asked to identify the display color of the Chinese character using keystroke. There were four keys and each key denoted a sort of color with the same color block. The subjects were trained to press the corresponding key before experiment in order to build correct mapping between color and keystroke. There were six blocks in the experiment and 96 stimuli in each block with color-meaning consistent word and color-meaning inconsistent word appeared randomly. Baseline corresponded to stare asterisk in the middle of screen.

3.2.2 Data Recording

EEG was recorded from 32 electrodes using ESI 128 channel workshop (NeuroScan, USA) with two referenced electrodes to two mastoids (band-pass 0.05~30 Hz, sampling rate 500 Hz). The analysis time course was at about 1000 ms post stimulus onset with baseline at 200 ms pre-stimulus. We chose four interested regions to analyze. Each 3 electrodes represented each region respectively as F3/Fz/F4 (frontal), CP3/CPz/CP4 (middle), P3/Pz/P4 (parietal) and O1/Oz/O2 (occipital).

3.2.3 Data Analysis

To each electrode, EEG data under the same stimuli were averaged across subjects to get a data set. The data set of each electrode was filtered by the FIR filters bank designed above with frequency range 1~30 Hz and a series of narrow band signal sets with different center frequency were obtained. We used Hilbert transform to extract instantaneous phase of every narrow band signal sets and applied ISE method to get phase synchrony indices between two electrodes. Integrating all indices of different frequency band, we obtained the time-frequency distribution of phase synchrony index of the two electrodes. Results of phase synchrony between 9 pairs of electrodes were chosen to show in Fig. 3.

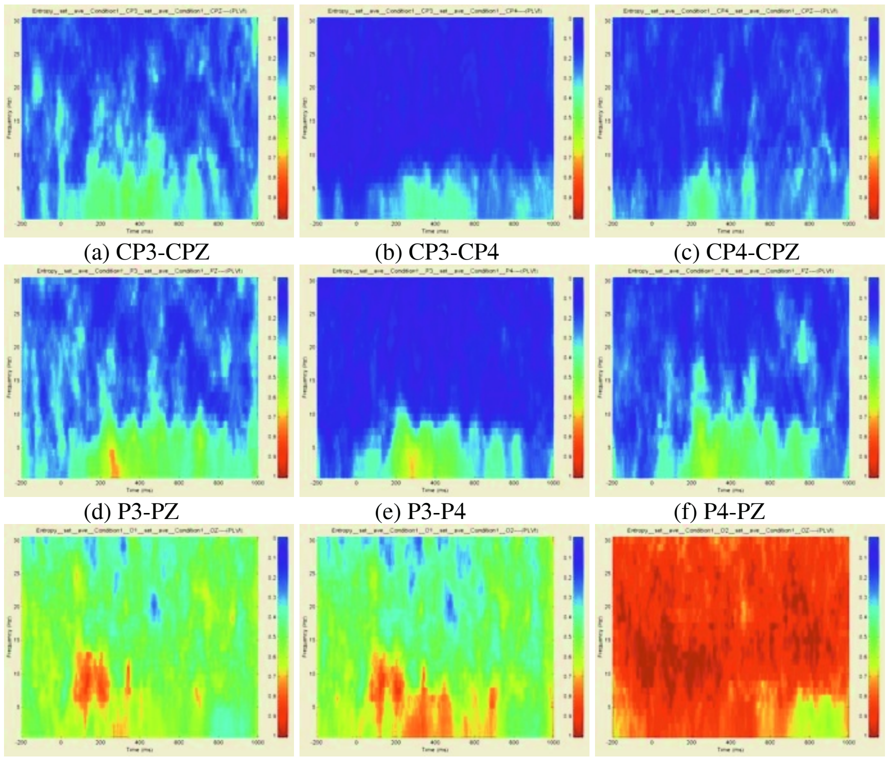

(a) CP3-CPZ (b) CP3-CP4 (c) CP4-CPZ
(d) P3-PZ (e) P3-P4 (f) P4-PZ
(g) O1-OZ (h) O1-O2 (i) O2-OZ

Fig. 3. Time-frequency distribution of phase synchrony between 9 pairs of electrodes.

4 Discussion

The time domain analysis from the experiment had shown there was an ERP component, which was a positive component occurring from 200 ms to 400 ms post stimulus onset [9]. The positive component was mainly distributed in the posterior scalp and should belong to P300 based on its polarity, latency and distribution in scalp. From Fig. 3, we found that the time period of phase synchrony was consistent with the latency of P300 and synchronization frequency was below 15 Hz. The synchrony activities were gradually weakened from posterior sites to frontal sites in the same time period. Especially, O2-OZ phase synchrony index was very high. This phenomena matched the response to visual stimulus in experiment because O2 located occipital lobe cortex was mainly related to visual function. Although O1-OZ and O1-O2 synchrony indices were comparatively high, they were obviously different from that of O2-OZ. This indicated that the coordination probability of nerve cell assemblies corresponding to right occipital lobe was higher. Maybe it was related to brain asymmetry or unilateral effect, maybe it was related to the nature and difficulty of task.

From the results, it was clear that synchrony analysis could reflect the neuronal network coordination of specific cognitive task in different rhythm. Furthermore, we can locate electrical active regions by EEG source localization in order to study brain functional connection more deeply. The study of phase synchrony supplies a way to explore how the neuronal network structures organize, interact and coordinate.

References

1. Weiss, S., Mueller, H.M.: The Contribution of EEG Coherence to the Investigation of Language. Brain Lang, **85** (2003) 325-343
2. Gray, C.M., Konig, P., Engel, A.K., Singer, W.: Oscillatory Responses in Cat Visual Cortex Exhibit Inter-Colummar Synchronization Which Reflects Global Stimulus Properties. Nature, **338** (1989) 334-337
3. Tass, P., Rosenblum, M.G., Weule, J., Kurths, J., Pikovsky, A., Volkmann, J., Schnizler, A., Freund, H.J.: Detection of n:m Phase Locking from Noisy Data: Application to Magnetoencephalography. Phys. Rev. Lett., **81** (1998) 3291-3294
4. Lachaux, J.P., Rodriguez, E., Martinerie, J.: Measuring Phase Synchrony in Brain Signals. Hum. Brain Mapp., **8** (1999) 194–208
5. Le Van Quyen, M., Foucher, J., Lachaux, J.P., Rodriguez, E., Lutz, A., Martinerie, J., Varela, F.J.: Comparison of Hilbert Transform and Wavelet Methods for The Analysis of Neuronal Synchrony. J. Neurosci. Methods, **111** (2001) 83-98
6. Rodriguez, E., George, N., Lachaux, J.P., Martinerie, J., Renault, B., Varela, F.J.: Perception's Shadow: Longdistance Synchronization of Human Brain Activity. Letters To Nature, **397** (1999) 430-433
7. Tallon-Baudry, C., Bertrand, O., Delpuech, C., Pernier, J.: Oscillatory-band (30-70 Hz) Activity Induced by a Visual Search Task in Humans. J. Neurosci., **17** (1997) 722-734
8. Tallon-Baudry, C., Bertrand, O.: Oscillatory Gamma Activity in Humans and Its Role In Object Representation. Trends Cogn. Sci., **3** (1999) 151-162
9. Peng, D.L., Guo, T.M., Wei, J.H., Xiao, L.H.: An ERP Study on Processing Stages of Children's Stroop Effect. Sci. Tech. Engng., **4** (2004) 84-88

Non-negative Matrix Factorizations Based Spontaneous Electroencephalographic Signals Classification Using Back Propagation Feedback Neural Networks

Mingyu Liu, Jue Wang, and Chongxun Zheng

The Key Laboratory of Biomedical Information Engineering of Ministry of Education
Xi'an Jiaotong University, Xi'an, Shannxi 710049,China

Abstract. The paper proposes a new spontaneous EEG classification method for attention-related tasks. The algorithm was based on back propagation feedback neural network. Non-Negative Matrix Factorization (NMF) was used as a feature extraction tool. Six electrodes were selected from 32 international 10-20 electrode placement systems according to surface power distributing of EEG activity. Several experiments were carried out to decide an adaptive and robust structure of BP-ANN. The final structure of the NMF-ANN preserved the spatio-temporal characteristics of the signal. Simulation results showed that the averaged classification accuracy for designed three-level tasks can reach 98.4%, 86%, and 82.8%, which were better than other two reference methods.

1 Introduction

Electroencephalogram (EEG) is used not only as a tool to investigate the electric activity of the brain, but also as a mode of communication between the man and his surroundings. Existing research in Brain Computer Interface includes two main areas. The first area is concerned with natural changes of EEG signal. The second area aims at investigating how to train the subject to consciously alter his EEG signal. Such a process forms the basis of EEG-based biofeedback training, which is employed to treat certain kinds of disorders and disease such as Attention Deficit Hyperactivity Disorder [1],[2]. Fourier spectrum of EEG is widely used as the controlling parameter. The advance of neural network theory opens a new window to distinguish different mental tasks from raw EEG. Here we designed a new system capable of efficiently learning a particular mapping between EEG features and different attention-level mental tasks. Non-negative matrix factorization was used as feature extraction tool. Back propagation ANN was employed as classifier. Certain attention-related mental tasks were designed in hopes of producing measurably different EEG features.

2 Methodology

The EEG data were collected by a NeuroScan SynAmp2 system. The bandpass analog filters were set at 0.1-70Hz and free of 50Hz current disturbance. EEG signals were sampled 256 times per second with 32 bits of accuracy. By SynAmp2 we removed the artificial caused by EMG, ECG, EOG manually. There were 32 electrodes located on the scalp according to the standard “10-20 system” placement. Data from all of the electrodes were recorded for 10 seconds during each task. Each task was repeated for 10 times per session, in which the least 5 contamination ones was selected. Thus we got 256×10×5×32 points per session. At last the 50s session EEG data was divided into 50 segments. Six subjects (four male and two female) attended the sessions.

2.1 Experiment Protocol

To fit the real conditions in which the ADD/ADHD sufferer could work, we designed an experiment to detect the changes in EEG of different attention levels. Following is the description of the tasks performed by each subject. To minimize the movement artifacts, the subjects were seated in a pacific room and asked not to move his/her body during the experiments.

Task 1 - Baseline Measurement
The subject was asked to perform no mental task. Just kept relaxing and thinking of nothing. This was done with both eyes opened and closed.

Task 2 - Attention State
The subject was sited facing with a computer screen. A Visual Game displayed on Screen, which run automatically and smoothly. The subject should pay attention to the game and try to imagine he/she was controlling the game. Fig.1 shows such a visual game. A fingerling is chasing the pearl under the sea ceaselessly.

Task 3 - Inattention State
The same visual game was presented on screen, but kept frozen. The subject was asked to keep the sight on the screen and think of other things or just think of nothing.

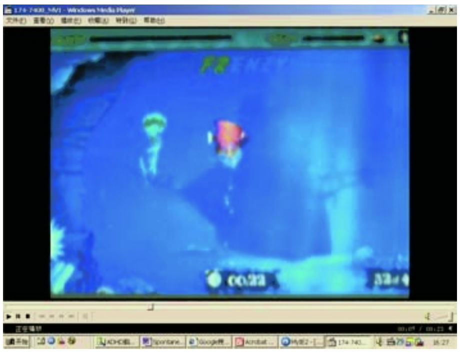


Fig. 1. Example of one pearl-chasing game for the attention-level task.

To reduce the EEG channels acquired in classification and do not depress the accuracy, we computed the spectrum maps of all subjects undertaking designed experiment. It was founded that for the EEG components (SMR, Theta and Beta1) of most subjects, the power was mainly focus on frontal lobe and occipital lobe. Considering the frontal lobe which was mainly related to attention, C3, C4, F4, F8, P3, P4 was selected, thus the actual number of EEG Channels was 6.

2.2 Non-negative Matrix Factorization (NMF) as Feature Extraction Tool

NMF is a subspace method that finds a linear data representation in non-negativity constraint [3]. Suppose that N observed data points, $\{v_t\}$, t = 1,…, N. Denote the data matrix by V = {v1 … v_M}. On this case N is the sampling rate (256) and M is the number of channels (6). Then we construct approximate factorizations of the form V=WH as formula.1 shows. W is called basis matrix while H is called encoding matrix. An encoding consists of the coefficients by which a vector set is represented with a linear combination of basis vectors. The dimensions of the matrix factors W and H are n×r and r×m, respectively. The rank r of the factorization is generally chosen so that (n+m)r < nm, and the product WH can be regarded as a compressed form of the data in V. Since NMF is an approximation factorization, we need to define the cost function to qualify this approximation. One natural way is to use the Euclidean distance between V and WH, another is to use the divergence of V and WH, as formula 2 and 3 shows separately. Thus the problem turns to minimize ||V-WH|| or D(V||WH) with respect to W and H, subject to the constraints W,H>0.

$$V_{i\mu} \approx (WH)_{i\mu} = \sum_{a=1}^{r} W_{ia} H_{a\mu} \tag{1}$$

$$\| V - WH \| = \sum_{i,j} (V_{ij} - (WH)_{ij})^2 \tag{2}$$

$$D(V \| WH) = \sum_{i,j} (V_{ij} \log \frac{V_{ij}}{(WH)_{ij}} - V_{ij} + (WH)_{ij}) \tag{3}$$

The procedure of iterative updates of W and H can be described as follows:

1. Initialize W and H to be two random non-negative matrices;
2. Keep updating W and H until the cost function converges. We should update W and H simultaneously. That means, after updating one row of W, we need to update the corresponding column of H. The multiplicative update rules are as the formula 4 shows. The calculated NMF basic vectors represent the so called feature-vector and are used as inputs to the neural network classifier.

$$W_{ia} \leftarrow W_{ia} \sum_{\mu} \frac{V_{i\mu}}{(WH)_{i\mu}} H_{a\mu} \qquad H_{a\mu} \leftarrow H_{au} \sum_{i} W_{ia} \frac{V_{i\mu}}{(WH)_{i\mu}}$$
$$W_{ia} \leftarrow \frac{W_{ia}}{\sum_j W_{ja}} \tag{4}$$

2.3 BP-ANN Classifier

The basic idea of ANN Classifier is that, during the training, units are pulled toward the EEG features of the attention-level task they represent and pushed away from other tasks. In this way the discrimination of mental tasks by means of the EEG signals is changed into classification of a system that has as output the EEG signals. The schematic block diagram for the final proposed NMF-ANN is shown in Fig.2.

A multiplayer perception ANN with one hidden layer was used to be the classifier. The number of network inputs varied from 3 to 256 depending on the actual length of feature vector. The output nodes were set at 3 according to three mental tasks: (1,0,0) for the baseline task (Task1), (0,1,0) for the attention task (Task 2), (0,1,0) for the in-attention task (Task3). If the classifier identified the task correctly, we named it as "Hit", otherwise named it as "Miss". The hidden layer nodes were varied from 10 to 50, which were decided by successive experiments. It was not true that more hidden units could always lead to lower error. We should balance between the number of hidden units and training times. Basing on these facts and considering classification accuracy together with performance, the final value of hidden units was 20.

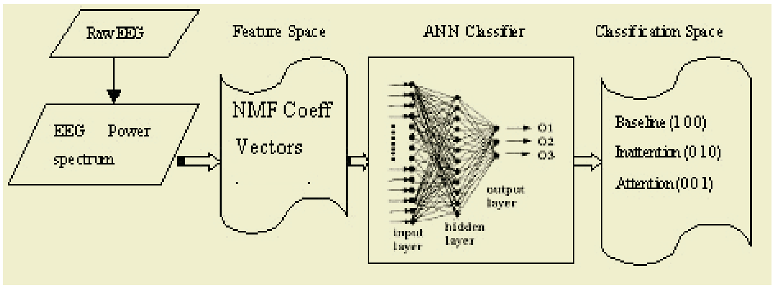


Fig. 2. Schematic diagram of our EEG classification system, including NMF feature extraction scheme and ANN classification scheme.

Then we explored different types of training algorithms, in which a back propagation algorithm [4] was adopted in our work at last. The training was initiated by weights of small and random values, and then performed a gradient-descendent search for a minimum of the squared error function of the output of the network. To measure the performance of the network, we used cross-validation method: the feature set was spitted to training set and validation setting in a proportion of 60% to 40%. The training was stopped when the mean square error was almost zero and the cross-validation error reached a minimum. Furthermore, if the cross-validation errors began to rise and the performance was good, the training must be stopped ahead. The later rule was to avoid over-training of the network.

3 Results and Discussion

In this section we present the results of classification experiment mentioned above. Fig.3 a shows the EEG data of the subject 3 (male, 24 aged, right-handed) from selected six channels, while Fig.4 b shows the corresponding SMR spectrum. It can be seen that the spectrum matrix of SMR is composed of high-dimension feature vectors (256×6). It is thought be larger than actually needed, while the EEG matrix has the same condition. So we used NMF to reduce the redundancy of EEG matrix. SMR spectrum and Principal Components Analysis (PCA) encoding variable vectors of EEG data were used as another two feature extraction methods as comparison.

Unlike other matrix factorization methods such as PCA and vector quantization (VQ), NMF allowed only additive combinations, thus the coefficient vectors showed more uniform distribution. It was compatible with the intuitive notion of combining parts to form a whole. Fig.4 gives the basis vectors and corresponding encoding matrixes of EEG by using NMF extraction methods. It shows uniform distribution.

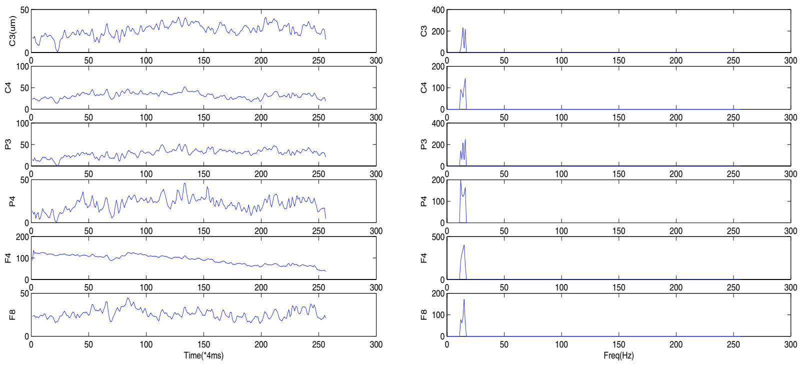


Fig. 3. One second EEG data of subject 3: a. Raw EEG data; b. SMR Spectrum.

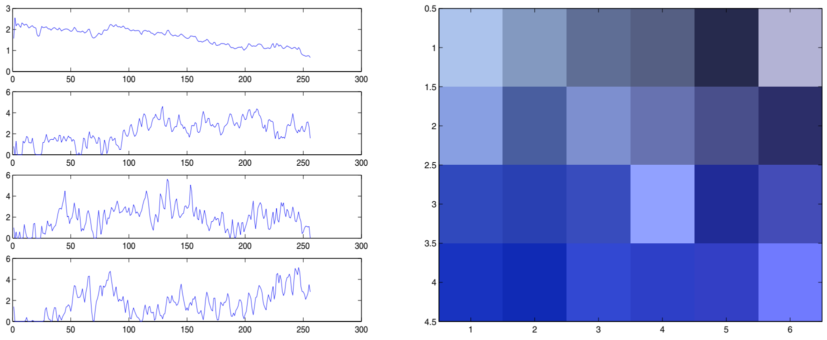


Fig. 4. A NMF basis vectors; b NMF coefficient vectors; both a and b were learned from EEG matrix of a male, 24 aged, right-handed subject. The coefficient vectors showed uniform distribution.

Table 1. Classification Accuracy Rates of Task 2 Based on Different Feature Extraction Methods.

The number of Hit (Feature Extraction Methods)			
	SMR	PCA	NMF
Subject1	40/10	41/11	43/7
Subject2	38/12	39/22	38/12
Subject3	41/9	44/6	44/6
Subject4	41/9	43/7	45/5
Subject5	42/8	44/6	45/5
Averaged	40.4/9.6	42.2/7.8	43/7

Table 1 reports the classification results of task 2 using a BP-ANN with 10 single hidden units. For the direct SMR input, the Classifier recognized 38(min)-42(max) segments correctly out of the 50 segments, which mean 'hit' 38~42 tasks. As for PCA and NMF features, 39-44 and 38-45 tasks were 'hit' out of the 50 segments. Individually, Subject 2 has the worst results over all representations. Subject 1, 3 and 4 performed better for NMF basis vectors, while PCA gave better results for subject 5. The results show that the NMF features have better performance for most of the subjects.

This fact was expected taking into account the non-negative constraint is suitable for EEG signal feature extraction. We repeated the computation using this NMF-ANN configuration algorithm and give the result in table 2.

Table 2. Average Classification Results using NMF Features and BP-ANN Classifier.

	The number of Hit		
	Task1	Task2	Task 3
Subject1	48	43	42
Subject2	50	38	39
Subject3	50	44	41
Subject4	49	45	42
Subject5	48	45	43
Accuracy	98.4%	86%	82.8%

4 Conclusions

The purpose of the paper was to investigate an efficient model for the extraction and classification of spontaneous EEG during different attention-level mental tasks. We repeated an experiment using different kinds of feature representations. It was found that the non-negative matrix factorization algorithm performs better than other two methods. Tests showed that for all the subjects, the method could be used to discriminate three mental tasks designed for different attention-level with accuracy more than 92.5%. This result was quite acceptable since there were three tasks instead of normally two tasks for EEG-based biofeedback system. Furthermore, to arrive a definite ANN solution, we tried different numbers of hidden nodes. By comparing the training times and the number of iterations with cross-validation method, we designed an ANN configuration suitable for our system.

Acknowledgements

This work is supported by the National Science Foundation under Grant No. 60271025

References

1. Shawn, C., Green, Bavelier, D.: Action Video Game Modifies Visual Selective Attention. Nature, **423** (2003) 534-537
2. Adam, R., Clarke, Robert, J., Barry, McCarthy, R.: EEG Analysis of Children with Attention-Deficit/Hyperactivity Disorder and Co. Journal of Learning Disabilities, May/Jun,**35** (2002) 3
3. Lee, D.D., Seung, H.S.: Learning the Parts of Objects by Non-Negative Matrix Factorization, Nature, **401** (1999) 788-791
4. Haykin, S.: Neural Networks-A comprehensive Foundation, 2nd edition. Prentice-Hall Inc., New Jersey (1999)

Neural Networks Preprocessing Based Adaptive Latency Change Estimation of Evoked Potentials*

Yongmei Sun[1,2], Tianshuang Qiu[1], Wenhong Liu[1], Wenqiang Guo[1], and Hui Li[2]

[1] Department of Electronic and Information Engineering, Dalian University of Technology, Dalian 116024, China
[2] Department of Electronic Information, Dalian Jiaotong University, Dalian 116028, China
sunym1118@sina.com

Abstract. Based on the nonlinear processing ability of neural networks, a new method of estimation of latency change in evoked potentials (EPs) is proposed in this paper. Neural networks are utilized as filters before DLMS algorithm in EP latency change estimation in order to suppressing impulsive background noises. The new latency change estimation method shows robust performance under non-Gaussian α-stable noise conditions.

1 Introduction

Evoked potentials (EPs) are responses of the central nervous system when it is stimulated by external light, sound and electricity. Latency is defined as the time interval between the onset of a controlled stimulus and a selected peak in the EP signal. The detection and quantification of latency changes can contribute to the detection and identification of possible brain injury [1],[2],[3].

In general, EP signals are always contaminated by ongoing electroencephalogram (EEG) signals which are considered noises in EP analysis. Traditional techniques used for detecting and estimating latency changes in EPs, such as the direct least mean square (DLMS) algorithm, are usually based on the assumption that the EEG and other ongoing noises in EPs are Gaussian distributed. However, the EEG and other ongoing noises in EP signals are found to be non-Gaussian fractional lower order α-stable process (FLOA) [2],[3],[4], which is significantly different from Gaussian process by its thick tail in its distribution and spikes in its waveform. It is known that, for a non-Gaussian stable process with characteristic exponent α ($0 < \alpha < 2$), only moment of order less than α is finite. So the performances of the second order moment based EP latency change detection algorithms degenerate seriously under such kind of noise conditions [2],[3].

Neural networks have shown good capabilities for non-linear mapping, self-learning and self-adjusting. The study of neural network is of special interest in many applications, such as classifying and filtering [5],[6],[7]. This paper uses a neural network as a non-linear preprocessor in order to suppress the spikes in noisy EP signals. It is verified that the output of neural network has finite second order moment,

* This work is supported by National Science Foundation of China under Grants 30170259, 60372081, and the Science Foundation of Liaoning Province under Grant 2001101057.

so traditional second order moment based algorithm (DLMS) can be adopted to the latency change estimation of EP signals. Simulation results show that the neural networks preprocessing based method of the latency change estimation in EPs is effective under α -stable noise conditions.

2 Models of EP Signals and Noises

The signal model for latency change estimation of EP signal is defined as [1]

$$\begin{aligned} x_1(k) &= s(k) + v_1(k), \qquad k = 0,1,\dots,K-1; \\ x_2(k) &= s(k-D) + v_2(k), \qquad k = 0,1,\dots,K-1; \end{aligned} \tag{1}$$

where $x_1(k)$ and $x_2(k)$ denote the reference and ongoing EP signals, $s(k)$ denotes the noise free signal, $s(k-D)$ is a delayed version of $s(k)$, and D is the latency change to be estimated. $v_1(k)$ and $v_2(k)$ denote the background EEG and other noises in EP signals and k is the discrete time index. In EP study, $x_1(k)$ is normally obtained by averaging many sweeps of EP so the noise $v_1(k)$ can be negligible (if the sweep number participating in the average is large sufficiently) [1].

The α -stable distribution process is an important class of statistical processes. A unique property of any α -stable process is that it is the only distribution that can be the limit distribution for sums of independently identically distributed random variables [4]. The α -stable distribution can be conveniently described by its characteristic function [4]:

$$\phi(t) = \exp\{j\mu t - \gamma |t|^{\alpha} [1 + j\beta \operatorname{sgn}(t)\omega(t,\alpha)]\}, \quad \omega(t,\alpha) = \begin{cases} \tan\dfrac{\pi\alpha}{2}, & \alpha \neq 1 \\ \dfrac{2}{\pi}\log|t|, & \alpha = 1 \end{cases} \tag{2}$$

where $\alpha \in (0,2]$ is the characteristic exponent, which measures the thickness of the tail in the distribution; γ is the dispersion parameter which is similar to the variance of Gaussian process, and β is the symmetry parameter. If $\beta = 0$, the distribution is symmetric and the observation is referred to as a $S\alpha S$; μ is the location parameter $\operatorname{sgn}(\cdot)$ is the sign function. When $\alpha = 2$, Equation (2) is same as that of Gaussian distribution, which means that the Gaussian distribution is the special case of the α -stable distribution. The $S\alpha S$ when $0 < \alpha < 2$, referred to as the FLOA distribution, is significantly different from Gaussian process by its thick tail in its distribution and spikes in its waveform although it maintains some properties of the latter [4].

3 Latency Change Estimation Based on Neural Network Preprocessing

3.1 DLMS Algorithm

DLMS algorithm is an adaptive time delay estimation method based on LMS criteria proposed by Etter et al., Kong et al introduced this method in the estimation of latency change in EPs [3]. The cost function of DLMS algorithm is

$$J = \mathrm{E}[e^2(k)] = 2\sigma_s{}^2 + \mathrm{E}[v_2{}^2(k)] - 2R_{ss}(D - \hat{D}) \tag{3}$$

where $e(k) = x_2(k) - x_1(k - \hat{D})$ is the error function, $\hat{D}$ is the estimation of D, $\sigma_s{}^2$ represents variance of $s(k)$, $R_{ss}(\cdot)$ is the auto-correlation of $s(k)$. Under Gaussian noise conditions, we have $\mathrm{E}[v_2{}^2(k)] = \sigma_v{}^2$, where $\sigma_v{}^2$ is the variance of noise. By using the gradient technique we get the adaptive iterative equation as

$$\hat{D}(k+1) = \hat{D}(k) + \mu e(k)[x_1(k - \hat{D} - 1) - x_1(k - \hat{D} + 1)] \tag{4}$$

DLMS algorithm works well when the background noises are Gaussian distributed. But it fails to work or does not perform optimally when the background noises are fractional lower order α -stable distributed. When the additive noise $v_2(k)$ in equation (3) is an α -stable distribution process, it doesn't have finite moment of order greater than α ($0 < \alpha < 2$), so $\mathrm{E}[v_2{}^2(k)] \to \infty$. This leads to $J \to \infty$.

3.2 Neural Network Preprocessing

Neural network is a kind of calculative technique which can realize the complex nonlinear mappings. In this paper, we design a neural network filter consisting of a feedforward multi-layer-perceptron (MLP) with one hidden layer (Fig.1). The transfer function of the hidden layer and output layer is the sigmoid function and linear function respectively.

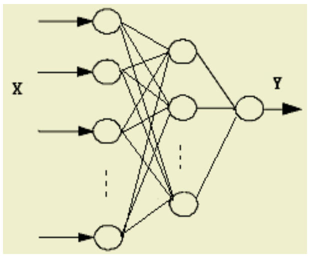


Fig. 1. Topology of the neural network.

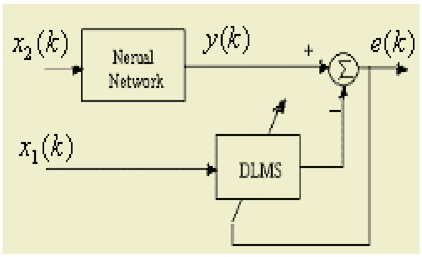


Fig. 2. Scheme of the propose architecture using neural network preprocessing.

The learning of the MLP is obtained by the Back-Propagation (BP) algorithm. The input signal is a set of noisy EP signals and the target signal is the noise free EPs. The network is trained by iteratively presenting the EP ensembles of the available set. Each ensemble consists of N samples which are successively fed into the network through a sliding window wide as the number of input nodes. After the training phase, the simulation output signals are a filtered version of the input noisy EP signals:

$$y(k) \approx s(k - D) \tag{5}$$

which means $v_2(k) \to 0$. So the cost function

$$J = \mathrm{E}[e^2(k)] \approx 2\sigma_s{}^2 - 2R_{ss}(D - \hat{D}) \tag{6}$$

becomes finite.

It is clear that the problem of the degeneration of DLMS algorithm is solved by neural network effectively. Thus, after the preprocessing of the neural network, DLMS algorithm can be used to estimate the latency change of EPs. The scheme of the proposed architecture is shown in Fig. 2.

4 Computer Simulation

Computer simulation is conducted to verify the robustness of the new method to estimate the EP latency changes under the FLOA noise conditions. Signals and noises are constructed as (1). The latency changes are set as follows:

$$D_n = \begin{cases} 0, & 1 \le n \le 100 \\ 10T_s & 101 \le n \le 200 \\ 10T_s(400-n)/200, & 201 \le n \le 400 \\ 0, & 401 \le n \le 500 \end{cases} \tag{7}$$

where T_s is the sampling interval of the EP signals and n denotes the sweep index. The mixed SNRs (MSNR) [1] is set based on $\mathrm{MSNR} = 10\log_{10}(\sigma_s^2/\gamma_v)$, in which σ_s^2 is the variance of $s(k)$ and γ_v is the dispersion parameter of the FLOA noises. Set $\mathrm{MSNR} = -5$ and $\alpha = 1.5$.

The neural network has 21 linear input nodes, 10 sigmoid hidden nodes and 1 single linear output node.

Fig.3 shows the comparison of input noisy EP signal and neural network output signal. Fig.3a is the purified EP signal, Fig.3b is the noisy EP signal contaminated by α-stable noises, Fig.3c shows the output signals of the neural network. It is clear that the neural network suppresses the contaminated α-stable noises successfully as a prefilter.

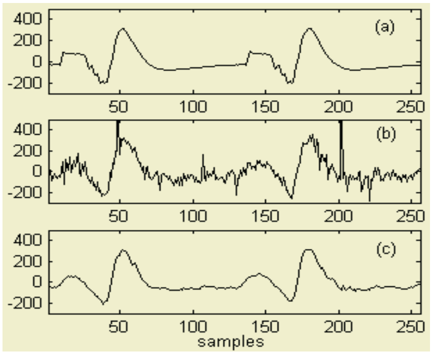


Fig. 3. Comparison of input noisy EP and neural network output.

Fig.4 shows the latency change estimation using DLMS algorithm and neural network preprocessing method under α-stable noise conditions. The simulation results show that the DLMS degenerate obviously under α-stable noise conditions (Fig.4a), but the neural network preprocess method perform well (Fig.4b). The estimation error power is 434.49 for the DLMS algorithm and 0.30 for the neural network preprocessing based method, respectively.

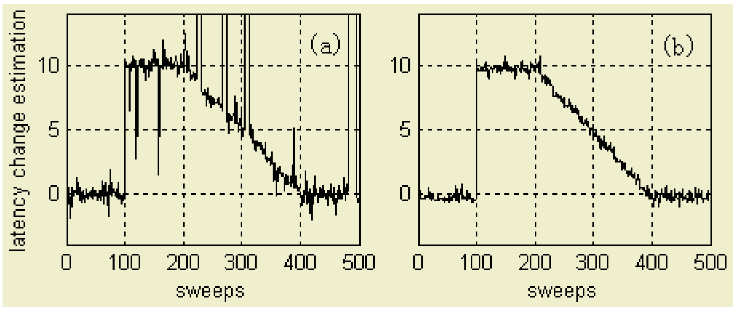


Fig. 4. Latency change estimations under α-stable noises ($\alpha = 1.5$).

5 Conclusions

Noises in EPs are non-Gaussian fractional lower order α-stable process. Traditional second order moment based latency change estimation algorithm degenerate seriously under such kind of noise conditions. The neural network is used in present paper as a nonlinear preprocessor before DLMS algorithm in the estimation of latency change of EP signals. Simulation results confirm the successful filter of neural network under α-stable noises. The neural network preprocessing based method of the estimation of EPs is effective under α-stable noise environment.

References

1. Kong, X., Qiu, T.: Adaptive Estimation of Latency Change in Evoked Potentials by Direct Least Mean P-norm Time Delay Estimation, IEEE Trans. on Biomedical Engineering, 46 (1999) 994-1003
2. Kong, X., Qiu, T.: Latency Change Estimation for Evoked Potentials: a Comparison of Algorithms, Medical & Biomedical Engineering & Computing, 39 (2001) 208-224
3. Kong, X., Thakor, N.V.: Adaptive Estimation of Latency Changes in Evoked Potentials, IEEE Trans. on Biomedical Engineering, 43 (1996) 189-197
4. Nikias, C. L., Shao, M.: Signal Processing with Alpha-Stable Distribution and Applications, New York: John Wiley & Sons Inc (1995)
5. Houya, T., Kamata, H., Ishida, Y.: Design of an Adaptive FIR Filter Using Symmetric Neural Networks (1993) 96-99
6. Barro, S., Fernandez-Delgado, M., Vila-Sobrino, J.A., et al.: Classifying Multichannel ECG Patterns with an Adaptive Neural Network, IEEE Engineering in Medicine and Biology Magazine, 17 (1998) 45- 55
7. Fung, K. S. M., Lam, F. K., Chan, F. H. Y. et al; Adaptive Neural Network Filter for Visual Evoked Potential Estimation, IEEE international conference on neural network, 5 (1995) 2293-2296

Blind Estimation of Evoked Potentials Based on Fractional Lower Order Statistics

Daifeng Zha, Tianshuang Qiu, and Xiaobing Li

School of Electronic and Information Engineering, Dalian University of Technology, Dalian, Liaoning 116024, China
zhadaifeng@163.com, qiutsh@dlut.edu.cn

Abstract. Traditional evoked potentials (EPs) analysis is developed under the condition that the background noises in EP are Gaussian distributed. Alpha stable distribution is better for modeling impulsive noises than Gaussian distribution. Conventional blind estimation method of EPs is based on second order statistics (SOS) and MMSE criterion. We propose a new algorithm based on minimum dispersion criterion. The simulation experiments show that the proposed new algorithm is more robust than the conventional algorithm.

1 Introduction

The brain EPs are electrical responses of the central nervous system to sensory stimuli applied in a controlled manner. The EPs have a number of clinical applications including critical care, operating room monitoring and the diagnosis of a variety of neurological disorders [1],[2]. The goal in the analysis of EPs is currently the estimation from the several potentials, or even from a single potential. Independent component analysis (ICA) appeared as a promising technique in signal processing. Conventional ICA [3] is optimal in approximating the input data in the mean-square error sense. The data are represented in an orthogonal basis determined merely by the SOS (covariance) of the input data [4]. Recent studies [5],[6] show that alpha stable distributions is better for modeling impulsive noise than Gaussian distribution. In general, EP signals are always accompanied by ongoing electroencephalogram (EEG) signals which are considered noises in EP analysis. Often the EEG signals are assumed to be Gaussian white noise for mathematical convenience. However, the EEG signals are found to be non-Gaussian in other studies [8]. An analysis shows that the alpha stable model fits the noises found in the impact acceleration experiment under study better than the Gaussian model [9]. The typical Evoked potentials are shown in Fig.1. Consequently, EP analysis algorithms developed under the Gaussian EEG assumption may fail or may not perform optimally.

Alpha stable distribution process has no its second order or higher order statistics. It has no close form probability density function so that we can only describe it by its characteristic function:

$$\varphi(t) = \exp\{ j\xi t - \delta|t|^{\alpha}[1 + j\beta \operatorname{sgn}(t)\omega(t,\alpha)]\} \ , -\infty < \xi < \infty, \delta > 0, 0 < \alpha \le 2, -1 \le \beta \le 1$$
$$\omega(t,\alpha) = \tan\frac{\alpha\pi}{2}(if\ \alpha \ne 1) \quad or \quad \frac{2}{\pi}\log|t|(if\ \alpha = 1) \quad) \tag{1}$$

The exponent α determines the shape of the distribution. The dispersion δ plays a role analogous to the variance. β is the symmetry parameter and ξ is the location parameter. Due to the thick tails, alpha stable processes do not have finite second or higher-order moments. This feature may lead all second order moment based algorithms to fail or to work sub-optimally.

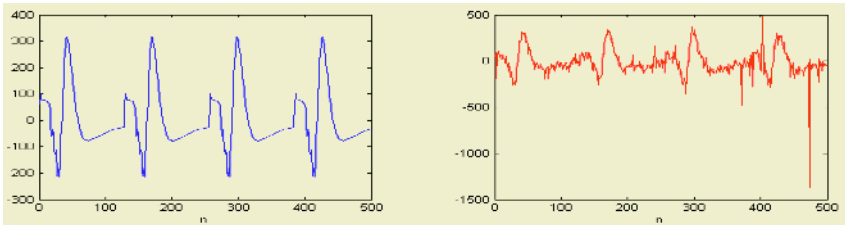


Fig. 1. Evoked potentials (Left: without noise; Right: with noise).

2 Data and System Model

We assume that P signals $s_i(n), i=1,2,...P$ are non-coherent, statistically independent. The noiseless linear ICA model with instantaneous mixing may be described by the equation $\mathbf{x}(n)=\mathbf{A}\mathbf{s}(n)=\sum_{i=1}^{P}\mathbf{a}_i s_i(n)$, where $\mathbf{x}(n)=[x_1(n),x_2(n),....,x_M(n)]^T$, $n=1,2,...N, M\geq P$ (T denoting the transpose) are observed signals, $\mathbf{S(n)}=[s_1(n),s_2(n),\cdots,s_P(n)]^T$ are the source signals containing alpha stable distribution signals or noises which are supposed to be stationary and independent, and **A** is an unknown mixing matrix. Data and system model is shown in Fig. 2. Our goal is to estimate $\mathbf{S}(n)$ from $\mathbf{X}(n)$, with appropriate assumptions on the statistical properties of the source distributions. The solution is $\mathbf{Y}(n)=\mathbf{W}\mathbf{Z}(n)$, where $\mathbf{W}$ is called the de-mixing matrix. The general ICA problem requires **A** to be a matrix of full column rank, with $M\geq P$. In this paper, we assume an equal number of sources and sensors to make problem simple.

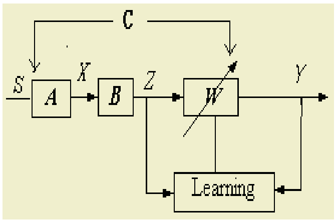


Fig. 2. Data and system model.

3 Whitening by Normalized Covariance Matrix

We introduce a two-step separation method of EPs. The first step is a whitening procedure that orthogonalizes the mixing matrix. Here we search for a matrix **B** which transforms mixing matrix $\boldsymbol{A}$ into a unitary matrix. Classically, the whitening matrix is computed as the inverse square root of the signal covariance matrix. In our case, impulsive EEG noises have infinite variances. However, we can take advantage from the normalized covariance matrix.

Theorem 1. [7]: Let $\mathbf{X} = [\mathbf{x}(1), \mathbf{x}(2), ..., \mathbf{x}(N)]$ be a stable process vectors data matrix, then normalized covariance $\mathbf{\Gamma}_x = \mathbf{X}\mathbf{X}^{\mathrm{T}} / \{N \cdot Trace(\mathbf{X}\mathbf{X}^{\mathrm{T}} / N)\}$ matrix of $\mathbf{X}$ converges asymptotically to the finite matrix when $N \to \infty$, i.e., $\lim_{N\to\infty} \mathbf{\Gamma}_x = \mathbf{A}\mathbf{D}\mathbf{A}^{\mathrm{T}}$, $\mathbf{D} = diag(d_1, d_2, ..., d_M)$, where $d_i = \lim_{N\to\infty} \Delta_i \Big/ \sum_{j=1}^{P} \Delta_j \left\|\mathbf{a}_j\right\|^2$, $\mathbf{a}_j$ is column of $\mathbf{A}$, $\Delta_i = \sum_{n=1}^{N} x_i^2(n) / N$.

Theorem 2: We have eigendecomposition of $\mathbf{\Gamma}_x$ as $\mathbf{\Gamma}_x = \mathbf{U}\mathbf{\Omega}^2\mathbf{U}^{\mathrm{T}}$ and we can obtain whitening matrix $\mathbf{B} = \mathbf{\Omega}^{-1}\mathbf{U}^{\mathrm{T}}$, then $\mathbf{Z} = \mathbf{B}\mathbf{X}$ is orthogonal.

Proof: $\mathbf{Z}\mathbf{Z}^{\mathrm{T}} = \mathbf{B}\mathbf{X}\mathbf{X}^{\mathrm{T}}\mathbf{B}^{\mathrm{T}} = \mathbf{B}\mathbf{\Gamma}_x N \cdot Trace(\mathbf{X}\mathbf{X}^{\mathrm{T}} / N)\mathbf{B}^{\mathrm{T}}$

$$= N \cdot Trace(\mathbf{X}\mathbf{X}^{\mathrm{T}} / N) \cdot \mathbf{B} \cdot \mathbf{U}\mathbf{\Omega}^2\mathbf{U}^{\mathrm{T}} \cdot \mathbf{B}^{\mathrm{T}}$$

$$= N \cdot Trace(\mathbf{X}\mathbf{X}^{\mathrm{T}} / N) \cdot \mathbf{\Omega}^{-1} \cdot \mathbf{I} \cdot \mathbf{\Omega}^2 \cdot \mathbf{I} \cdot \cdot (\mathbf{\Omega}^{-1})^{\mathrm{T}} = N \cdot Trace(\mathbf{X}\mathbf{X}^{\mathrm{T}} / N) \cdot \mathbf{I}$$

So we can write $\mathbf{B}\mathbf{\Gamma}_x\mathbf{B}^{\mathrm{T}} = \mathbf{B}A\mathbf{D}\mathbf{A}^{\mathrm{T}}\mathbf{B}^{\mathrm{T}} = (\mathbf{B}A\mathbf{D}^{1/2})(\mathbf{B}A\mathbf{D}^{1/2})^{\mathrm{T}} = \mathbf{I}$ and thus the whitening of $\mathbf{x}(n)$ is performed through $\mathbf{z}(n) = \mathbf{B}\mathbf{x}(n)$.

4 De-mixing Algorithm

The minimum dispersion (MD) criterion is used in discussing linear theory of stable processes [5], [6]. The MD criterion is thus a direct generalization of the MMSE criterion in the Gaussian case. Thus, by minimizing the dispersion we minimize the average magnitude of estimation errors. Interestingly, the MD criterion can be used as a cost function to achieve the estimation of evoked potentials.

Let $\mathbf{y}(n) = \mathbf{W}\mathbf{z}(n)$, $\mathbf{W}$ is a 'separating' matrix to be estimated and $\mathbf{W}$ is unitary. Let $\mathbf{C} = \mathbf{W}\mathbf{B}\mathbf{A}$ then $\mathbf{y}(n) = \mathbf{W}\mathbf{B}\mathbf{A}\mathbf{s}(n) = \mathbf{C}\mathbf{s}(n)$. $\mathbf{y}(n)$ is an orthogonal mixture of the sources and $\mathbf{C}$ is orthogonal . We consider the global minimum dispersion optimization given by:

$$\mathbf{W}^* = \arg\min_{\mathbf{W}} J(\mathbf{W}) = \arg\min_{\mathbf{W}} \sum_{i=1}^{P} \gamma_{y_i} \ (\mathbf{W} \text{ is unitary}) . \tag{2}$$

γ_{y_i} is dispersion of $\mathbf{y}_i(n)$. As $y_i = \sum_{j=1}^{P} C_{ij} s_j$, we have $\gamma_{y_i} = \sum_{j=1}^{P} | C_{ij} |^{\alpha} \gamma_{S_j}$,(2) can be written as

$$\mathbf{W}^* = \arg\min_{\mathbf{W}} \sum_{i=1}^{P} (\sum_{j=1}^{P} | C_{ij} |^{\alpha} \gamma_{S_j}) = \arg\min_{\mathbf{W}} \sum_{j=1}^{P} (\sum_{i=1}^{P} | C_{ij} |^{\alpha}) \gamma_{S_j} \ . \tag{3}$$

Since $\mathbf{C}$ is unitary (which implies that $| C_{ij} | \le 1$) and $\sum_{i=1}^{P} | C_{ij} |^{\alpha} \ge \sum_{i=1}^{P} | C_{ij} |^2 = 1$, when $C_{ij} = 0$ or $C_{ij} = 1$, we can get $\sum_{i=1}^{P} | C_{ij} |^{\alpha} = 1$ .Therefore we can arrive at the conclusion that when $\mathbf{C}$ is a generalized permutation matrix (i.e., $\mathbf{C}=\mathbf{P\Lambda}$, $\mathbf{P}$ is a permutation matrix and $\mathbf{\Lambda}$ is a non-singular diagonal matrix), $\sum_{i=1}^{P} |C_{ij}|^{\alpha}=1$. That is to say, if $\mathbf{W}$ is a separation matrix, $J(\mathbf{W})$ can reach its minimum point. Since $\mathbf{W}$ is unitary, $\mathbf{W}$ can be chosen as a product of Givens rotation matrix $\mathrm{M}_{(c,s)}$ (which diagonal elements are 1 except for the two elements c in rows i, j and off-diagonal elements are 0 except for the two elements s, $-s$ at (i, j) and (j,i), $c^2 + s^2 = 1$, $c = \cos(\theta)$, $\sin(\theta)$). Givens matrix is $\mathbf{M}(\cos(\theta),\sin(\theta))$, then $\mathbf{W}=\prod\prod_{1\le i<j\le P} \mathbf{M}(\cos(\theta),\sin(\theta))$. The minimization of $J(\mathbf{W})$ is done numerically by searching θ using a grid into $[0,\pi]$. In brief, we propose the following recursive algorithm:

```
1) initialization: $\mathbf{W}=\mathbf{I}$;
2) $k=1$,  $i=1, j=2$;
3) estimating $\gamma_{y_i}$ according to [5], [6];
4) $\theta\in[0,\pi]$, searching $\theta^{*}=\arg\min_{\theta} J(\mathbf{W})$;
5) $\mathbf{Y}(n)=\mathbf{M}(\cos(\theta^{*}),\sin(\theta^{*}))\cdot\mathbf{Z}(n)$;
6) $\mathbf{W}=\mathbf{M}(\cos(\theta^{*}),\sin(\theta^{*}))\cdot\mathbf{W}$;
7) $j=j+1$, goto 3) until $j=P$;
8) $i=i+1$, $j=i+1$, goto 3) until $i=P-1$;
9) $k=k+1$, goto 3) until $\mathbf{W}$ converges.
```

5 Experimental Results

From Section I we know that the noise for EP could be a lower order alpha stable process. Through computer simulations, we will demonstrate the effectiveness of the proposed algorithm under alpha stable noise conditions. We use correlation coefficient $\tau(s_i,y_j)=\left|\sum_{n=1}^{N} s_i(n)y_j(n)\right| \Big/ \sqrt{\sum_{n=1}^{N} s_i^2(n)\sum_{n=1}^{N} y_j^2(n)}$ to evaluate the performances of the proposed algorithm.

Experiment 1: Two independent sources are linearly mixed. One is the periodical noise free EP signal, and the period is 128 points, the sampling frequency is 1000Hz. The other is an alpha stable non-Gaussian EEG noise with $\alpha=1.7$. Two algorithms are used in the experiment, including: (1) SOS based on MMSE criterion (2) FLOS based on MD criterion, respectively. Fig.3 shows the stability and convergency of the algorithms. We can get signals waveforms in time domain shown in Fig.4. For FLOS algorithm, the correlation coefficient of EP signals is 0.9213, and the correlation coefficient of EEG noise is –0.9098.

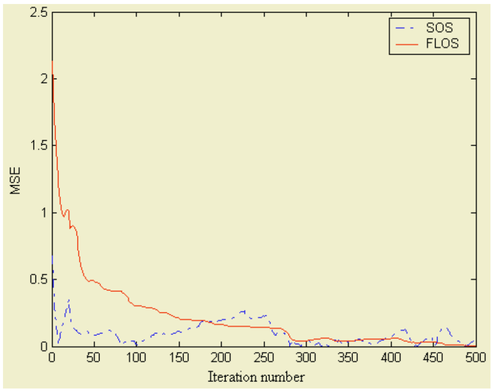


Fig. 3. The stability and convergence performance.

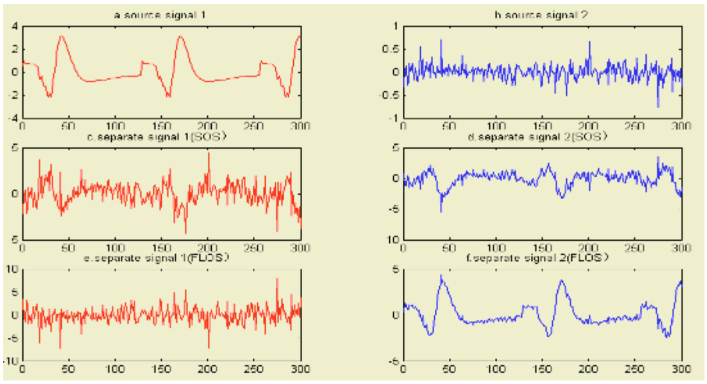


Fig. 4. Separating results: (a)-(b) are the source signals. (c)-(d) are the separated signals with SOS. (e)-(f) are the separated signals with FLOS.

Experiment2: Separate the mixed signals again with the above two algorithms. And the results of 10 independent experiments are shown in Table 1. The correlation coefficients of EP and EEG noise are calculated at some iteration times.

Table 1. Comparison between the two algorithms.

Iteration times	Correlation coefficient (FLOS)		Correlation coefficient (SOS)	
	EP	noise	EP	noise
50	0.1244	0.1044	0.0044	0.0004
100	-0.3450	-0.3050	-0.0050	-0.0063
150	0.4378	0.4378	0.1378	0.1072
200	0.6766	0.7706	0.1716	0.1212
250	-0.9291	-0.9091	-0.1711	-0.1451
300	-0.9287	-0.9107	-0.3937	-0.2231
350	-0.9293	-0.9113	-0.4993	-0.2923
400	0.9295	0.9195	0.3945	0.3045
450	0.9299	0.9292	0.2935	0.1935
500	-0.9501	-0.9593	-0.2804	-0.1904

6 Conclusions

We modify conventional algorithms and analyze the performances of the new algorithm through above simulation. We know the algorithm based on FLOS and MD criterion has better stability and convergency than the algorithm based on SOS and MMSE criterion so that its separation capability is greatly improved.

References

1. Gharieb, R. R., Cichocki, A.: Noise Reduction in Brain Evoked Potentials Based on Third-Order Correlations. IEEE Transactions on Biomedical Engineering, **48** (2001) 501-512
2. Davila, C. E., Srebro, R., Ghaleb, I. A.: Optimal Detection of Visual Evoked Potential. IEEE Transaction on Biomedical Engineering, **45** (1998) 800–803
3. Zhang, Y., Ma, Y.: CGHA for Principal Component Extraction in the Complex Domain. IEEE Transaction on Neural Network, **8** (1997)
4. Mutihac, R.,Van Hulle, M. M.: PCA and ICA Neural Implementations for Source Separation–a Comparative Study. Proceedings of the International Joint Conference on Neural Networks, **1** (2003) 20-24
5. Nikias, C. L., Shao, M.: Signal Processing with Alpha-Stable Distributions and Applications. New York, John Wiley & Sons Inc (1995)
6. Shao, M. and C. L. Nikias: Signal Processing with Fractional Lower Order Moments: Stable Processes and Their Applications. Proceedings of IEEE, **81** (1993) 986-1010
7. Samorodnitsky, G., Taqqu, M. S.: Stable Non-Gaussian Random Process: Stochastic Models with Infinite Variance. New York, Chapman and Hall (1994)
8. Hazarika, N., Tsoi, A. C., Sergejew, A. A.: Nonlinear Considerations in EEG Signal Classification. IEEE Trans. on Signal Processing, **45** (1997) 829–936
9. Kong, X., Qiu, T.: Adaptive Estimation of Latency Change in Evoked Potentials by Direct Least Mean p-Norm Time-Delay Estimation. IEEE Trans. on Biomedical Engineering, **46** (1999)

Wavelet Denoise on MRS Data Based on ICA and PCA

Jian Ma[1], Zengqi Sun[1], Guangbo Dong[1,2], and Guihai Xie[2]

[1] Department of Computer Science, Tsinghua Univ., Beijing100084, China
majian03@mails.tsinghua.edu.cn
{szq-dcs,dgb}@mail.tsinghua.edu.cn
[2] Department of Control System Engineering, Ordnance Engineering College,
Shijiazhuang, Hebei 050003, China

Abstract. In this paper, experiments on previous works of automatic decomposition of MRS based on PCA and ICA were conducted on our small amount of low SNR dataset. New experimental results were derived. Results show that only PCA cannot decomposes MRS into meaningful components when small amount of low SNR data are available and that the denoise ability of PCA is limited and heavily affected result of consequent ICA. A new method combined wavelet with PCA is proposed. Experimental results on the dataset show the improvement presented by wavelet. The new method is promising to further research, such as MRS interpretation and classification.

1 Introduction

Magnetic Resonance Spectra (MRS) has been used in diagnosis with better performance than other techniques, such as CT, PET. However, the difficulty of interpretation on MRS hampered it to be widely used. Many researches have been contributed to address this problem by Pattern Recognition (PR) techniques, especially statistic PR. [1],[2]

Because of high dimensionality of clinical MRS data, feature selection and extraction becomes a necessity for robust analysis of MRS. One direction of feature extraction is extracting components, which contains information corresponding to different metabolites. Statistical Analysis techniques, including Bayesian decomposition,[3] Principal Component Analysis(PCA)[4] and Independent Component Analysis (ICA)[5], was used. PCA projects MRS into subspace coordinated by orthogonal principal components. When dataset is small size and low SNR, noise cannot be fully separated with normal data in such orthogonal subspace. Recently developed ICA is a way of finding a linear non-orthogonal co-ordinate system in any multivariate data. More elaborated review can be found in [6],[7] It has been first applied by Nuzillard to extract the signal components corresponding to different metabolites in MRS.[8] Ladroue applied PCA preceded to ICA to eliminate noise on artificial data and on MRS dataset collected by INTERPRET respectively. ICA is a noise sensitive method. Since the denoise ability of PCA is limited, Independent Components (ICs) extracted from data denoised by PCA are not always good quality. Low SNR real data will frustrate ICA because that noise in data is so complicated that PCA cannot eliminate noise effectively by itself. Thus independent components(IC) extracted by ICA from low SNR data denoised by PCA are also bad quality. To be used in further PR research, ICs quality should be improved.

To obtain a better denoise performance, we designed a new scheme which combined wavelet[9] with PCA to eliminate different order of noise. It had been widely applied to biomedical area [10],[11]. Wavelet represents signal at different frequency level. By setting threshold on wavelet coefficients at certain levels, corresponding frequency noise elements can be filtered. Combined wavelet with PCA, a better IC set will be achieved. We demonstrate our method on real In Vivo MRS data.

This paper is organized as follows: in second section, a introduction on material acquisition is presented. In third section, Experiments and results are presented. Some discussions are contained in fourth section. Conclusions are drawn in fifth section.

2 Material

The study was performed on single-voxel in vivo human hepatic encephalopathy MR spectra dataset. Two types of data acquisition (PRESS and STEAM) were used with an echo time of 35 ms and a repetition time of 1500 ms on a 1.5Tesla GE Medical machine. The dataset contains 64 spectra, including 7 normal cases and 57 abnormal cases. Each data has 512 dimensions, covering the range of [4.2952 - 0.5714] ppm. Dataset is low SNR and the average SNR of is about 6.

3 Experiments

3.1 MRS Decomposition

In the study, experiments are conducted on the above dataset. In the first experiment, PCA proposed by Stoyanova, and ICA were tried to extract components without preprocessing. Since small amount of data are available, little statistical property can be exploited. Thus, the components are not well separated by PCA. PCs are ranked by their corresponding eigenvalues.

Figure 1 shows the first four PCs of 46 PCs which were extracted by a direct PCA from 64 MRS data. Not as Stoyanova mentioned that each metabolite component is represented by one PC, PCA treats the whole range of spectra like a single component. #PC1 is a mean spectra of the dataset and that other PCs represents variations of the dataset. Not as expected, the result did not represent any meaningful bioinformation. A representation of spectrum in a feature space coordinated by interpretable components is expected. Direct ICA derived 47 ICs from the dataset. Most of them cannot be corresponding to biomedical metabolites, information of which has been presented by MRS. The reason is noise contained in the data severely deteriorated the performance of ICA.

In the second experiment, ICA preceded by PCA was used to extract the component. Noise was separated from the desired signal through PCA. PC is ordered by its corresponding eigenvalues. A dataset was reconstructed from first PCs, the sum of eigenvalues of which is bigger than a threshold. FastICA is applied on the new dataset to extract IC components. Since the initial step of FastICA is randomized, the algorithm runs many times. Results show that when the same number of PCs preserved, IC set have almost the same, regardless of order. We reconstructed the original data with the first 10 PCs. Compare with 47 ICs in the result of the first experiment, 15 ICs were extracted, of which the last 3 ICs can be judged as noise components. However,

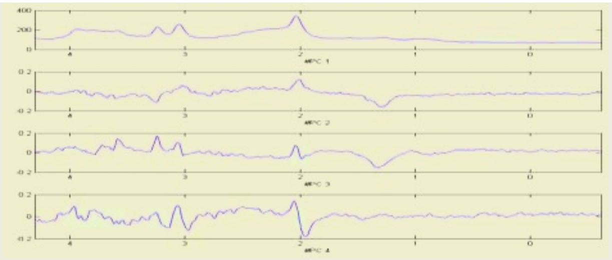


Fig. 1. First four PCs extracted by PCA on the dataset.

some of low SNR ICs may not be interpreted well, because PCA cannot fully eliminate the effect of noise.

To investigate the robustness of ICA extraction, the number of preserved PCs is changed to compare the change between IC sets. Part of Results is listed in Table1. It can be learn from table that the number of IC varies with the number of PCs preserved and SNR of ICs set vary a little. This is because that ICA is sensitive to noise. The More PCs preserved, the more noise remained. Hence the number of ICs increased. However, the average SNR of IC set hardly changed. The new result revealed that the denoise ability of PCA is limited. It can be learned from experiments that when the number of PC varies at 15, most common components of different IC set can be interpreted. Some are illustrated in Figure3. Most of ICs have relatively low SNR.

To improve the performance of ICA feature extraction, wavelet transform was used in the third experiment to filter noise before PCA was used. In order to get more suitable parameter, we tested different type of wavelet. The results show that there is little difference between results when different types of wavelets were chosen. In the experiments, 'db3' wavelet type is chosen; wavelet decomposition is performed at level 3 with soft threshold.

As listed in Table1, The SNR of IC result was improved compared with the result when only PCA applied. In Figure 2, some of improved important ICs are compared with corresponding ICs without wavelet preprocessing. SNR of those ICs is compared in Table2. It can be learn that those improved ICs has well-illustrated shape and thus can be identified by clinicians as certain metabolites or be used in further classification or interpretation.

3.2 Interpretation of ICs and IC Scores

According to standard chemical shift of different molecules, we interpreted the meaning of ICs in term of metabolites. For example, #IC1 contains a peak at 3.2476 ppm, representing Taurine; #IC3 has a peak at 2.048 representing NAAG; the moderate range of peak in #IC4 cover the interval where m-In is; #IC6 represents Choline for its peak at 3.105 ppm and GluX for its peak at 2.095 ppm; the broad peak in #IC7 reflects the big variation in the range of Lactate; #IC8 have three peaks: the big one representing NAA at 1.991 ppm, two others at the interval covered the Creatines at 3.162-2.991 ppm; #IC9 has two peaks closer to each other, representing GluX; #IC12 has a broader peak covered NAA and GluX.

Table 1. IC results when different number of IC preserved. SNR1 is average SNR value of recovered data. SNR2 is average SNR of IC set from recovered data. SNR3 is average SNR of IC set when wavelet is applied.

#PC preserved	#IC	SNR1	SNR2	SNR3
15	14	6.1656	6.5918	7.6574
25	24	6.1430	6.8642	8.1318
35	34	6.1304	6.9516	8.1129
40	39	6.1284	7.0900	7.9661

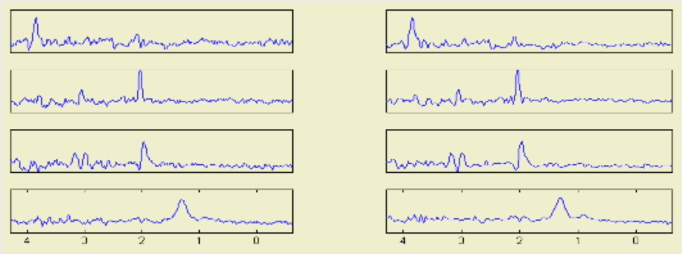


Fig. 2. Comparison on same meaningful ICs from results before (left) and after (right) wavelet is used.

Table 2. Comparison between SNR of ICs before and after wavelet is used.

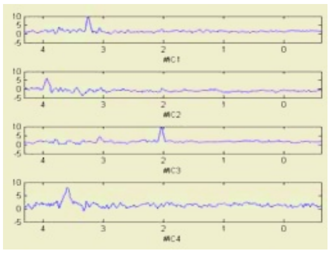
	SNR2	SNR3
Lactate	7.3594	7.8465
NAA	6.5834	7.0517
NAAG	10.0162	11.3430
m-In	7.9081	9.6971

Companied with ICs, a separating matrix W and a mixing matrix A store IC scores of every MRS data. Figure 3(d) illustrates the distribution of ICs scores of the data in terms of ICs in figure 3.a-c. In Figure 3, most of data has similar scores with small variations. The averages of Scores of #IC3, #IC6, #IC8, #IC10, #IC12 are much higher than the others. Referring to the above interpretation of ICs, It can be learn that NAAG, NAA, GluX, Creatine are the main metabolites in the sample. IC corresponding to m-In which is normally high, now is very low can be explained that the density of m-In in those brain tissue samples decreased.

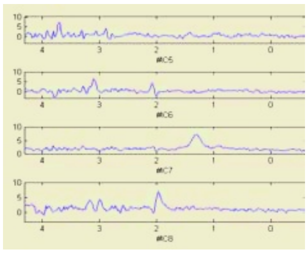
4 Discussions

When dataset are inadequate to reflect the statistical property of the original problem, the decomposition performance of PCA is severely deteriorated, when small amount of data are available and the data has a low SNR, partly because the variation of hidden components is mixed with the variation of high-order noise.. This is because the rank of PCs is also affected by noise.

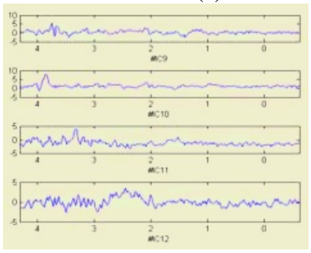
The linear combination of independence model of ICA is more suitable for MRS than the second-order statistical measure of PCA. ICA decomposes data into ICs corresponding biochemical meaning as features. Hence, results of ICA may be appropriate for further classification. Referred to the results before and after wavelet is used, it can learn that ICA has almost no ability to filter noise.



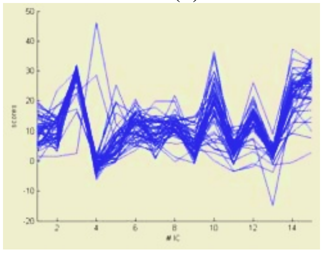
(a)



(b)



(c)



(d)

Fig. 2. ICs and IC scores derived from ICA on MRS data.

Wavelet can improve SNR of dataset without affecting the meaningful components in the dataset. By carefully tuning the type and parameters of wavelet packet, the high-order noise can be getting rid of from the spectral.

IC scores composed of mixing matrix A is the projection of data on feature subspace coordinated by ICs. They give a clear illustration of biochemical composition of MRS. IC scores can be used for classification in two means: one is using scores combined with class labels directly as the input of classifier and then training a classifier for machines recognition. Since the extraction by ICA need not human interaction, ICA can be used into automatic classification. The other is extracting inducing knowledge from scores for clinicians.

5 Conclusions

In this paper, experiments were conducted on small amount of low SNR dataset. Experiments were based on some previous work on automatic decomposition of MRS based on PCA and ICA. Some new experimental results were derived. Analysis on them indicates that ICA with PCA cannot decompose MRS into meaningful components when small amount of low SNR data are available for the denoise ability of PCA is limited. We applied a new method which combined wavelet with PCA. Experimental results show the improvement presented by wavelet. We introduce this method to further research on MRS data analysis.

Acknowledgements

This work was supported by National Science Foundation of China grants 60372042.

References

1. Tate, A.R., et al.: Towards a Method for Automated Classification of 1H MRS Spectra from Brain Tumours. NMR in Biomedicine, **11** (1998) 177-191
2. Hagberg, G.: From Magnetic Resonance Spectroscopy to Classification of Tumors. A Review of Pattern Recognition Methods, NMR in Biomedicine, **11** (1998) 148-156
3. Ochs, M.F., et al.: A New Method for Spectral Decomposition Using a Bilinear Bayesian Approach. Journal of Magnetic Resonance, **137** (1999) 161-176
4. Stoyanova, R., A.C. Kuesel, T.R. Brown: Application of Principal-Component Analysis for NMR Spectral Quantitation. Journal of Magnetic Resonance, Series A, **115** (1995) 265-269
5. Ladroue, C., et al.: Independent component analysis for automated decomposition of in vivo magnetic resonance spectra. Magnetic Resonance in Medicine, **50** (2003) 697-703
6. Amari, S., A. Cichocki, H.H. Yang: A New Learning Algorithm for Blind Source Separation. Advances in Neural Information Processing, MIT Press, Cambridge (1996) 757-763.
7. Hyvörinen, A., E. Oja: Independent Component Analysis: Algorithms and Applications. Neural Networks, **13** (2000) 411-430
8. Nuzillard, D., S. Bourg, J.-M. Nuzillard: Model-Free Analysis of Mixtures by NMR Using Blind Source Separation. Journal of Magnetic Resonance, **133** (1998) 358-363.
9. Mallat, S.G.: A Theory for Multiresolution Signal Decomposition: The Wavelet Representation. Pattern Analysis and Machine Intelligence, IEEE Transactions on, **11** (1998) 674-693
10. Unser, M.A., A.: A Review Of Wavelets in Biomedical Applications. Proceedings of the IEEE, **84** (1996) 626-638
11. Antoine, J.-P., Chauvin, C., Coron, A.: Wavelets and related time-frequency techniques in magnetic resonance spectroscopy. NMR in Biomedicine, **14** (2001) 265-270

Hard Margin SVM for Biomedical Image Segmentation

Chen Pan, Xiangguo Yan, and Chongxun Zheng

Key Laboratory of Biomedical Information Engineering of Education Ministry,
Xi'an Jiaotong University, Xi'an, Shaanxi 710049, China
panchen@mailst.xjtu.edu.cn, cxzheng@mail.xjtu.edu.cn

Abstract. Biomedical image is often complex. Using SVM for pixel-based segmentation may achieve good results, but training by conventional way always leads to high time cost. In this paper, a novel and real-time training strategy is presented. First, the mean-shift procedures are used to find local modes in RGB 3D histogram. Second, pure samples are selected by the divided modes. Third, the training set is constructed by uniform sampling from the pure samples, so its size can be reduced sharply. In the no-niose case, hard margin criterion replaces soft margin criterion for classification. This strategy constructs an unsupervised support vector classifier. Experimental results demonstrate that the new classifier can achieve accurate results, is more robust to change of the color and faster than watershed algorithm. The new method is suitable to segment blood and bone marrow microscopic images.

1 Introduction

Segmentation may be viewed as image classification problem based on color and spatial features. Many classifiers can be adopted to realize segmentation task. SVM is an efficient tool for machine learning tasks involving classification, regression or novelty detection, and recently has been implemented in segmentation of MRI, ultrasonic and hyperspectral remote sensing images [1],[2],[3]. In this paper, we propose a novel parallel method for image segmentation, in which SVM is utilized to avoid the computational complexity caused by the multi-dimensional features and deal with the nonlinearity of data distribution. In order to speed up the training of SVM, we select pure samples corresponding to the clustering modes in histogram by mean-shift, and adopt hard margin criterion[4] for classification. The novel training strategy can sharply reduce the size of the training set and result in simplification of parameters. SVM training can be completed in real-time in our work. Due to excellent performance inherited from the mean-shift and SVM, our segmentation method is an unsupervised learning-classification procedure, which possess some preferred properties such as minimum user interaction, fast computation, and accurate and robust segmentation results.

The rest of the paper is organized as follows. Section 2 introduces the improvement strategy for SVM training. Section 3 presents an unsupervised learning-classification method. Section 4 shows some experimental results and comparisons with watershed algorithm. Conclusion is given in Section 5.

2 Improvement of SVM Training

2.1 Hard Margin and Soft Margin

SVM implements a classification strategy that exploits a margin-based “geometrical” criterion rather than a purely “statistical” criterion. In other words, SVM defines the classification model by exploiting the concept of margin maximization, while does not estimate the statistical distributions of classes for classification. There are two types of margin in SVM [4], hard margin and soft margin corresponding to formula (1) and (2) respectively.

$$\min \frac{1}{2}\|\mathbf{w}\|^2 . \tag{1}$$

$$\min \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_i^n \xi_i . \tag{2}$$

Hard margin classifier works well in no-noise cases, but fails in the noisy data due to overfitting [5]. Soft margin classifier may achieve much better generalization results by relaxing the hard margin and ignoring the noisy data. However, on the one hand, two kinds of support vectors coexist in the soft margin case: (1) margin support vectors that lie on the hyperplane margin and (2) nonmargin support vectors that fall on the “wrong” side of this margin. The latter are regarded as noisy samples that should be ignored. It is well known that reducing the computational cost of the SVM is equivalent to decreasing the number of the support vectors. So removing noisy samples from the training set may benefit to training. On the other hand, training an optimal soft margin needs to adjust more parameters and cross-validation often be used, those make the training speed slowly as the training set is large.

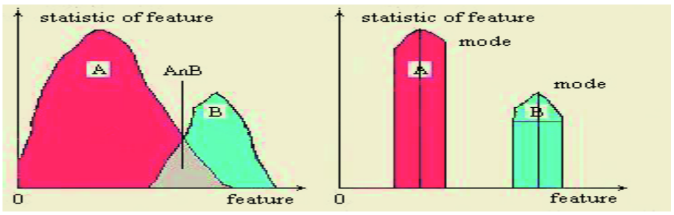
Many approaches have been presented to training by minimizing the number of training samples, such as active learning in [1], [6] and selecting appropriate training samples in [7]. Although these algorithms have been proved to accelerate the training, they cannot operate in real-time image segmentation.

To speed up the training and avoid overfitting, we present a strategy that select pure samples for training and use optimal hard margin criterion instead of soft margin one.

According to the discussion in [5], the noisy data that soft margin need overcome should be one of the following types: (1)overlapping class probability distribution, (b)outliers and (c)mislabeled patterns. It is easy to restrain such noisy data in histogram.

Image shows its statistical features in histogram, in which all the evidence for the presence of a significant feature is pooled together, providing excellent tolerance to a noise level. Fig. 1 shows an example of 1D histogram, A and B represent two classes respectively and an intersection between them. Obviously, if we find the modes in histogram, the pixels corresponding to the modes will be pure samples without noise, see figure 1b.

As mentioned above, SVM need not estimate the density distributions of classes for classification, only need some samples inhabit the distribution fields in feature space (histogram). So uniform sampling could be a rough way to select some pixels for representation the feature distribution, i.e. the original dataset of pixels can be represented with a subset by uniform sampling when they have approximate feature distributions in histogram.



(a) (b)

Fig. 1. (a) exist noisy data, (b) pure samples without noise

2.2 Mean-Shift

Mean-shift, a simple nonparametric technique for estimation of the density gradient, is used in our work in order to find the local modes (local maxima of frequency) in histogram. It is an iterative procedure to find fixed point by an initial search window. See reference [8] for detail.

In our work, RGB space is the mean shift workspace, the pixels frequency instead of probability density. Mean shift algorithm is described as the following:

(1) Choose the bandwith h of the search window.
(2) Choose the initial location p of the window.
(3) Compute the mean shift vector and translate the search window by that amount.
(4) Repeat till convergence.

The fixed point found by a mean shift is the local highest density region that corresponds to clustering mode in color space.

3 Unsupervised Procedure for Segmentation

A four-step unsupervised learning-classification procedure for segmentation can be designed as follows:

① Find clustering modes by mean-shift and preparing candidates I^+ and I^- for training. In order to avoid uncertainty, we set $I^+ \cap I^- = \Phi$ (empty set). Let the candidate set $(\mathbf{x}_i, l_i)$, $\mathbf{x}_i \in I^+ \cup I^-$, $l_i = \begin{cases} +1, if & \mathbf{x}_i \in I^+ \\ -1, if & \mathbf{x}_i \in I^- \end{cases}$.

② Uniform sample N pixels from I^+ and I^- to construct a reduced training set whose features are represented with $(R, G, B, x, y)^T$ or $(R, G, B)^T$, where (R,G,B) is RGB color value of the pixel, (x, y) is coordinate of the pixel (spatial information).

③ Train a SVM online by SMO method using the reduced training set and hard margin criterion.

④ The SVM model classifies the image pixels with corresponding features.

4 Experiments and Discussion

We carry out an experiment on analysis of blood and bone marrow smears, which are conventionally prepared with Wright-Giemsa stain. According to the prior knowledge about cell image, three mean shift procedures are firstly used to find clustering modes of nucleus, mature erythrocytes and background. Then the region of nucleus is dilated suitably in image in order to get a part of color pixels of cytoplasm around them. By doing so, four different regions can construct candidates I^+ and I^-. (See Fig.2).

The corresponding initial locations of search window are defined as p_n, p_e and p_b, the bandwidths are h_n, h_e and h_b respectively. Since background always near the brightest point and nucleus near the darkest point in color space, we set $p_n(R,G,B)=(0,0,0)$ and $p_b(R,G,B)=(255,255,255)$. The color of mature erythrocytes is often various in different image acquisition cases, so p_e could be specified manually by mouse clicking or set experientially, for example, $p_e(R,G,B)=(180,120,130)$ in this paper. To guarantee mean shift procedure convergence to the mode, h_n , h_e and h_b will increase adaptively in program before the search window first shift. Fig.2 presents a schematic example.

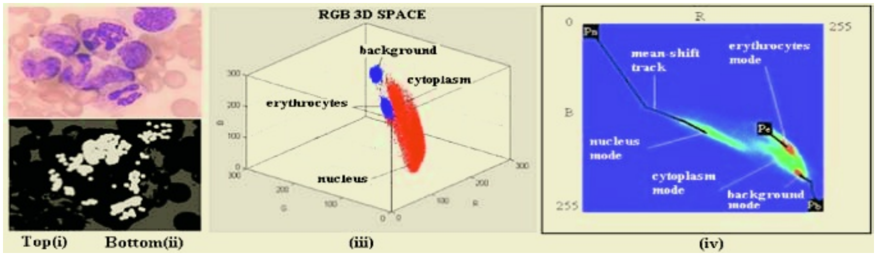


Fig. 2. (i) A blood cell image. (ii) Candidate regions labeled with different gray level (bright gray I^+, deep gray I^- and black not labeled). (iii) Color vectors distribution in RGB space. (iv) Three clustering modes and corresponding mean shift tracks be illustrated in RB color plane

In our experiments, the kernel function of SVM is radial basis function, which is

$$K(\mathbf{x}_i,\mathbf{x}_j)=\exp(-q\left\|\mathbf{x}_i-\mathbf{x}_j\right\|^2)\,. \tag{3}$$

We present two types of SVM-based classifiers with 5-dimension and 3-dimension features respectively. The parameters are fixed, N=1000, q=1, $h_n=h_e=h_b=5$. Hardware is P4_1.7GHz cpu, 256MB memory. Software is C++ program, Windows 2000.

As a comparison, watershed based method also performs the same segmentation task [9], the labeled seeds prepared by mean-shift procedure similar as candidate regions in Fig.2(ii). More than 200 images (768*570) and 1,000 nucleated cells had been tested in our work. The segmentation results and average time cost by our method and watershed algorithm are compared in table 1. The appraisement criterion of segmentation accuracy is satisfied degree judged by pathologist [9].

Table 1. Comparative results of segmentation accuracy and average time cost.

Method	Accuracy of nucleus	Accuracy of cytoplasm	Time cost
Watershed-based	92.0%	89.8%	8.1 s
SVM-based (5D)	99.1%	92.5%	7.7 s
SVM-based (3D)	99.1%	91.5%	2.0 s

Some typical experiment results are shown in Fig.3. The (i), (ii) and (iii) are three original images. Their preparation and illumination are large varied. Watershed algorithm can separate cluster cells, but it is impossible to separate all the clustering cells in once process. Moreover, in the presence of noise, clutter, and occlusion, it is difficult to obtain good segmentation results. Fig.3. (iv),(v) and (vi) show some shortages of watershed algorithm. In 3D case, our method can obtain more accurate results, see Fig.3.(vii), (viii), (ix). Because more information is used in 5D case, our method is powerful to restrain segmentation noise, see Fig. 3. (x),(xi),(xii).

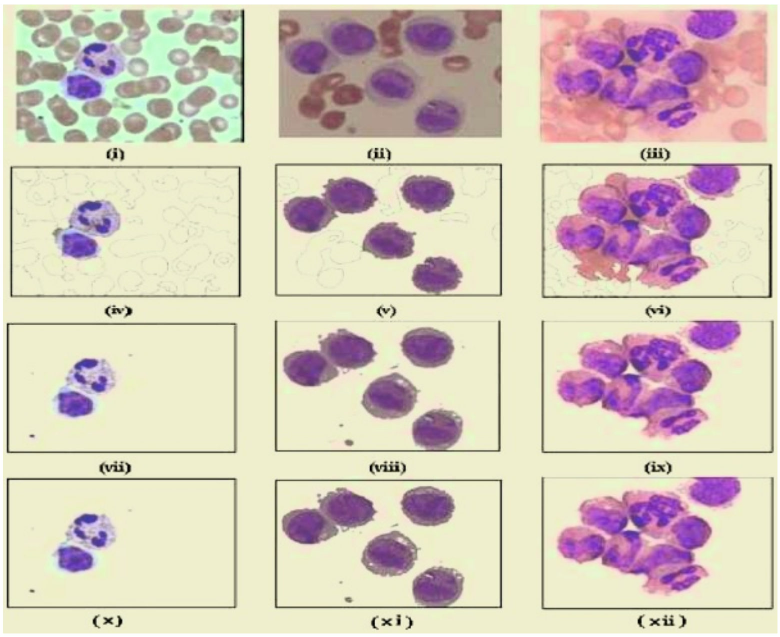


Fig. 3. Color image segmentation results achieved by the watershed-based (iv, v, vi), SVM-based method on 3D(vii, viii, ix) and 5D(x, xi, xii). Oversegmentation appears in (iv), undersegmentation appears in (vi), and (v) loses little edge pixels, while coresponding segmentation results by SVM-based method are more accurate.

Since initial search window of mean-shift is an initial location, the corresponding mode can tolerate it in anywhere of the attraction field. This tolerated margin leads to capability of robust. So our method is robust to many variations such as in the staining process of the slides and in illumination caused by thickness of the smear. The parameters also could be universal to same type images whose color distributions are similar in histogram.

5 Conclusions

In this paper, we present unsupervised learning-classification method for image segmentation. The mean-shift procedures are firstly used to find color clustering modes to prepare candidates for SVM training. Then, SVM can be trained online by a reduced training set through uniform sampling from the candidates. Finally, the SVM model classifies the pixels of the image. Color and spatial information could be utilized together in feature representation. Due to robust and good generalization performance inherited from the mean-shift and SVM algorithms, the new method brings more robust and accurate segmentation. Experimental results demonstrate that the new method is suitable to segment blood and bone marrow microscopic images.

References

1. Zhang, J.-G., Ma, K.-K, ER, M.-H.: Tumor Segmentation From Magnetic Resonance Imaging By Learning Via One-Class Support Vector Machine. Proceeding of International Workshop on Advanced Image Technology (IWAIT04), (2004) 207-211
2. Kotopoulos, C., Pitas, I.: Segmentation of Ultrasonic Images Using Support Vector Machines. Pattern Recognition Letters, **24** (2003) 715-727
3. Melgani, F., Bruzzone, L.: Classification of Hyperspectral Remote Sensing Images With Support Vector Machines. IEEE Trans. on Geoscience and Remote Sensing, **42** (2004) 1778-1790
4. Vapnik, V.: Statistical Learning Theory. John Willey & Sons, New York (1998).
5. Ratsch, G., Onoda, T., Muller, K.R.: Soft Margins for AdaBoost. Machine Learning, **42** (2001) 287-320
6. Schohn, G., Cohn, D.: Less is More: Active Learning with Support Vector Machines. Proceedings of the Seventeenth International Conference on Machines Learning (ICML2000), Publisher:Morgen Kaugmann, Standord, CA,USA (2000) 839-846
7. Zhan, Y.-Q., Shen, D.-G.: Design Efficient Support Vector Machine for Fast Classification. Pattern Recognition, **38** (2005) 157-161
8. Comaniciu, D., Meer, P.: Mean Shift: A Robust Approach Toward Feature Space Analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence, **24** (2002) 603-619
9. Lezoray, O., Elmoataz, A., Cardot, H., et al.: Segmentation of color images from serous cytology for automated cell classification. Analytical and Quantitative Cytology and Histology, **22** (2000) 311-322

Multisensors Information Fusion with Neural Networks for Noninvasive Blood Glucose Detection

Wei Wang[1], Lanfeng Yan[2], Baowei Liu[1], and Heng Zhang[1]

[1] School of Information Science and Engineering, Lanzhou University, Lanzhou, Gansu 710000, China
[2] People's Hospital of GanSu Province, Lanzhou, Gansu 730000,China
kinggreat@163.com

Abstract. A multisensors information fusion model (MIFM) based on the Mixture of Experts (ME) neural networks was designed to fuse the multi-sensors signals for infrared noninvasive blood glucose detection. ME algorithm greatly improved the precision of noninvasive blood glucose measurement with multisensors. The principle of ME, design and implementation of MIFM were described in details. The standard deviation of the error of predication (SO) was 0.88 mmol/l from blood and 0.65 mmol/l from water-glucose. The correlation coefficient (CC) to training data from blood analysis was 0. 9.

1 Introduction

For prevention from complication, the tight control of blood glucose level is necessary. There are numerous methods in noninvasive blood glucose measurement; near-infrared spectroscopy [1], far-infrared spectroscopy [2], middle-infrared oscillating thermal gradient spectrometry [3], laser photoacoustic [4], optical rotation of polarized light [4] and reverse iontophoresis fluid extraction [5]. In those methods near-infrared can be realized at low cost and fits to use in small device, which can measure blood glucose by a beam of infrared through special tissues of body and without any blood sample and pain, it extracts blood glucose information from the spectroscopy [6]. Infrared noninvasive measurement of blood glucose in body shows a great of problems for precision and stability [7]. With traditional noninvasive measurement methods, single infrared sensor and wavelength range was used to determine blood glucose information [8]. Those methods cannot get enough blood glucose information to yield nice detecting results [9]. Infrared noninvasive blood glucose measurement based on multisensors and Mixture of Experts (ME) [10] is a new method, which can relieve the pain from a finger stick and avoids the infection of disease via blood. This method can improve the effectiveness of blood glucose monitoring and measurement in health care of diabetes. ME neural networks (MENN) was first time used in multisensors information fusion model (MIFM). It can divide a difficult task into appropriate subtasks, each of which can be solved by a very simple expert network [11] and may offer better way for multisensors information fusion and signal processing in multisensors measurement.

2 Model of Infrared Blood Glucose Measurement

Near-infrared noninvasive blood glucose measurement is developed from technology of component measuring based on spectroscopy and bases on the Lambert-Beer Law, which takes advantage of glucose absorption feature in spectroscopy [8]. The range of near-infrared wavelength from 700 to 2500nm can pass through or be reflected by tissues, and components show different absorption feature. Utilizing the absorption feature of glucose can differ glucose component from other components in blood, and its concentration has a direct corresponding relation. Owing to special absorption peak in spectroscopy [3], the measurement of certain wavelength spectroscopy was detected to extract information of blood glucose. The relation writes as:

$$D = LnI \; / \; Io = \varepsilon cd \; . \tag{1}$$

Where: *Io*=export intensity of light; *I*=import intensity of light; *c*=concentration of substance, *d*=distance of light through substance; ε =light absorptive coefficient of certain substance; *D* is the light density. If the distance *d* is a constant, the concentration of this substance c shows a direct ratio to *D*. Omitting the scattering, the transmission *T* through an absorbing sample follows the absorption:

$$T(\lambda) = \exp(-da_m(\lambda)) = \exp(-S(\lambda)) \, . \tag{2}$$

Where: *d*=distance of absorption path; $a_m(\lambda)$=absorption coefficient of the matrix at wavelength λ; $S{=}da_m(\lambda)$ There are many factors affect infrared absorption, fat, protein, water, skin of detecting position, nail, tissue and little bones. In those factors, the reflection of skin, water and fat show main affection. With the analysis of relation between ages, height, weight and fat, water, reflection of skin, a set of affecting parameters were built and fingers were selected as measuring position. In past studies those factors were ignored. The absorption should be modified as:

$$\Delta I = I - (I_{dc} + I_f + I_r + I_t + I_g) = I - I(a1 + a2 + a3 + a4 + a5) \, . \tag{3}$$

In equation (3): ΔI=alteration of infrared absorption, I_{dc}=intensity of scattering, I_f=absorption intensity of fat, I_t=other absorption intensity, I_g=absorption intensity of glucose. According to the Lambert-Beer Law, when portion ΔG represents absorption of glucose, the same part of the absorption of spectroscopy $a_m(\lambda)$ is replaced by $\alpha G(\lambda)$, if $S{=}\ da_m(\lambda)$, $T{=}(a1{+}a2{+}a3{+}a4{+}a5)$, the sensitivity was defined as a ratio of relative changes in transmitted *I* induced by change ΔG in glucose, written as:

$$\frac{(1-T)}{\Delta G} = \frac{[\dfrac{I\exp(-d\Delta G(\alpha_G - \alpha_M)) - I}{I}]}{\Delta G} =\approx -d(\alpha_G - \alpha_M) = -S(\lambda)CG(\lambda) \tag{4}$$

Therefore, the concentration of blood glucose is:

$$CG(\lambda) = \frac{\alpha_G(\lambda)}{\alpha_M(\lambda)} - 1 \, . \tag{5}$$

From relation of equation (5), the maximal sensitivity wavelength should be selected to maximize the value of $CG(\lambda)$, at the same time $S(\lambda)$ should also be selected optimally. If $S(\lambda)$ is too low, it will produce low sensitivity and too high will result in a low signal to noise ratio due to exponential decrease in intensity [8].

For keeping a measuring accurate degree, the sensitive and unsensitive wavelength (600 to 2500nm) was selected, which can effectively include most absorption of sensitive or unsensitive absorption in blood glucose. Based on selected multiple wavelength at every sensor, N*M sensors were used to acquire special information and prepared input data for MIFM to process.

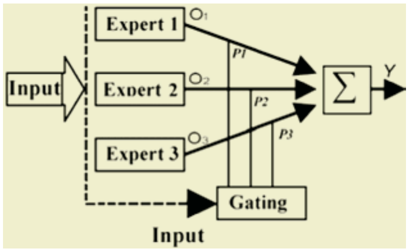


Fig. 1. Principle frame of Mixture of Experts neural networks.

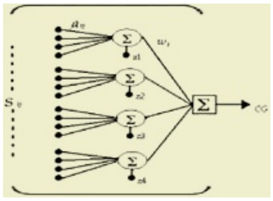


Fig. 2. Model of multisensors information fusion based on Mixture of Experts.

3 Frame of Mixture of Experts Neural Network

ME is a model that estimates the conditional probability distribution of a set of training patterns (Fig 1) [7]. It consists of multiple supervised neural networks, trained to specific regions of the input space. These networks use the idea of competitive learning where each Expert competes with the other Experts to generate the correct desired output. The gating network acts as a mediator to determine which Expert is best suited for that specific region of input space and assigns that portion to the Expert. This system is to divide a large complex set of data into smaller subsets by the gating networks that allow each Expert to better represent a specific subset. It is easier to learn multiple simple functions than to learn large complex ones. The experts are assigned to a specific region of the input space, which allows them to model that space. The idea of Experts is that each Expert assigned to its own unique input space. In combining these networks, it combines all the Experts that generate the system output with a better performance. The equation for the output is:

$$Y = \sum_i P_i Y_i \tag{6}$$

Where p_i is the proportion that the i Expert contributed to the output determined by the gating network; Y_i is the output from the i Expert. The values of all weights within the last layer were set to random values between -1 and 1. For training the networks, it is necessary to modify the weights between the last two layers of the networks based on the whole system error. Below relation is sum of squares error:

$$E^c = \frac{1}{2}\sum_i P_i^c \, \| d^c - O_i^c \|^2 \tag{7}$$

Where d^c is the desired output and o^c is the actual output of the i Expert. The objective is to minimize the total error to the output of the entire system. The system would assume that the prior probability, P_i generated by the Gating network was correct and would modify the weights based on the error from only the i Expert and not the error of the entire system. Therefore, the error function used can be written as:

$$E^C = -\ln\left(\sum_i p_i^C e^{-\| d^C - o_i^C \|^2}\right) \tag{8}$$

This relation is considered to be the negative log of the associative Gaussian mixture model. The derivative of this equation is:

$$\frac{\partial E^c}{\partial o_i^c} = -\left[\frac{p_i^c e^{-\frac{1}{2}\|d^c - o_i^c\|^2}}{\sum_j p_j^c e^{-\frac{1}{2}\|d^c - o_j^c\|^2}}\right](d^c - o_i^c) \tag{9}$$

The equation represents the amount of error that the i Expert contributed to the total system error. In training a network using gradient descent, the derivative of the error function with respect to the weights is used. The change in weight for the Expert networks, using gradient descent should be:

$$\Delta w_{ij} = nb_i\left[\left[\frac{p_i^c e^{-\frac{1}{2}\|d^c - o_i^c\|^2}}{\sum_j p_j^c e^{-\frac{1}{2}\|d^c - o_j^c\|^2}}\right](d^c - o_i^c)\right] f^i(\sum_i w_{ij} b_i) \tag{10}$$

The b_i equals to the input from the previous layer. The $f(x)$ is the activation function of the network. The gating network determines what probability each Expert will generate the desired output. The gating network is considered to be a single layer Perceptron. The gating network uses the same inputs as all the Expert systems, with the number of outputs equal to the number of Experts. The output from the network is the estimated probability, p_i, that the specific Expert will generate the desired output. For the gating network, the Softmax activation function shows as:

$$p_i = \frac{e^{x_j}}{\sum_j e^{x_j}} \tag{11}$$

The equation (11) is used to generate positive values with a sum equal to one. Initially, Each Expert should have the same probability of successfully generating the desired output; therefore the weights in the last layer of the gating network must be set to 0, which will cause the probabilities to be equally distributed. Similar to the Expert networks, the gating network is desirable to minimize the entire systems error in training. To substitute the Softmax function into the systems error function, the error function was written in the following form:

$$E^c = -\ln(\sum_i \left[\frac{e^x}{\sum_j e^{x_j}}\right] e^{-\|d^c - o_i^c\|^2}) = -\left[\ln(\sum_i e^x e^{-\|d^c - o_i^c\|^2}) - \ln(\sum_j e^{x_j})\right] \tag{12}$$

$$\frac{\partial E^c}{\partial u_i} = (h_i - p_i)x \tag{13}$$

The equation (13) is from equation (12), where u_i represents the weights in the last layer of the gating network, h_i and x is the input. The p_i can be considered to be the prior probability of selected i Expert, while h_i is considered the posterior probability of selected i Expert. When training by gradient descent learning, the change in weights of the gating network is defined as:

$$\Delta w_{ij} = n(h_i - p_i)x \tag{14}$$

4 Model of Multisensors Information Fusion Based on ME

Using near-infrared to extract information of blood glucose must need enough signal to noise ratio to identify weak signals of blood glucose from noise of other components. For this goal, suitable method of multisensors information fusion is very important. The coarse signal from multisensors must be fused and calibrated by MIFM and then glucose concentration is calculated. For single sensor glucose measuring, partial least-square methods (PLS) was used [12] in past, which is not a ideal algorithm to fuse multisensors information. In this study ME [8] was designed to fuse multisensors signals, which showed a better precision and nonlinear feature (Fig.2). ME algorithm is based on the Expectation Maximization (EM), and can be used to fuse multisensors signals and extract enough information from blood glucose spectroscopy. With regard to MENN, there are *n* Individual Experts to predict the value of an output. The variable *CG* is given by the summation of the values of Individual Expert; each Expert weighted by a suitable weighting factor W_i, written as:

$$CG = \sum_{i=1}^{n} w_i \sum_{j=1}^{m} a_{ij} S_{ij} + z_{ij} \cdot \tag{15}$$

The output of each Expert is a linear summation of *m* appropriate input parameters S_{ij} pulsed a constant Z_{ij}. At the same time the weighting factor W_i is determined by another relationship of input parameters S_{ij}, which can be utilized to process nonlinear factors, the weighting factor *Wi* is optimized by Expectation Maximization (EM) [8]:

$$w_i = \frac{\exp\left(\sum_{j=1}^{m} a_{ij} s_{ij} + \xi_i \right)}{\sum_{i=1}^{n} \exp\left(\sum_{j=1}^{m} a_{ij} s_{ij} + \xi_i \right)} \cdot \tag{16}$$

Where ξ_i is a constant, there are *n* Experts and m input parameters in a system, 2n (m+1) unknown coefficients (a_{ij}, z_i, S_{ij}, ξ_i) need to be determined [7,8]. The optimized procedure implements in two steps: In step one, the weighting factors associated with each output are fixed and then the parameter values are optimized. In step two, the parameters values are fixed and then the weighting factors are optimized. For the whole procedure step one and step two are then iterated until convergence is achieved. The coefficients (a_{ij}, z_i, S_{ij}, ξ_i) are input coefficients, the output variable *CG* is fused information of blood glucose, and the variable S_{ij} is acquired signals from different channel of multisensors through several gathering. The weights W_i has a responding relationship with each output, it is optimized in the distribution of Experts *i*. The coefficients (a_{ij}, z_i, S_{ij}, ξ_i) can be initiated by affecting factors in blood glucose measuring model, which can be used to resolve some nonlinear problems, to remove interfering of other factors and process superposition of information. Through adjusting factors of age, height, weight etc. the system modifies the individual difference of every body; the measuring accurate degree can be mostly improved. ME can work in linear or linear state. The model used data from blood biochemical detecting as learning arm *CG*, and data from multisensors with different range of wavelength as the input to train the networks. According to the trained model, the MIFM give out correct information of blood glucose. It is important to note the learning: a). The training data and learning arm must synchronously come from the same person; b). According to range of blood glucose, the coefficients must be modified to respond the range of input data in the training. The change of coefficients must be limited to appropriate step to improve learning speed.

5 Experiments and Results

The 4*4 sensors of different wavelength were used to accept the information from different sensors. The MIFM with multisensors and the biochemical analysis method were applied to detect glucose values in water-glucose, plasma and blood at the same time. The nine group of data set from water-glucose, plasma and blood were collected by MIFM and inputted into S_{ij} for training model, and the data set from biochemical analysis were inputted into *CG* as learning arm. Variable *i* was four responding to Experts, and *j* was four according to the number of sensors belong to each Expert (Fig.2). MIFM of infrared noninvasive blood measurement was improved by inputting the affecting factors of age, height and weight, which related with fat, water, protein and reflection of skin. The model modified the weights W_{ij} of each Expert network. After training, the data from multisensors measurement was inputted to S_{ij}, and then the value of glucose was given out. To validate MIFM, the training data sets were re-inputted into the model, and the value of glucose *CG* was extremely approximated to the training data sets (*CC*=0.91±0.14, *n*=16). The results from MIFM showed better relativity to results from blood sample analysis. Comparing with results of biochemical analysis, the other experimental results from methods of single sensor, enzyme pole, microanalysis and MIFM were described in Table 1:

Table 1. The comparing analysis of experimental results from different methods.

Methods	Water-glucose		Plasma		Blood	
	SO	CC	SO	CC	SO	CC
Single Sensor	0.90	0.96	1.10	0.70	1.14	0.51
Enzyme Pole	0.66	0.90	0.68	0.80	0.92	0.72
Microanalysis	0.58	0.93	0.66	0.85	0.70	0.92
MIFM	0.65	0.95	0.70	0.90	0.88	0.90

The standard deviation of the error of predication (SO) was 0.88 mmol/l from blood and 0.65 mmol/l from water-glucose. The correlation coefficient (CC) to training arm from blood analysis was 0. 9. The training of MENN was simple and the evaluative results were: Mean Absolute Error (MAE 14.4), Least Squares Slop (LSS 0.96), Least Squares Intercept (LSI 2.2), Orthogonal Slop (OS 1.04).

6 Discussion and Conclusion

Due to that glucose content is 1/1000 of other components in blood, thus weak variety in glucose is difficult to detect. The infrared MIFM can improve the performance of identifying weak changes of glucose concentration. The information fusion algorithm based on ME makes measurement more precise than other methods. The training of

MIFM divides to three processes：1.when the spectral information of glucose from multisensors is collected, at the same time the glucose biochemical analysis as training arm must be achieved; 2.according to physiologic modifying coefficient, spectral data, and data of training arm, the MIFM is trained; 3.with trained MIFM, glucose value is calculate out. The ME can divide a large, difficult task into appropriate simple and each of subtasks can be solved by a very simple expert network [7], which is easier to implement in the application than other methods. This method may offer robustness for multisensors information fusion in development of small instrument.

References

1. Arnold, M.A., Burmeister, J.J., Small, G.W.: Phantom Glucose Calibration Models from Simulated Noninvasive Human Near-infrared Spectra. Anal Chem, **70** (1998) 1773-1781
2. Klonoff, D.C.: Noninvasive Blood Glucose Monitoring. Diabetes Care, **20** (1997) 433-437
3. Zheng, P., Kramer, C.E., Barnes, C.W., Braig J.R., Sterling, B.B.: Noninvasive Glucose Determination by Oscillating Thermal Gradient Spectrometry. Diabetes Technol Ther, **2** (2000) 17-25
4. Christison, G.B., MacKenzie, H.A.: Laser Photoacoustic Determination of Physiological Glucose Concentrations in Human Whole Blood. Med Biol Eng Comput, **31** (1993) 284-290
5. Robinovitch, N.: Noninvasive Glucose Determination by Reverse Iontophresis Fluid Extraction from Skin. Diabetic Care, **5** (1982) 259-265
6. Tenhunen, J., Kopola, H., Myllylä, R.: Non-invasive Glucose Measurement Based on Selective Near Infrared Absorption; Requirements on Instrumentation and Spectral Range. Measurement, **24** (1998) 173-177
7. Pan, S., Chung, H., Arnold, Ma., Small, G.W.: Near-infrared Spectroscopic Measurement of Physiological Glucose Levels in Variable Matrices of Protein and Triglycerides. Anal Chem, **68** (1996) 112-1135
8. Muller, U.A., Mertes, B., Fischbacher, C., Jageman, K.U., Danzer, K.: Noninvasive Blood Glucose Monitoring by Means of Near Infrared Methods for Improving the Reliability of the Calibration Models. Int J Artif Organs, **20** (1997) 285-290
9. Gabriely, I., Wozniak, R., Mevorach, M., Kaplan, J., Aharon, Y., Shamoon, H.: Transcutaneous Glucose Measurement Using Near-Infrared Spectroscopy During Hypoglycemia. Diabets Care,**22** (1999) 2026-2032
10. Jacobs, R.A., Jordan. M.: Adaptive Mixtures of Local-Experts. Neural Computation, **3** (1991) 79-87
11. Kurnik, R.T., Oliver, J.J, Waterhouse, S.R., Dunn, T.: Application of the Mixtures of Experts Algorithm for Signal Processing in A Noninvasive Glucose Monitoring System. Sensors and Actuators, b**60** (1999) 19-26
12. Malin, S.F., Ruchti, T.L., Blank, T.B., Thennadil, S.N., Monfre, S.L.: Noninvasive Prediction of Glucose by Near-infrared Diffuse Reflectance Spectroscopy. Clin Chem., **45** (1999) 1651-1658

Disease Diagnosis Using Query-Based Neural Networks*

Ray-I Chang

Department of Engineering Science & Oce. Engineering,
National Taiwan University, Taipei, Taiwan, China
rayichang@ntu.edu.tw

Abstract. The ability of high tolerance for learning-by-example makes neural networks flexible and powerful in resolving various application problems. However, while being applied in real world, the time required to induce models from large data sets should be considered. In this paper, we apply QSS (*Q*uery-based learning with *S*elective-attention and *S*elf-regulation) to back-propagation neural networks for resolving the data classification problem in biomedical applications. Results show that the proposed method can significantly reduce the training set cardinality. Additionally, the quality of training results can be ensured. It provides a powerful tool to help physicians analyze, model and make sense of complex clinical data for disease diagnosis.

1 Introduction

Disease diagnosis is a process that discovers the diseases from the manifested symptoms. Some diseases may have similar symptoms and complications. A clinician must carefully investigate patients' symptoms, complaints, and the results of pathological examinations to decide possible diseases. Such decisions are really difficult to make as the related etiology is hard to discern or when multiple diseases are suffered [1]. Neural networks (NN) provide a powerful tool to help physicians analyze, model and make sense of complex clinical data across a broad range of biomedical applications [2]. In this paper, based on our early study [3], we apply the concept of QSS (Query-based learning with Selective-attention and Self-regulation) [3] to back-propagation NN (BPN) for disease diagnosis. The ability of high tolerance for learning-by-example makes NN very flexible and powerful [4]. However, the time required to induce models from large data sets should be considered while applying NN in real world [5]. In this paper, a query-based BPN with both selective-attention and self-regulation is introduced for disease diagnosis. Query-based learning is a methodology that requires asking a partially trained NN to respond to the questions [6]. In conventional query-based BPN, only the selective-attention is applied to respond the goal-directed behavior of NN with self-focus [7]. In many applications, there may be no supervisor or the cost is too expansive to verify the self-focus. We need a compromise between the self-focus (the desired sample from NN learning) and the environment-focus (the exist sample in data set) with self-regulation. In this paper, the environment-focus that is close to the self-focus of BPN is applied as the extra sample for next training. As the space is limited, this paper considers two benchmark problems selected among the many (disease diagnosis) datasets available in UCI [17] – disease diagnosis of Heart Disease and Breast Cancer. Experiments show that the perform-

* This paper was partially supported by NSC, under grants NSC93-2213-E-002-086-.

ance is significantly improved and the quality of training results can be ensured. For the diagnosis of Heart Disease, the obtained classification accuracy is larger than BPN (with 3.3% increases) and the required CPU time is smaller than BPN (with nearly 400% decreases). For the diagnosis of Breast Cancer, the obtained classification accuracy is larger than BPN (with 3.5% increases) and the required CPU time is smaller than BPN (with nearly 900% decreases). We have applied the proposed method to other data mining problems recently. Its application is not restricted to medical diagnosis problems.

2 Related Works

BPN is constituted of a set of neurons organized in layers [9]-[11]. Each neuron of a layer is connected to each neuron of the next layer directly. It produces output by taking a linear combination of the input signals and transforming it using the activity function as follows.

$$u_i(\ell+1)=\sum_{j=1}^{N_l} W_{ij}(l+1)a_j(l)+\theta_i(l+1) = \sum_{j=0}^{N_l} W_{ij}(l+1)a_j(l) \text{ and } a_i(l+1)=f(u_i(l+1)). \tag{1}$$

where $a_i(l+1)$ is the output of the generic neuron belonging to layer $(l+1)$ and w_{ij} is the synaptic weight associated with the connection between the generic neurons belonging to layers i and j, respectively. θ_i is the bias term and f is the activity function. In NN learning, the connection weights among neurons are adjusted according to sets of known input-output patterns called samples [12]. The error function E is defined as the difference between the output and target values.

$$E=\sum_{i=1}^{N}(t_i-a_i(L))^2 \text{ and } \frac{\partial E}{\partial w_{ij}}=-(t_i-a_i(L))a_j . \tag{2}$$

Once the partial derivative for each weight is known, the aim of minimizing the error function is achieved by performing a simple gradient descent, according to the updating rule $w_{ij}=w_{ij}+\eta(t_i-a_i(L))a_j$ where η is the learning rate. According to Oates [13], when a machine learning algorithm is to learn, what are usually needed are some particular samples for training. With these samples, the algorithm could learn almost completely what it is taught. According to Baum [14], query-based learning is approximate to the way humans learn. It not only employs the samples for training that are presently at hand, but also uses the query method to produce extra samples. The source of the training information for query could be modeled as an oracle. Thus, the oracle should co-work with the query method that initiates training using a small input set, and then produces appropriate training information with minimum cost. However, the oracle could be very expensive in some applications, such as samples that are simulated by a supercomputer. In a laboratory, we usually have no supervisor to specify the correct output of an arbitrary input. To resolve this kind of drawback, we proposed the QSS scheme to un-supervised learning of self-organizing maps. Notably, BPN is supervised. So far as we known, it is the first paper to consider selective-attention and self-regulation in query-based learning of BPN. It is also the first study to apply query-based neural networks for disease diagnosis.

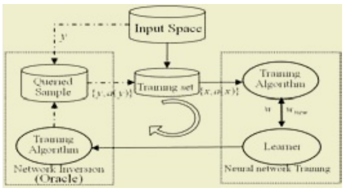


Fig. 1. Query-based learning.

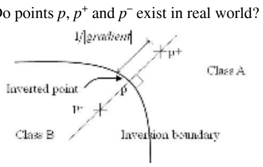
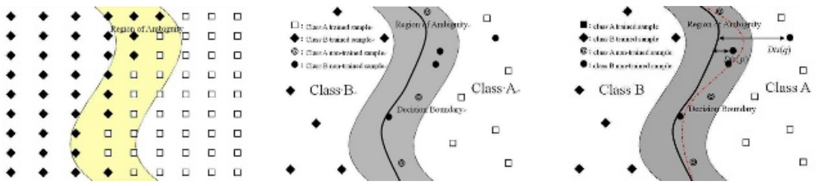


Fig. 2. Conjugate data pairs.

3 Proposed Method

Fig. 1 shows the framework of a query-based learning algorithm. When the point of query is set as y, the oracle would respond with $a(y)$. $\{y, a(y)\}$ is called the queried sample. In [15], information from the decision boundary has been proved to produce the best training results. In other words, it would search among the input pieces of information for the points whose output value is 0.5 (the neuron output is between 0 and 1). Given $a(y)$=0.5, we need to decides the point y. As [6], we first employ the inversion algorithm (a more detail description is shown in [6]) to gain the points that could erase errors. Like the back-propagation algorithm, the inversion algorithm propagates the error signal backward to the input layer to update the activation value of the input units and eventually to lower the rate at which an error happens to the output value.



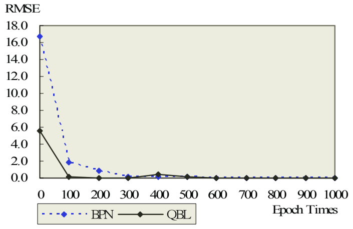
(a) (b) (c)

Fig. 3. Query-based learning with self-regulation.

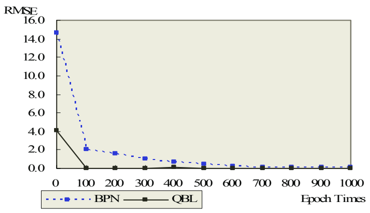
As shown by Fig. 2, the conjugate training data pair is extracted along the reverse boundary. The inverted boundary point p (where the NN output $a(p)$=0.5) would be used to calculate the reciprocal of the magnitude of gradient, and then along the magnitude of gradient we would pick up two information points in symmetry, that is, p^+ and p^-. In conventional approaches, we would include these three points into the extra training set. Notably, these approaches have assumed that, for each point, the NN output is known and the input-output pattern is exist as shown in Fig. 3(a). It is inappropriate for applications such as disease diagnosis where an arbitrary input-output pattern may not be exist in real world or in our data set (see Fig.3 (b)). In this paper, we follow the self-regulation rule [23] to select samples (environment-focus) those are close to the conjugate data pair (self-focus) as the queried samples. As Linden [15] and Reed et al. [16] suggested, the input data of the noise/jitter or the outlier would not influence the decision boundary produced by NN. We would consider the queried samples that have been put in wrong classes. Further, the queried samples would be saved in the priority queue sorted by min-heap to select the queried samples that are the most close to the boundary. Take Fig. 3(c) as an example, □ and ◆ mean the initial training samples of class A and class B, respectively. Usually, they are randomly selected from the data set. The black line means the boundary after training. In the proposed method, the non-trained samples of class A and class B (◎ and ●, respectively) would be examined by the oracle to detect whether they are put in the wrong class. Then, we calculate distance from the boundary to each sample and pick the samples that are the most close to the boundary as the extra samples. When the RMSE (root of mean squared error) of the previous prediction and the present calculation is lower than the given threshold, the learning process is finished.

4 Experiment Results

In this paper, we examine two NN – our QSS BPN (called QBL) and the original BPN to verify the feasibility and effectiveness of the proposed method. Two data sets – Heart Disease and Breast Cancer – in UCI Machine Learning Repository Database [17] are applied as the test data. Notably, there are four heart disease data sets (Cleveland, Hungary, Switzerland, and the VA Long Beach) in UCI. The one used in the experiment is Cleveland wth 240 samples in training sets and 30 samples in test sets. 13 of the 75 attributes were used for prediction in 2 separate tests. The chosen 13 attributes are all continuously valued. For correctness and reliability, we use the rotation method to randomly sample the training data. It would train and examine group by group the pieces of training data in the data sets. In initial, QBL chooses 20% of samples for training randomly. After inquiring oracle, new training samples are added. Our system has only one hidden layer as [18],[19] recommended that one-hidden-layer network is sufficient to model any complex system with any desired accuracy. The number of neurons on the input (hidden, output) layer is *m* (2*m*, 1) where *m* is the number of input attributes and *m* = 13 (9) for the Heart Disease (Breast Cancer) data set. The learning rate is set as 0.45 for efficient computation although lower rates tend to give better results [12]. The momentum term is 0.9. The convergence criteria are either RMSE is less than 0.01 or a maximum of 1000 iterations is reached. Fig. 4 shows the convergence behavior in NN training. QBL is faster than BPN in convergence. Comparing the required CPU time and the obtained accuracy listed in Table 1, QBL is shown to be effective and efficient. For the diagnosis of Breast Cancer, the obtained accuracy has 3.5% increases and the required CPU time has 900% decreases.



(a)



(b)

Fig. 4. Convergence behavior for (a) Heart Disease and (b) Breast Cancer.

Table 1. The required CPU time and the obtained accuracy.

	Heart Disease		Breast Cancer	
	QBL	BPN	QBL	BPN
CPU time (sec)	*14.76*	52.79	*5.88*	53.29
Accuracy (%)	*86.7%*	83.4%	*97.9%*	94.4%

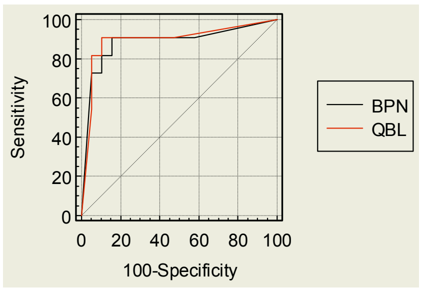
Table 2. The confusion matrix.

	Predicted / Actual	Heart Disease		Breast Cancer	
		Positive	Negative	Positive	Positive
QBL	Positive	10	1	88	2
	Negative	3	16	3	140
BPN	Positive	9	2	80	10
	Negative	3	16	3	140

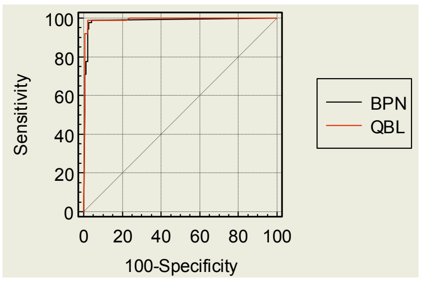
Table 3. The obtained results.

	Heart Disease		Breast Cancer	
	QBL	BPN	QBL	BPN
ROC Curve (AUC)	*0.895*	0.885	*0.992*	0.985
Sensitivity	*0.77*	0.75	*0.97*	0.96
Specificity	*0.94*	0.89	*0.99*	0.93
PPV	*0.91*	0.82	*0.98*	0.89
NPV	0.84	0.84	0.98	0.98

The confusion matrix [20] is a common tool for classification analysis. It is created by matching the predicted and actual values: True Negative (TN), False Positive (FP), False Negative (FN) and True Positive (TP) as shown in Table 2. We applied this matrix to calculate four criterions to deal with the acquired diagnostic mode: sensitivity, specificity, PPV (positive predictive value) and NPV (negative predictive value) as shown in Table 3. In addition we applied to the ROC (Receiver-Operator Characteristic) analysis [21] as an evaluation to look at AUC (Area Under the Curve) and to identify which curve dominates the other for selecting the best-performing algorithm [22]. As shown in Fig. 5, QBL is better than BPN.



(a)



(b)

Fig. 5. ROC Curves for (a) Heart Disease and (b) Breast Cancer.

5 Conclusions and Future Works

Query-based learning differs from traditional methods in that the samples could be selected out of its will for training, instead of accepting whatever information it is fed. The objective of such learning is to produce a training sample that is comprehensive and educative. It is very flexible and powerful in disease diagnosis. Experiments show that the proposed method could gain effective classification with less training cost. Our future works are to extend this concept to develop other NN learning methods and to resolve other biomedical problems.

References

1. Hsu, C.C., Ho, C.S.: A New Hybrid Case-Based Architecture for Medical Diagnosis. Information Sciences - An International Journal, **166** (2004) 231-247
2. Choua, S.M., Leeb, T.S., Shaoc, Y.E., Chen, I.F.: Mining the Breast Cancer Pattern Using Artificial Neural Networks and Multivariate Adaptive Regression Splines. Expert Systems with Applications, **27** (2004) 133–142
3. Chang, R.I., Hsiao, P.Y.: Unsupervised Query-Based Learning of Neural Networks Using Selective-Attention and Self-Regulation. IEEE Trans. Neural Networks, **8** (1997) 205-217
4. Azimi, S., Newman, M. R., Zeligman, F.: Detection of Clusters of Micro Calcifications Using Neural Network-Based Schemes. IEEE Engineering in Medicine and Biology Magazine, (1997) 533-534
5. Craven, M.W., Shavlik, J. W.: Using Neural Networks for Data Mining. Future generation computer systems, **13** (1997) 211–229.
6. Hwang, J., Choi, J., Oh, S., Marks II, R.: Query-Based Learning Applied to Partially Trained Multilayer Perceptrons. IEEE Transactions on Neural Networks, **2** (1991) 131-136
7. Chang, R.I., Hsiao, P.Y.: VLSI Circuit Placement with Rectilinear Modules Using Three-Layer Force Directed Self-Organizing Maps. IEEE Trans. Neural Networks, **8** (1997) 1049-1064
8. Cios, K.J., Chen, K., Langenderfer, R.A.: Use of Neural Network in Detecting Cardiac Disease from Echocardiographic Images. IEEE Engineering in Medicine and Biology Magazine, (1990) 58-60
9. Park, D., et al.: Electric Load Forecasting Using an Artificial Neural Network. IEEE Trans. on Power Systems, **6** (1991) 442-449
10. Lee, K.T., Cha, Y.T., Park, J.H.: Short-term Load Forecasting Using an Artificial Neural Network. IEEE PES Winter Power Meeting, (1991)
11. Djukanovic, M., Babic, B., Sobajic, D.J., Pao, Y.H: Unsupervised/Supervised Learning Concept for 24-Hour Load Forecasting. IEEE Proceedings-C, **140** (1993) 311-318
12. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning Internal Representations by Error Propagation. Paralleled Distributed Processing, **1** (1986) 318-362
13. Oates, T., Jensen, D.: The Effects of Training Set Size on Decision Tree Complexity. The Fourteenth International Conference on Machine Learning, (1997) 254-262
14. Baum, E.B.: Neural-Net Algorithms that Learn in Polynomial Time from Examples and Queries," IEEE Trans. on Neural Networks, **1** (1991) 5-19
15. Reed, R., Marks II, R.J., Oh, S.: Similarities Of Error Regularization, Sigmoid Gain Scaling, Target Smoothing, And Training With Jitter. IEEE Trans. on Neural Networks, 6 (1995) 529-538
16. Linden, A., Kindermann, J.: Inversion of Multilayer Nets. IJCNN, **2** (1989) 425-430

17. Blake, C., Keogh, E., Merz, C. J.: UCI Repository of Machine Learning Databases. http://www.ics.uci.edu/~mlearn/MLRepository.html, Irvine, University of California, (1998)
18. Cybenko, G.: Approximation by Superpositions of a Sigmoidal Function. Mathematical Control Signal Systems, **2** (1989) 303–314
19. Hornik, K., Stinchcombe, M., White, H.: Multilayer Feedforward Networks are Universal Approximations. Neural Networks, **2** (1989) 336–359
20. Kohavi, R., Provost, F.: Glossary of Terms. Machine Learning. **30** (1998) 271–274
21. DeLeo, J. M., Rosenfeld, S. J.: Essential Roles for Receiver Operating Characteristic (ROC) Ethodology in Classifier Neural Network Applications. IJCNN, **4** (2001) 2730-2731
22. Woods K., Bowyer, K. W.: Generating ROC Curves for Artificial Neural Networks. IEEE Trans. Medical Imaging, **16** (1997) 329-337
23. Carver, C.S., Scheier, M.F.: Attention and Self-Regulation: A Control-Theory Approach to Human Behavior. Springer-Verlag, New York (1981)

Study of BP Neural Network and Its Application in Lung Cancer Intelligent Diagnosis

Xuemei Huang, Zhide Tang, and Caixin Sun

MOE Key Laboratory of High Voltage Engineering and Electrical New Technology, College of Electrical Engineering, Chongqing University, Chongqing 400044, China

Abstract. In this paper, the learning algorithm of networks is discussed. The programming example of 3-layer BP networks is given with Visual C++6.0 program langue. Based on this model, a lung cancer intelligent diagnosis system is successfully implemented. Furthermore, the paper introduces network's structure design, preferences and the source of sample datum in factual applications. The ameliorative arithmetic is applied to the study of networks and BP dynamic evolving process is designed. The experiments indicate cell images are recognized and classified by the trained neural network. The study illustrates the system has feasibility and clinical value in lung cancer diagnosis.

1 Introduction

Artificial neural network is a popular method for optimization in many fields. Especially the Back-propagation (BP) network proposed by Rumelhart [1] has been successfully adopted to solve many problems. In recent years, BP networks have played an important role in pattern recognition such as lung cancer diagnosis. Falchini [2] reported the associative neural network technology based on CAD improved the earlier detection of pulmonary nodules on chest radiographs whose sensitivity is increased to 80%, and its accuracy attained 98%. Besides, Nakamura [3] referred computerized analysis of the likelihood of malignancy in solitary pulmonary nodules with use of artificial neural networks.

In this project, 3-layer BP network is adopted to diagnose lung cancer. Firstly, we extract the feature parameters of lung cancer cells according to clinical medicine. By this way many samples are achieved and used to train the neural network. Finally the trained neural network is utilized to diagnose lung cancer.

2 BP Networks and BP Algorithm

This system adopts a 3-layer BP network composed of one input layer (A), one hidden layer (B) and one output layer (C). Each layer contains a number of nodes. Connections exist only between the nodes of successive. Its structure can be illustrated as in Fig.1.

The functions animating the network are the same for all hidden layers and output layer:

$$o_{pi}^{(k)} = f(I_{pi}^{(k)}) = 1/(1+\exp(-I_{pi}^{(k)})) \,. \tag{1}$$

$$I_{pi}^{(k)} = \sum_{j=1}^{n^{k-1}} (w_{ij}^{(k)} \cdot o_{pj}^{(k-1)}) + bias_{i}^{(k)} \,. \tag{2}$$

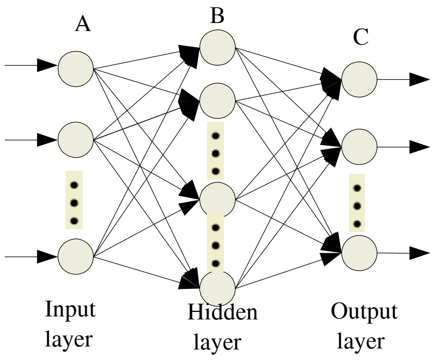


Fig. 1. 3-layer BP network architecture.

Here $o_{pi}^{(k)}$ is the output of node i in layer k, the subscript p denotes the order of arrangement of the training patterns used in the training phase, $w_{ij}^{(k)}$ is the corresponding weight between the node j in layer $k-1$ and the node i in layer k, $bias_i^{(k)}$ is the bias for the node i in layer k, and $n^{(k-1)}$ is the number of nodes in layer $k-1$. The output value of each node ranges of [0, 1]. BP networks are usually trained by following delta rule with a momentum term [4]:

$$\Delta w_{ij}^{(k)}(n) = -\frac{1}{n_p} \cdot \eta_{ij}^{(k)}(n) \cdot \sum_p \frac{\partial E_p(n)}{\partial w_{ij}^{(k)}} + \beta \cdot \Delta w_{ij}^{(k)}(n-1) \cdot \tag{3}$$

$$\Delta bias_i^{(k)}(n) = -\frac{1}{n_p} \cdot \eta_i^{(k)}(n) \cdot \sum_p \frac{\partial E_p(n)}{\partial bias_{ij}^{(k)}} + \beta \cdot \Delta bias_{ij}^{(k)}(n-1) \cdot \tag{4}$$

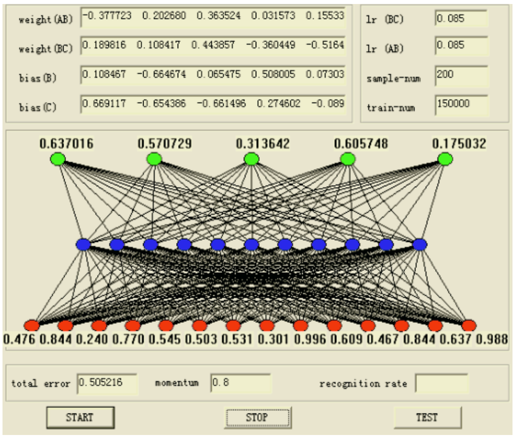
Here $E_p(n)$ is the square error function of the pattern p after $n\,th$ iteration and n_p is the number of training pattern. At $(n+1)th$ iteration during training, the update of connection weight $\Delta w_{ij}^{(k)}$ is based on two components. One is a function of the error gradient. The other is proportional to the amount of weight change in the previous iteration. β is the momentum factor which helps prevent the oscillation problem near the solution point. η denotes the learning rate.

3 System Realization

Our system performs two kinds of function: train and test. The user interface of the training process is illustrated at Fig.2. Its interface as follows:

(1) The number of neurons of input layer, hidden layer and output can be selective.
(2) The parameters designed by user freely include learning rate, momentum factor, iteration number and the number of train sample.
(3) The results displayed are total error, weights, biases and recognition rate.

After training is accomplished, it can be used to diagnose the condition of lung cancer patients. The diagnosis result is achieved according to the output values of simulation of trained network. The test result of a microscopic section picture of lung cancer cells is shown in Fig.3.



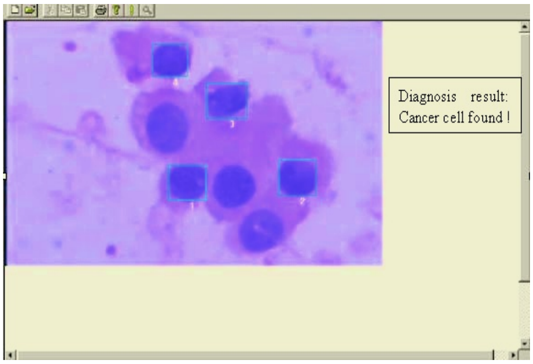
```
weight(AB)  -0.377723  0.202680  0.363524  0.031573  0.15533     lr (BC)     0.085
weight(BC)   0.189816  0.108417  0.443857  -0.360449  -0.5164    lr (AB)     0.085
bias(B)      0.108467  -0.664674  0.065475  0.508005  0.07303    sample-num  200
bias(C)      0.669117  -0.654386  -0.661496  0.274602  -0.089    train-num   150000

0.637016  0.570729  0.313642  0.605748  0.175032

0.476 0.844 0.240 0.770 0.545 0.503 0.531 0.301 0.996 0.609 0.467 0.844 0.637 0.988

total error  0.505216     momentum  0.8     recognition rate
```

Fig. 2. User interface of training network.



Diagnosis result:
Cancer cell found !

Fig. 3. Diagnosis result.

4 Concrete Application and Result Analysis

4.1 Determination of Sample and Network Structure

(1) Source of sample data

The process to construct sample is to abstract machine knowledge from the characteristics of the lung cells of lung-cancer sufferer and healthy people. In this project, we first collect the section pictures or the painted pictures of the specimen by puncturing. Then, they go through image preprocessing, image segmentation and characteristic extracting [5] in order. By this way, the sample datum are obtained successfully. All input sample value must be scaled in the range of [0,1]. Besides, output sample value is defined to be “0” or “1” according to the category of lung cells.

(2) Determination of network structure
The input layer is selected to have 14 neurons corresponding to 14 characteristic value of lung cell: 6 morphologic characteristic values that are selected according to the main differentness between the normal cells and the lung cancer cells. A series of geometrical feature of cell field is extracted by tracking binary image with eight chain code. They are the length of chain code, the width and the height of cell field, the area of cell field, the similarity to rondure of cell field, the similarity to rectangle of cell field and the elongation degree of cell field. 6 colorimetric characteristic values are set to {R, G, B} and {H, I, S} after trial and error. The other two feature values are the mean value of red component of the whole slice picture and the mean value of gray level of cell field.

11 neurons are set in the hidden layer according to experimental value and actual condition. 5 neurons is determined in the output layer corresponding to 5 different diagnosis results: no-carcinoma (NC), adenocarcinoma (AC), squama-carcinoma (SC), cellule-carcinoma (CC), nucleus-allotype (MA).

4.2 Training Procedure

After the elementary structure of the network has been designed, the next step is to select the initial values of every parameter. Generally speaking, the affection of learning rate to training process is that: the system will be instable if the learning rate is too big, while the too small learning rate can cause longer training time. In our project, 0.085 is defined. In the term of the initial selections of weights and biases, random value of [-0.5, +0.5] is set. In addition, the biggest iteration number is 150000. The goal function (square-root error) is set to 0.0001.

Because the momentum factor has great impact on the convergence of network, we make many experiments with the momentum from 0.7 to 0.9 with an increment of 0.05. Finally, we select 0.8 as the momentum factor by comparison in detail. Under the other same parameters, adding momentum can lead to the less error. It proved that the momentum factor accelerate greatly the convergence and decrease vibration of network.

4.3 Training Strategy and Classifying Ability

After many experiments, we apply the different amount of samples to train the network separately, and then save the weights and the biases into a text for test. We collect 500 samples in this system and choose a part of them as a training set to train the network, while the remain part as a test set to test the network. The recognition rate is calculated and shown in Tab.1 meantime. In the testing process, if the output unit of corresponding class produces high value (≥ 0.95) and the output values of other notes are all low values (≤ 0.05), the result will be recognized correctly [6].

In Tab.1, we can see that the recognition rate is higher when the number of samples in training set is bigger, which results from the generalization ability of the neural network. It proved that the increasing number of sample to train enhanced greatly the generalization ability.

Table 1. Results with different train sample.

Number of sample for training	Number of sample for Testing	Recognition rate (%)
100	400	81.5
150	350	92.3
200	300	94.3
250	250	95.6
300	200	97.1
400	100	98.9

4.4 Application Illustration

The trained BP network by the way stated in Sect.4.3 is used to test the classifying ability. We collect the painted pictures of the specimen by puncturing, and then dye them. There are 69 painted pictures in all including 5 NC, 17 AC, 23 SC, 15 CC and 9 MA. By collecting from 8 visual fields for every picture, so there are 552 images in all. After image preprocessing, image segmentation and characteristic extracting, we get 552 test samples including 41 NC cells, 138 AC cells, 164 SC cells, 136 CC cells and 73 MA cells. The test result is shown in Tab.2.

Table 2. Diagnosis result.

Cell category	Test sample	Correct recognition	Misjudgement
NC	41	39	2
AC	138	134	4
SC	164	156	8
CC	136	129	7
MA	73	70	3

The diagnosis result in Tab.2 conforms with the sample datum. The method has higher accuracy and lower misjudgment rate which makes advantage the detection and treatment of earlier lung cancer.

5 Conclusions

The BP networks and BP algorithm with momentum factor are discussed and the programming example of 3-layer BP networks is accomplished. The user interface for training and testing is designed reasonably. The trained neural network is applied to the intelligent diagnosis system for lung cancer. The method improved the robustness of the system and developed a new way. The experiment proved it had feasibility and its total recognition rate increased to 92% which chimed with the diagnosis accuracy of pathological experts.

References

1. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning Internal Representations by Error Propagation. In: D. E. Rumelhart and J. L. McClelland (Ed.), Parallel distributed processing: exploration in the microstructure of cognition: Vol.1, Foundations, The MIT press, Cambridge, Massachusetts (1986)

2. Falchini, M., Stecco, A.L., Carmigalni: Neural Network Based Detection of Pulmonary Nodules on Chest Radiographs, Radio Med (Torino), **98** (1999) 259-263
3. Nakamura, K., Yoshida, H., Engellmann, R.: Computerized Analysis of the Likelihood of Malignancy in Solitary Pulmonary Nodules with Use of Artificial Neural Networks. Radiology, **214** (2000) 823-830
4. Hagan, M.T., Demuth, H.B, Beale, M.H: Neural Network Design, Beijing, China machine press (2002) 232-235
5. Gonzalez, R.C., Woods, R.E: Digital Image Processing, Beijing, Publishing House of electronics industry (2003) 34-75
6. Zheng, N.: Computer Vision and Pattern Recognition, Beijing, Publishing House of De-fence Industry (1998) 8-9

New Methodology of Computer Aided Diagnostic System on Breast Cancer

HeeJun Song[1], SeonGu Lee[1], Dongwon Kim[1], and GwiTae Park[1]

[1] Dept. of Electrical Engineering, Korea University, Seoul, Korea
{nyaong7,cr981027,upground,qtpark}@korea.ac.kr

Abstract. In this paper, a new approach using ANFIS as a diagnosis system on WBCD problem is proposed. The automatic diagnosis of breast cancer is an important, real-world medical problem. It is occasionally difficult to attain the ultimate diagnosis even for medical experts due to the complexity and non-linearity of the relationships between the large measured factors. It is possibly resolved with using AI algorithms. ANFIS is an AI algorithm which has the advantages of both fuzzy inference system and neural networks. Therefore, it can deal with ambiguous data and learn from the past data. Applying ANFIS as a diagnostic system was considered in our experiment. In addition, the computational performance of diagnosis system is an important issue as well as the output correctness of the inference system. Methods of using recommended inputs generated by the Genetic-Algorithm, Decision-Tree and Correlation-Coefficient computation with ANFIS was proposed to reduce the computational overhead.

1 Introduction

A major problem in medial science is attaining the correct diagnosis of disease in precedence of its treatment. For the ultimate diagnosis, many tests are generally involved. All of these test procedures are said to be necessary in order to reach the ultimate diagnosis. However on the other hand, too many tests could complicate the main diagnosis process so that even the medical experts might have difficulty obtaining end results from those tests. Particularly in the case of where there are many tests performed for the ultimate diagnosis, wherein the results are not so relevant the diagnosis process could be even more complex even for medical experts.

This kind of difficulty could be resolved with the aid of a computerized diagnosis system. A well-designed computerized diagnosis system could be used to directly attain the ultimate diagnosis with the aid of several artificial intelligent algorithms which perform roles as classifiers.

There have been a substantial previous research works with WBCD database to achieve an automatic ultimate diagnostic system. Genetic Algorithm (GA) [10], Fuzzy Inference System (FIS) [7],[12], Neural Networks [9], Adaptive Boosting (AdaBoost) [2] and Neuro-Fuzzy Hybrid Models [9],[11] have been applied to this problem. The performances of each inference system were evaluated with calculating the degree of correctness in predicted results against diagnosed results represented as PPV (Positive Predicted Value) in each work. Each system shows the PPV within the range from less than 60% (AdaBoost) up to over 95% (Neuro-Fuzzy Hybrid Models). Among those algorithms, Neuro-Fuzzy Hybrid models provide relatively remarkable performances in diagnosis. Those models are the combination of Neural Networks and Fuzzy Inference Systems encouraging the advantages and resolving the drawbacks of both NNs and FIS models. For our experiments, a modified method of using Adaptive Neuro-Fuzzy Inference System (ANFIS) [1] was mainly applied to attaining the ultimate diagnosis and Adaptive Boosting (AdaBoost) [2],[6] was also tested for the comparison. These 2 algorithms are available as functioning classifiers so that they are used for the intelligent diagnosis system which attains the ultimate diagnosis as being either *benign* or *malignant*.

Although the ultimate diagnosis is possibly attained by the computerized diagnosis system, important consideration as to how reliable the ultimate diagnosis is and the availability of the inference system could be used in the real medical field. Therefore a good model of breast cancer diagnosis system is concluded to have high accuracy on the diagnosis and low cost in computational and storage devices.

In this point of view, we focused our experiments on designing a diagnosis system which has a lower computational cost with highest possible accuracy on providing the diagnosis. In applying ANFIS and other artificial intelligent algorithm, the required system size is in proportion to the size of the inference system such as the number of inputs, the number of internal nodes and the number of learning iteration. Among those critical factors of the inference system, the number of internal nodes and learning iteration are changeable only in the process of designing the system. Therefore, methods for reducing the number of input factors within range of not losing the accuracy of diagnosis were considered. For the purpose, we used these 3 methods as the pre-processes / input-recommenders, which are Genetic Algorithm [3], Decision Tree [4] and the correlation coefficient between the individual inputs and the test diagnosis results. Also, ANFIS and AdaBoost algorithms are used for establishing the main part of the diagnosis system.

The rest of this paper is organized as follows: in Section 2, WBCD database is introduced and Section 3 briefly describes the ANFIS structure. In Section 4, ANFIS and its combinational way of using with input recommenders is proposed. The experiment result is given and compared with previous research works in Section 5.

2 Wisconsin Breast Cancer Diagnosis (WBCD) Database

Breast cancer is the most common tumor-related disease among women in Korea and throughout the world, and the mortality rate caused by breast cancer is dramatically increasing. It is considered to be the major cause to death to women, seriously threatening women health. In the diagnosis process, fine needle aspiration of breast masses is a mostly non-invasive diagnostic test that obtains information needed to evaluate malignancy [5],[7].

The Wisconsin breast cancer diagnosis (WBCD) database is the result of efforts provided by the University of Wisconsin Hospital based on microscopic examination of breast masses with fine needle aspirate tests. The WBCD database consists of nine measures represented as an 1-10 integer value as follows: (1) *Clump Thickness*(X_1); (2) *Uniformity of Cell Size* (X_2) (3) *Uniformity of Cell Shape* (X_3) (4) *Marginal Adhesion* (X_4) (5) *Single Epithelial Cell Size* (X_5) (6) *Bare Nuclei* (X_6) (7) *Bland Chromatin* (X_7) (8) *Normal Nucleoli* (X_8) (9) *Mitosis* (X_9). The database itself contains 683 cases, with each entry representing the classification for a certain ensemble of measured values:

Table 1. WBCD database.

	case	X_1	X_2	X_3	…	X_9	*Diagnostics*
	1	5	1	1	…	1	*Benign*
Training data	2	3	2	2	…	1	*Malignant*
	:	:	:	:		:	:
	341	4	8	8	…	1	*Malignant*
	342	6	6	6	…	2	*Benign*
Test data	:	:	:	:		:	:
	683	4	8	8	…	1	*Malignant*

Note that the diagnostics do not provide any information about the degree of benignity or malignancy. In considering of the relationship between the measured values and the diagnostics, there are almost no relationships which stand out. Therefore, there is no convenient and effective method to attain the ultimate diagnostics with this original data even for the specialists. In addition to that, there may be the possibility that one or more of the measured pieces do not affect the diagnosis result. These are the reasons that artificial intelligent system can be used as an expert to assist the specialist in diagnosing correctly.

In our experiments, the data in the WBCD database was divided into 2 sets: training and test datasets. The datasets are normalized to the range [0, 1]. The output is classified with the following classification rules:

$$Class = \begin{cases} Benign, & if\ y < 0.5 \\ Malignant, & if\ y \geq 0.5 \end{cases} . \tag{1}$$

The training dataset was used to figure out what were the most effective and dominant inputs of the inference system. They result in: genetic algorithm, decision tree and correlation coefficient computation, as well training the system with the recommended inputs using ANFIS and AdaBoost algorithms. The test dataset was used for correctness verification of the output.

3 Diagnostic System Using ANFIS for Breast Cancer

A human-like learning and decision making system has been mostly modeled with fuzzy system. [7] The existing fuzzy inference system block diagram is shown in (Fig. 4). However, the fuzzy inference system has several weak points in its learning algorithm. They are:

- No standard methods exist for transforming human knowledge or experience into the rule base and database of a fuzzy inference system.
- There is a need for effective methods for tuning the membership functions so as to minimize the output error measure or maximize performance index.

The Adaptive Neuro-Fuzzy Inference System (ANFIS), proposed by J. S. R. Jang [1] is an alternate method which combines the advantages of both fuzzy rule based system and neural networks. It is a hybrid learning algorithm which is a fuzzy inference system implemented in the framework of adaptive neural networks. (Fig. 4) shows the block diagram of ANFIS structure. As shown in the diagram, ANFIS uses a hybrid learning rules to optimize the fuzzy system parameters. In the forward pass, functional signals go forward till layer 4 and the consequent parameters are identified by the least squares estimate (LSE). In the backward pass, the error rates propagate backward and the premise parameters are updated by the gradient descent.

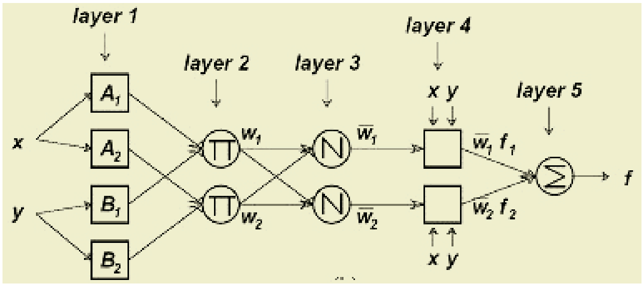


Fig. 1. ANFIS structure diagram.

For simplicity, the fuzzy inference system having only 2 inputs and 1 output was depicted in (Fig. 1). In our experiment, an ANFIS structure containing 4 to 9 inputs and 1 output was used. The ANFIS system has the inference function of fuzzy system and the learning function of neural networks. Supposing that the rule base contains a fuzzy if-then rule, the function in each layer is described below.

Rule1: IF x ***is*** A_1 ***and*** y ***is*** B_1***, then*** $f_1 = p_1x + q_1y + r_1$

Rule2: IF x ***is*** A_2 ***and*** y ***is*** B_2***, then*** $f_2 = p_2x + q_2y + r_2$

Layer 1: Membership functions (MF) are formed at each node in layer 1. Usually bell-shaped MFs are chosen with maximum equal to 1 and minimum equal to 0, such as,

$$O_i^1 = \mu_{A_i}(x),\ \mu_{A_i}(x) = \frac{1}{1+\left[\left(\frac{x-c_i}{a_i}\right)^2\right]b_i} \ or\ \mu_{A_i}(x) = \exp\left\{1-\left(\frac{x-c_i}{a_i}\right)^2\right\} \tag{2}$$

where, $\{\, a_i, b_i, c_i \,\}$ *is parameter set (premised parameters) .*

Layer 2: The firing strengths are generated in layer 2.

$$w_i = \mu_{A_i}(x) \times \mu_{B_i}(y) \qquad i = 1,2\ . \tag{3}$$

Layer 3: Normalizing the firing strengths

$$\bar{w}_i = \frac{w_i}{w_1 + w_2} \qquad i = 1,2\ . \tag{4}$$

Layer 4: Every node in layer 4 has parameter referred to consequence parameters

$$O_i^4 = \bar{w}_i f_i = \bar{w}_i(p_i x + q_i y + r_i)$$
$$where\ \{p_i, q_i, r_i\}\ is\ the\ parameter\ set\ . \tag{5}$$

Layer 5: The overall output is computed as the summation of all signals

$$O_1^5 = z = \sum_i \bar{w}_i f_i = \frac{\sum_i \bar{w}_i f}{\sum_i \bar{w}_i}\ . \tag{6}$$

In this procedure, an ANFIS system can learn from the existing data with correct output and possibly predict a result with new input set by tuning its internal node values. In our experiments, ANFIS is used with 9 inputs (full database) or less inputs from the WBCD database and generates 1 output (benign or malignant). The experiment result data with ANFIS is shown in section 5.

4 ANFIS Diagnostic System with Dominant Input Selectors

4.1 ANFIS with GA Input Recommender

Genetic Algorithm (GA) was used as a dominant input selector. In out experiment, a GA input recommender system was used to have the optimized input value set with ANFIS and AdaBoost test error estimation as its object function. From the result of preceding cost evaluation experiment, the maximum number of recommended inputs by GA was restricted up to 6 measured data inputs. (Fig. 2) describes how the GA is used for dominant input feature selecting process. Each individual in the population consists of 9 strings which represent the candidate inputs for the output classification algorithm such as ANFIS and AdaBoost. Each string in an individual is represented as either 0 or 1: 0 for not use the input data, 1 for use the input data. Since we limited the maximum number of recommended inputs up to 6, the number of all cases to be considered is calculated as

$$\sum_{n=9,r=1}^{r=6} \frac{n!}{(n-r)!r!} = 456\ (cases) \cdot \tag{7}$$

With the assumption that there is no re-appearance of individual which is already considered, the number of generations will be calculated as

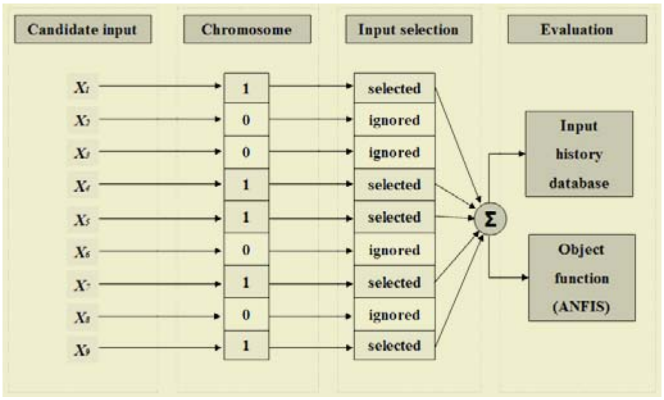
$$number\ of\ generations = \frac{num.\ of\ all\ cases}{num.\ of\ cases\ in\ 1st\ generation} = \frac{456}{10} \doteq 46 \cdot \tag{8}$$

Therefore the relevant parameter settings are:

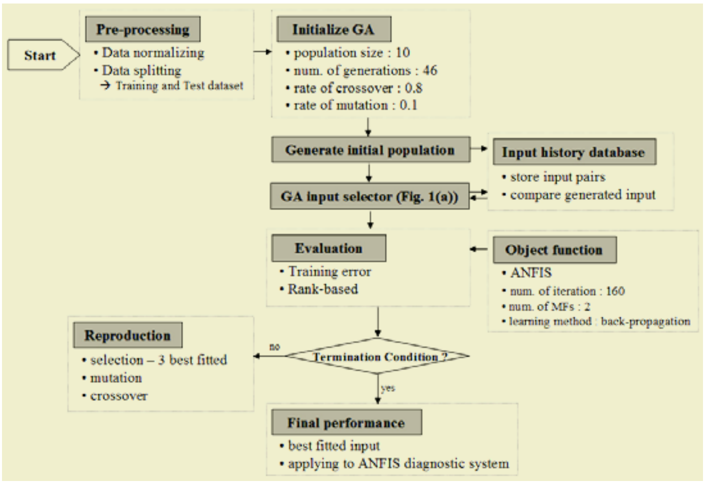
Table 2. Parameter setting for GA input recommender.

Population size	10
Number of generations	46
Probability of crossover	0.9
Probability of mutation	0.1

The brief block diagram of the GA input recommender is shown in (Fig. 2). Once the GA input recommender generates 10 individuals in a generation, the inputs are applied to the output classification algorithm and the 10 output errors between the computed output values by the output generator and the real diagnosis results. The output generators such as ANFIS or AdaBoost have restricted values of their parameters – iteration number in ANFIS and the number of hypotheses in AdaBoost for the fast computation time. Then the GA ranks the candidates in the current generation and remains the best fitted result as the parent of the next generation. Then, the 9 other candidates are selected with another random input generation process, crossover and mutation process. Finally, the GA input recommender attains the best input combination for the diagnosis process through this procedure. The result derived by this experiment with GA is shown in section 5.



(a) GA input selector



(b) GA input recommender

Fig. 2. Block diagram of GA input recommender.

4.2 Decision Tree

The second input recommender used in our experiment was a decision tree learning algorithm using SAS9™ package. A decision tree (DT) is known as a good classifier of huge data. It classifies input data by partitioning example spaces with entropy calculation. DT is especially useful for these cases: examples are represented by attribute-value pairs and the target function has discrete output value. The WBCD consists of 9 measured data which are represented as an integer between 1 and 10. Therefore

DT can be a good classifier for WBCD dataset. In our experiment, a binary decision tree was constructed to select dominant inputs. The diagram of decision tree is presented in (Fig. 2). In each node of the tree, the most useful attribute for classifying whole data is selected by calculating the information gain with following formula.

$$Gain(D, X) \equiv Entropy(D) - \sum_{v \in Values(X)} \frac{|Dv|}{|D|} Entropy(Dv) \cdot \tag{9}$$

$$where,\ Entropy(D) \equiv \sum_{i=1}^{c} -p_i \log p_i \cdot \tag{10}$$

The DT is constructed in a way to reduce the entropy with the attribute which has the highest gain value at each node. Through this way, the final DT model has the most useful measured data on the top node, next useful one on the right node of the top node and so on. The input selection result derived by DT is presented in (Fig. 3).

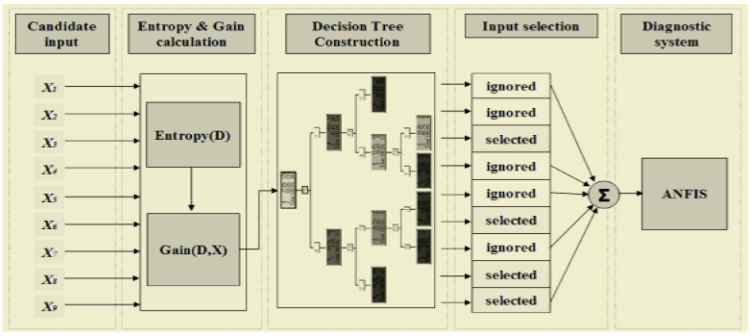


Fig. 3. DT dominant input selection for ANFIS diagnostic system structure diagram.

4.3 ANFIS with Correlation Coefficient Computation

The third method used for dominant input selecting process is calculating correlation coefficients between each measured input data and the diagnosis results. The Correlation Coefficient is a numerical measure of the degree of the linear relationship between two variables. The value of the correlation coefficient always falls between –1 and 1. A positive correlation coefficient indicates a direct relationship, and a negative correlation coefficient indicates an inverse relationship between two variables. The closer to 1 the correlation coefficient is, the stronger relationship the two data have and a value near 0 indicates no relationship. In the calculation, first it is possible to assume that all the correlation coefficients by calculating with data in WBCD should be positive. Then we selected 4 measured input data from the one which has the highest correlation coefficient for diagnosis system. The correlation coefficient can be calculated by this following formula.

$$\rho_{ij} = \frac{C_{ij}}{\sqrt{(C_{ii}C_{jj})}} = \frac{cov(X_i, X_j)}{\sigma(X_i)\sigma(X_j)} \cdot \tag{11}$$

$$where,\ Cii = \sum \left(Xi - \bar{X}\right)^2,\ Cjj = \sum \left(Xj - \bar{X}\right)^2,\ Cij = \sum \left(Xi - \bar{X}\right)\left(Xj - \bar{X}\right) \cdot \tag{12}$$

Each result by the correlation coefficient calculation between each measured input and the correct output indicates the degree of linear relationship between them. The inputs highly correlated with the output were selected as dominant inputs in our experiment. The result of dominant input selection by the correlation coefficient calculation is given in (Table. 3).

Table 3. Result of calculating Correlation coefficient (computed by SAS9™ package).

	X_1	X_2	X_3	X_4	X_5	$X_{6,}$	X_7	X_8	X_9
Correlation with X_{10}	0.6974	0.9438	0.9058	0.8215	0.7953	0.8828	0.9257	0.7924	0.4274
Rank	8	1	3	5	6	4	2	7	9

5 Experiment Results

In our experiment, we focused on the cost down in computation time and data storage. Therefore, the experiment about the computation time with the number of inputs from 2 to 9 and the number of nodes (required memory) using ANFIS are conducted in advance. (Table. 4) shows the result of computational cost evaluation tests using ANFIS with various numbers of inputs. As shown in the table, after 7 inputs applied, the computation time and the needed data storage increase significantly so the input selection method could be appropriate for faster result and costing down as long as the system is not losing the diagnostic accuracy.

Table 4. Result of computational cost evaluation tests using ANFIS.

number of inputs	2	3	4	5	6	7	8	9
computation Time with ANFIS (minutes)	0.1	0.2	0.4	1.1	1.5	4.4	13.1	37.3
number of nodes	21	34	55	92	161	294	555	1072
number of parameters	24	50	104	222	484	1066	2352	5174
Number of fuzzy rules	4	8	16	32	64	128	256	512

(*All tests were conducted with Pentium4™ 2.0Ghz, 256 Mb memory system using MATLAB™ r13*)

The recommended inputs are presented in the following table in each case.

Table 5. Recommended inputs by each input recommender.

		Selected input
Genetic Algorithm	*With ANFIS*	$X_1, X_4, X_5, X_6, X_7, X_8$
	With AdaBoost	X_1, X_5, X_6, X_7, X_8
	Decision tree	X_6, X_3, X_7, X_8
	Correlation coefficient	X_2, X_7, X_3, X_6

These selected input dataset were applied to ANFIS and AdaBoost inference system to train and test the proposed system. The structure of ANFIS used in the experiment is shown in (Fig. 1) in Section 3 with expanded inputs between 4 and 6.

To evaluate the correctness of the proposed system, PPV (positive predicted value) was computed in each case. (Table. 6) shows the experiment result with ANFIS and the following table presents PPVs in each case of experiment. (Table. 7) is given for the comparison between our experimental result and previous works. The results conducted with AdaBoost are presented to compare with the ANFIS results in our experiment. In the simulation with ANFIS, the iteration number is set to 300 and the number of hypotheses with AdaBoost is set to 30. PPV is computed as:

$$PPV = \frac{Correct\ results}{All\ results} \times 100\ (percent)\ . \tag{13}$$

Table 6. Experimental results using ANFIS and AdaBoost with recommended inputs.

	Experimented case		*Number of correct results*	*PPV (percent)*
ANFIS	*Full data*	9	327 / 341	95.89
	Genetic algorithm	6	333 / 341	97.65
	Decision tree	4	334 / 341	97.95
	Correlation coefficient	4	332 / 341	97.36
AdaBoost	*Full data*	9	211 / 341	61.87
	Genetic algorithm	5	212 / 341	62.17
	Decision tree	4	208 / 341	61.01
	Correlation coefficient	4	215 / 341	63.05

Table 7. Experimental results of previous works.

	Experiment dataset	*PPV (percent)*	*Reference*
ANFIS	USF	52.9	[8]
AdaBoost	USF	57.6	[8]
SIANN	WBCD	100	[9]
Fuzzy-Genetic	WBCD	97.07	[10]
ILFN	WBCD	97.23	[11]
Fuzzy	WBCD	96.71	[11]
ILFN &Fuzzy	WBCD	98.13	[11]
SANFIS	WBCD	96.07~96.3	[12]
NNs	WBCD	97.95	[13]

In our experiment, ANFIS shows even better performance than AdaBoost as an inference system on breast cancer diagnosis problem. And the reduced input dataset shows almost the same performances or better performances with the same learning iteration number and shows better/similar performance against the results of previous works. Even though the result derived by the reduced input dataset shows little worse performance, if it has significantly higher advantage in computation, it would be a better method to be implemented in real situations. Therefore, the proposed methods – combined algorithm with ANFIS and dominant input data recommenders, can be appropriate methods of inference system for the problem of breast cancer diagnosis.

6 Conclusions

In this paper, methods of automatic breast cancer diagnosis system with ANFIS were proposed with experiments focused on computational performance improving by input data selection. Artificial intelligence algorithms are highly available for automatic diagnosis systems in real medical field. However, slow performance caused by huge computations and required storage device are still remained as a problem to be solved. Our experiment indicates a way to have higher computational performance with these powerful inference system algorithms with providing reliable results. In addition to those benefits which mentioned in this paper, the method using ANFIS with input reduction can be used for many other problems which have high complexity and strong non-linearity with huge data to be analyzed.

References

1. Jang, J.R.: ANFIS: Adaptive-Network Based Fuzzy Inference System. IEEE Trans. on System, Man and Cybernetics, **23** (1993)
2. Freund, Y., Schapire, R.E.: Experiments with a New Boosting Algorithm. Machine Learning: Proceedings of the Thirteenth International Conference (1996)
3. Goldberg, D.: Genetic Algorithm in Search, Optimization, and Machine Learning. Addison-Wesley (1989)
4. George, H.J. Kohavi, R., Pfleger, K.: Irrelevant Features and the Subset Selection Problem. Machine Learning: Proceedings of the Eleventh International Conference (1994)
5. Mangasarian, O.L., Street, W.N., Wolberg, W.H.: Breast Cancer Diagnosis and Prognosis Via Linear Programming. Mathematical Programming Technical Report 9410, University of Wisconsin (1994)
6. Schapire, R.E.: Theoretical Views of Boosting and Applications. Proceedings of Algorithmic Learning Theory (1999)
7. Pena-Reyes, C. A., Sipper, M.: Evolving Fuzzy Rules for Breast Cancer Diagnosis. In Proceedings of 1998 International Symposium on Nonlinear Theory and Applications (1998)
8. Land. Jr., W.H., Veheggen, E.A.: Experiments Using an Evolutionary Programmed Neural Network with Adaptive Boosting for Computer Aided Diagnosis of Breast Cancer. IEEE International Workshop on Soft Computing in Industrial Application (2003)
9. Arulampalam, G., Bouzerdoum, A.: Application of Shunting Inhibitory Artificial Neural Networks to Medical Diagnosis. Seventh Australian and New Zealand Intelligent Information Systems Conference (2001)
10. Pena-Reyes, C. A., Sipper, M.: Designing Breast Cancer Diagnostic System via a Hybrid Fuzzy-Genetic Methodology. IEEE International Fuzzy Systems Conference Proceeding (1999)
11. Meesad, P., Yen, G.G.: Combined Numerical and Linguistic Knowledge Representation and Its Application to Medical Diagnosis. IEEE Transactions on Systems, Man, and Cybenatics (2003)
12. Wang, J.S., Lee, G.C.S.: Self-Adaptive Neuro-Fuzzy Inference Systems for Classification Applications. IEEE Transactions on Fuzzy Systems (2002)
13. Setiono, R.: Generating Concise and Accurate Classification Rules for Breast Cancer Diagnosis. Artificial Intelligence in Medicine (2000)

Spiculated Lesion Detection in Digital Mammogram Based on Artificial Neural Network Ensemble

Ning Li, Huajie Zhou, Jinjiang Ling, and Zhihua Zhou

National Laboratory for Novel Software Technology,
Nanjing University, Nanjing, Jiangshu 210093, China
{ln,zhouzh}@nju.edu.cn

Abstract. Among breast abnormalities, spiculated lesions are one of the most difficult type of tumor to detect. In this paper, we apply a feature extraction method to generate four feature images for a single mammogram, and then partition every feature image into a series of small square blocks. The four average feature values of each block are considered as an instance describing the block. Finally we use an artificial neural network ensemble method to detect the spiculated lesions. Experiments show that the accuracy of this method is well on digital mammograms.

1 Introduction

Breast cancer is the second major cause of death in women, exceeded only by lung cancer. Early detection and treatment of breast cancer helps save lives, where screening mammography is considered to be the most effective tool [1]. With the rapidly increasing amount of mammograms, computer-aided diagnosis systems have been shown to be useful in assisting radiologist. During the last two decades, many approaches have been proposed to detect and classify anomalies in breast images. Most of them involve enhancement, segmentation and classification. Among these approaches are wavelet-based segmentation, texture-based feature analysis, stochastic methods using Markov model and fractal model, fuzzy logic based method, ect. Classification methods mainly involve neural networks, Fisher linear classifier and pattern matching methods [2].

Among the masses, spiculated lesions are highly suspicious signs of breast cancer [3]. Liu *et al.* presented a feature extraction method based on multiresolution feature analysis [4]. In this paper, we apply Liu *et al.*'s method to generate four feature images for a single digital mammogram. Then we partition each feature image into a series of small squared blocks, whose features are the average values of the features in the block. Finally, we consider each small square block as an instance and use artificial neural network ensemble for classification.

We organize this paper as follows. In section 2, we briefly introduce artificial neural network ensemble. The feature extraction method used in this paper is described in section 3. Section 4 reports the experimental result. Finally we conclude and discuss future work in section 5.

2 Artificial Neural Network Ensemble

Artificial neural networks are very useful in pattern recognition. Hornik *et al.* [5] showed that feedforward artificial neural networks with one hidden layer can approximate any functions in any accuracy. However, the performance of the neural networks heavily depends on the user's experiences.

Recently, neural network ensemble becomes a hot topic of machine learning and neural networks. A neural network ensemble trains a collection of a finite number of neural networks for the same task [6]. Hansen and Salamon [7] showed that the generalization ability of an artificial neural network system could be significantly improved through ensembling artificial neural networks. Later, Krogh and Vedelsby [8] derived that the generalization ability of the ensemble is determined by the average generalization ability and the average ambiguity of the individual artificial neural networks.

In general, a neural network ensemble is built in two steps, i.e. generating component neural networks and then combining their predictions. As for generating component networks, subsampling methods such as *Boosting* [9] and *Bagging* [10] are prevailing methods. Boosting generates a series of component networks whose training sets are determined by the performance of the former networks in the way that training instances that are wrongly predicted by the former networks play more important role in the training of the later networks. Bagging generates many training sets from the original training set via bootstrap sampling and then trains a component network from each of those training sets. As for combining component predictions, *voting* is prevailing for classification, which regards the class label receiving the most number of votes as the final output of the ensemble. In cases where different component neural networks are with different importance, *weighted-voting* is often used.

Since artificial neural network ensembles work remarkably well and are easy to be used, they are regarded as a promising methodology that can profit not only experts in artificial neural network research but also engineers in real-world applications. It is noteworthy that the artificial neural network ensembles have already been introduced into the field of computer aided medical diagnosis [11],[12].

3 Feature Extraction

By definition, a spiculated mass is characterized by spiculations radiating from the margin of the mass [13]. Here we use the feature extraction method proposed by Liu *et al* [4] to generate four features for each pixel of digital mammogram.

Let (i,j) be the spatial location in the mammogram at row i and column j, $f(i,j)$ be the pixel brightness at (i,j), $N(i,j)$ be the neighborhood of (i,j), K be the number of pixels in $N(i,j)$.

The four features generated by Liu *et al.*'s method [4] are defined as follows:

1. Mean pixel brightness:

$$\overline{f(i,j)} = \frac{1}{K} \sum_{(m,n)\in N(i,j)} f(m,n) . \tag{1}$$

2. Standard deviation of pixel brightness:

$$\sigma_f(i,j) = \sqrt{\frac{1}{K-1} \sum_{(m,n)\in N(i,j)} (f(m,n) - \overline{f(i,j)})^2} . \tag{2}$$

3. Standard deviation of gradient orientation histogram:
This feature is just the same as the standard deviation of the edge orientation histogram (ALOE) described in [14].
In detail, let $dy(i,j)$ and $dx(i,j)$ be the vertical and horizontal spatial derivatives at (i,j) respectively. Let $\theta(i,j) = \arctan(dy(i,j)/dx(i,j))$ be the gradient orientation at (i,j) with value between $-\pi/2$ and $\pi/2$. Also let $histij$ be the histogram of θ in $N(i,j)$ using 256 bins, so the value of $histij(n)$ is the number of pixels in $N(i,j)$ with $\theta \in (-\pi/2 + n\pi/256, -\pi/2 + (n+1)\pi/256]$, where n = 0,1,2…,255. Let $\overline{hist(i,j)}$ be the average bin height of $histij$.
Then the standard deviation of gradient orientation histogram is defined as:
$$\sigma_{hist}(i,j) = \sqrt{\frac{1}{255}\sum_{n=0}^{n=255}(histij(n) - \overline{hist(i,j)})^2} \cdot \tag{3}$$

4. Standard deviation of the folded gradient orientation:
Let KP and KN be the number of positive and negative gradient orientations in $N(i,j)$ respectively. Let
$$\overline{\theta_+}(i,j) = \frac{1}{KP}\sum\nolimits_{\theta(m,n)\ge 0,(m.n)\in N(i,j)} \theta(m,n) \cdot \tag{4}$$
$$\overline{\theta_-}(i,j) = \frac{1}{KN}\sum\nolimits_{\theta(m,n)\le 0,(m.n)\in N(i,j)} \theta(m.n) \cdot \tag{5}$$
The folded gradient orientation is defined as:
$$\beta(i,j) = \begin{cases} \theta(i,j) + \pi & \text{if } (\overline{\theta_+}(i,j) - \theta(i,j)) > \frac{\pi}{2} \text{ and } KP \ge KN \\ \theta(i,j) - \pi & \text{if } (\theta(i,j) - \overline{\theta_-}(i,j)) > \frac{\pi}{2} \text{ and } KP < KN \\ \theta(i,j) & \text{otherwise} \end{cases} .$$
Then the standard deviation of the folded gradient orientation is defined as:
$$\sigma_\beta(i,j) = \sqrt{\frac{1}{K-1}\sum_{(m,n)\in N(i,j)} (\beta(m,n) - \overline{\beta(i,j)})^2} \cdot \tag{6}$$

Generally speaking, in breast images, pixels in normal areas are less brighter than those in spiculated lesion regions because lesions usually have higher density. The two features, $\overline{f}$ and σ_f, can help detect this character of spiculated lesions. Also, in normal regions, pixels usually have similar gradient orientations, while pixels near spiculated lesions tend to have gradient orientations in diverse directions, so the value of σ_β near the lesion regions is bigger. For the same reason, the value of σ_{hist} in the neighborhood of lesion is smaller than that in normal regions. Figure 1 gives an illustration, where the neighborhood is set to be a circular area containing 30 pixels in radius. Four feature images are obtained, in which each pixel value represents the responding feature value at the point. We can see that σ_{hist} and σ_β are sensitive to the spiculated lesions.

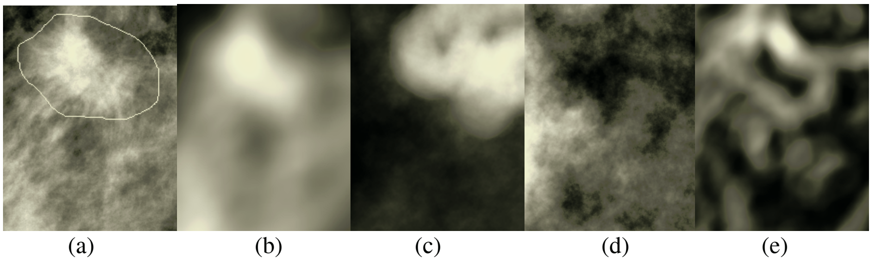


Fig. 1. From left to right are: (a) original mammogram with spiculated lesion, (b) feature image obtained by using the first feature, (c) feature image obtained by using the second feature, (d) feature image obtained by using the third feature and (e) feature image obtained by using the fourth feature.

4 Experiments and Results

In our experiments, fifty images are used. Considering that the data volume is very large because every image contains 1104000 pixels, which can be transformed to 11040 10*10 sized blocks, we divide the data to 10 partitions, i.e. each partition contains five images. 10-fold cross validation is performed on each partition, where ten neural networks are trained by *Bagging* [10] in each fold and each neural network has the same weight in voting. As for the neural networks, we choose BP networks, where the inputs are the four features of each block described above, and the output is the label of instances, i.e. whether the concerning block is positive or negative, which means whether it belongs to spiculated lesion area or not, and the hidden unit number is set to 7. Table 1 shows the result of our experiment.

Table 1 shows the generalization error of each fold in each partition. We can find that the average errors are all less than 1.8%, which means the accuracies are all higher than 98.2%, and the standard deviations are all less than or equal to 0.0014.

Table 1 also gives the ratio of positive instances and negative instances. It can be found that if all the unknown blocks are regarded as negative, then the classification error will be about 17.7% to 31.1%, which is significantly bigger than 1.8%. Therefore, the performance of our method is for better than this baseline.

5 Conclusions

This paper applies Liu *et al.*'s feature extraction method [4] and artificial neural network ensemble constructed by *Bagging* [10] to spiculated lesion detection in digital mammogram and obtains good result. In the future, we will compare the performance with other methods on digital mammogram. Moreover, we find that Liu *et al.*'s feature extraction method [4] is quite slow when dealing with high-resolution digital mammograms, which can not meet the requirement of real-time diagnosis. Therefore, another future work is to improve the feature extraction and instance generating method.

Table 1. Experimental results.

Number of images	1-5	6-10	11-15	16-20	21-25	26-30	31-35	36-40	41-45	46-50
Ratio of instances	.2647 / .7353	.1773 / .8227	.2402 / .7598	.2227 / .7773	.2726 / .7274	.3113 / .6887	.2716 / .7284	.2491 / .7509	.2890 / .7110	.2403 / .7597
Fold										
1	.0180	.0208	.0136	.0155	.0121	.0074	.0089	.0070	.0170	.0123
2	.0178	.0217	.0104	.0159	.0127	.0076	.0096	.0070	.0189	.0136
3	.0183	.0217	.0106	.0149	.0117	.0074	.0093	.0091	.0176	.0125
4	.0168	.0200	.0115	.0142	.0121	.0083	.0096	.0089	.0168	.0147
5	.0170	.0206	.0115	.0159	.0130	.0083	.0091	.0096	.0155	.0151
6	.0172	.0206	.0113	.0170	.0119	.0081	.0091	.0081	.0157	.0130
7	.0166	.0208	.0119	.0176	.0123	.0079	.0093	.0087	.0161	.0168
8	.0168	.0193	.0121	.0174	.0117	.0062	.0091	.0081	.0164	.0138
9	.0161	.0208	.0125	.0180	.0106	.0070	.0089	.0083	.0170	.0140
10	.0168	.0204	.0132	.0168	.0115	.0070	.0087	.0085	.0161	.0125
ave	.0171	.0171	.0117	.0163	.0120	.0075	.0092	.0083	.0167	.0138
std	.0007	.0007	.0011	.0012	.0007	.0007	.0003	.0008	.0010	.0014

Acknowledgements

This work was supported by the Jiangsu Science Foundation Key Project, under the Grant No. BK2004001.

References

1. American Cancer Society: Cancer Facts And Figures 2003: Technical Report, American Cancer Society, Atlanta, GA (2003)
2. Christoyianni, I., Koutras, A., Dermatas, E., Kokkinakis, G.: Computer Aided Diagnosis of Breast Cancer in Digitized Mammograms. Computerized Medical Imaging and Graphics, 26 (2002) 309–319.
3. Kophans, D.B.: Breast Imaging. Lippincott Williams, 2nd edition (1998)
4. Liu, S., Babbs, C.F., Delp, E.J.: Multiresolution Detection of Spiculated Lesions in Digital Mammograms. IEEE Transactions on Image Processing, 10 (2001) 874-884
5. Hornik, K.M., Stinchcombe, M., White, H.: Multilayer Feedforward Networks Are Universal Approximators. Neural Networks, **2** (1989) 359-366
6. Sollich, P., Krogh, A.: Learning with Ensembles: How Over-Fitting Can Be Useful. In: Touretzky, D., Mozer, M., Hasselmo, M. (eds.): Advances in Neural Information Processing Systems 8, Cambridge, MA: MIT Press (1996) 190-196
7. Hansen, L.K., Salamon, P.: Neural Network Ensembles. IEEE Trans. Pattern Analysis and Machine Intelligence, **12** (1990) 993-1001
8. Krogh, A., Vedelsby, J.: Neural Network Ensembles, Cross Validation, and Active Learning. In: Tesauro, G., Touretzky, D., Leen, T. (eds.). Advances in Neural Information Processing Systems 7, Cambridge, MA: MIT Press (1995) 231-238
9. Schapire, R.E.: The Strength of Weak Learnability. Machine Learning, **5** (1990) 197-227
10. Breiman L.: Bagging Predictors. Machine Learning, **24** (1996) 123-140
11. Zhou, Z.-H., Jiang, Y., Yang ,Y.-B., Chen, S.-F.: Lung Cancer Cell Identification Based on Artificial Neural Network Ensembles. Artificial Intelligence in Medicine, 24 (2002) 25-36

12. Zhou, Z.H., Jiang, Y.: Medical Diagnosis with C4.5 Rule Preceded by Artificial Neural Network Ensemble. IEEE Transactions on Information Technology in Biomedicine, **7** (2003) 37-42
13. American College of Radiology: Illustrated Breast Imaging Reporting and Data System, Reston(VA): American College of Radiology, 3nd edition (1998)
14. Kegelmeyer Jr., W.P., Prundeda, J.M., Bourland, P.D., Hillis, A., Riggs, M.W., Nipper, M.L: Computer-aided Mammographic Screening for Spiculated Lesions. Radiology, **191** (1994) 331-336

Classification of Psychiatric Disorders Using Artificial Neural Network

Shishir Bashyal

Department of Computer and Electronics Engineering,
Nepal Engineering College, Pokhara University,
PO Box 10210, Kathmandu, Nepal
sbashyal@yahoo.com

Abstract. One fourth of the world population is affected by mental disorders during their lives. Due to lack of distinct etiology, classification of such mental disorder is based on signs and symptoms and is vulnerable to errors. In this paper, an Artificial Neural Network (ANN) classifier is proposed that is trained using past classification data so that it can correctly classify new patients based on their signs and symptoms. A set of signs and symptoms to be used as feature input has been identified. A multilayer neural network with a single hidden layer is used for the purpose of classification. The average accuracy of the proposed classifier when trained with the past data of 60 patients is found to be 96.5%. The ANN output is to be used for validating the classification of an expert so that the reliability of the traditional classification process is improved.

1 Introduction

A psychiatric disorder is a disturbance of Cognition, Conation, Affect or any disequilibrium between the three. Classification of any psychiatric disorders, like that of medical illnesses, should ideally be based on etiology. Since a large majority of psychiatric disorders lack known distinct etiology, the only rational way to classify them is using syndromes [1],[2]. A psychiatrist must understand the global view of the patient for proper classification. In practice the classification largely depends on how the patients present at interview [2]. As the classification of a patient with this approach is not much reliable, a common consent of experts is desirable.

In the present work, a supervised artificial neural network (ANN) is developed to classify the disorder depending on signs and symptoms of the patient. A list of signs and symptoms has been identified that can be successfully used in classification of psychiatric disorders. The classification result of the ANN classifier is proposed to be used for verifying the classification of an expert so that the reliability of the traditional disorder classification process is improved.

2 Method for Classification

Psychiatric classification is usually based on the feature of disorders [2]. In recent times, ANN has become a widely used method for medical diagnostics [3]. Several applications of ANN in medical diagnosis have been successful [3],[4]. In this work, a multi-layer feed forward network is used to classify the psychiatric disorder. The block diagram of the proposed classifier is shown in Fig. 1.

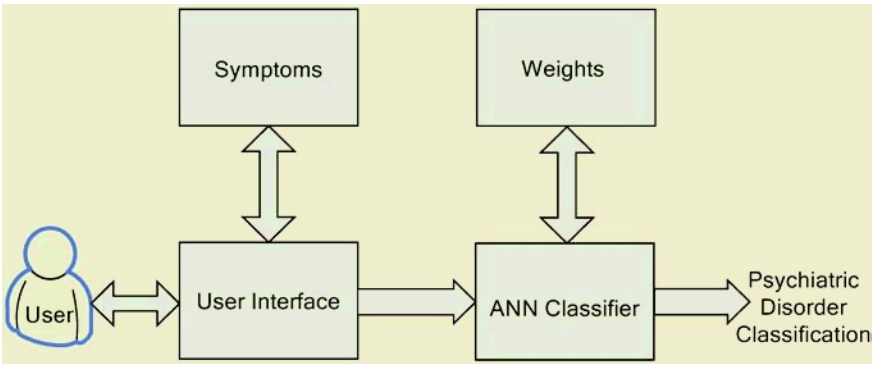


Fig. 1. Block diagram of the proposed classifier.

For the scope of this work, the accuracy of a classifier system is defined as the ratio of number of correct classification to the total number of classification tests.

2.1 Classes of Psychiatric Disorders

At present, there are two major classifications in Psychiatry, namely International Classification of Diseases- 10th Revision (ICD-10) and Diagnostic and statistical Manual of Mental Disorders-IV edition (DSM-IV-TR) [1]. Both of the standards classify the disorders based on the symptoms of the patient. In this work, only the four most frequent classes Schizophrenia, Mania, Depression and Alcohol Dependence Syndrome (ADS) have been dealt with.

2.2 Signs and Symptoms

35 different signs and symptoms were identified that could be used as a feature input to the neural network. In both the training and testing data, the symptoms with missing value were considered to have a normal value (for example if nothing was mentioned about appetite of the patient, the encoding of apetite would be considered to be 1 i.e. average). The list of signs and symptoms along with the encoding technique used are listed in Table 1.

2.3 ANN

A multilayer feed-forward network using Error Back Propagation algorithm is used for training the classifier. Each node of the network implements a sigmoid activation function. The number of nodes in the input layer is 35 (which is equal to number of symptoms), that in hidden layer is 117 and the number of nodes in the output layer is 4 (equal to the number of classes). The weight of each node is initialized randomly and updated using back-propagation algorithm. The average time taken by the ANN to converge to an accuracy of more that 99.5 (for training data) is less than four minutes (on a Pentium III computer running at 800 MHz). The ANN classifier has been successfully implemented in Turbo C 3.0.

Table 1. List of Signs and Symptoms that are input to the Neural Network.

SN	Sign or Symptom	Encoding	SN	Sign or Symptom	Encoding
1	Concentration	Good – 1 Poor – 0	19	Alcohol Drinker	Regular-1 Otherwise- 0
2	Delusion	Yes – 1 No – 0	20	Alcohol Withdrawal Syndrome	Yes – 1 No – 0
3	Destructive	Yes – 1 No – 0	21	Anger	High – 1 Low – 0
4	Fear	Yes – 1 No – 0	22	Appetite	Avg. – 1 Otherwise-0
5	Flight of Ideas	Yes – 1 No – 0	23	Body Pain or Itches	Yes – 1 No – 0
6	General Condition	Good – 1 Poor – 0	24	Delayed Response	Yes – 1 No – 0
7	Grandiosity	Yes – 1 No – 0	25	Headache	Yes – 1 No – 0
8	Hallucination	Yes – 1 No – 0	26	Insight	Avg. – 1 Partial – 0.5 No – 0
9	Lethargy	Yes – 1 No – 0	27	Insomnia	Yes – 1 No – 0
10	Volume of Speech	High – 1 Avg. – 0.5 Low – 0	28	Mood	Elated – 1 Avg. – 0.5 Depressed -0
11	Memory	Good – 1 Poor – 0	29	Obedient, cooperative	Yes – 1 No – 0
12	Personal Hygiene	Good – 1 Poor – 0	30	Orientation	Avg. – 1 Partial – 0.5 No – 0
13	Talkative, Crying, Shouting and Singing	High – 1 Avg. – 0.5 Low – 0	31	Psychomotor Activity	High – 1 Avg. – 0.5 Low – 0
14	Sadness	Yes – 1 No – 0	32	Sex	Male – 1 Female – 0
15	Suicidal Tendency	Yes – 1 No – 0	33	Sleep	Avg. – 1 Otherwise– 0
16	Runs away	Yes – 1 No – 0	34	Suspicious	Yes – 1 No – 0
17	Talks to Self	Yes – 1 No – 0	35	Walking without purpose	Yes – 1 No – 0
18	Low feeling	Yes – 1 No – 0			

2.4 Training and Testing Data Sets

Supervised learning in Neural Network requires standard data for training. A set of 80 medical records and their corresponding diagnosis was obtained from Psychiatry Ward of the Patan Hospital, Lalitpur, Nepal. In the set of 80 records, each of the four disorders had 20 representative cases belonging to their class. Out of the 20 representative data belonging to a disorder, 15 randomly selected data were used for training and the remaining 5 were isolated for testing.

Some of the records in the data set did not have specified values for a number of signs and symptoms. But both training and testing if the ANN requires that all 35 signs and symptoms have some value. In such cases, the signs and symptoms with missing values were assigned a normal value assuming that if there had been any abnormalities; it would be mentioned in the medical records.

3 Results and Discussion

Initially, the ANN is trained with a set of 60 data using supervised learning algorithm. The training is continued until and unless the accuracy of classification for the training data set is more than 99.5%. Once the training is over, the classifier is tested for its performance in retrieval mode in which testing data do not take part in training.

Ten different instances of the ANN have been trained using the same training data (but different initial weights) and each instance have been tested using the same set of testing data. The performance of the ANN classifier is summarized in Table 2.

The average overall accuracy of the classifier was found to be 96.5%. Close analysis of some misclassifications showed that the misclassifications are due to presence of some unconvincing signs and symptoms in the test data (for example a test data of Depression was found to have Grandiosity =1 while no training data had that feature). In practice, as the classifier is to be used as a consultation tool, the mismatch would urge the expert to analyze the classification to a greater depth.

Table 2. Performance of ANN classifier in testing phase.

Class	Number of Correct Classifications	Total Number of Classifications	Accuracy (%)
Schizophrenia	49	50	98
Mania	47	50	94
Depression	48	50	96
ADS	49	50	98

4 Limitations and Further Work

The present work incorporates only four psychiatric disorders. Classification of other different disorders can also be addressed by extending the list of signs and symptoms and training the ANN with additional set of data.

The training and test data both were obtained from a single hospital and thus the learning of the ANN is based on the skills of the psychiatrists in the hospital. Data from several hospitals should be used for training and testing to ensure proper functioning of the ANN.

5 Conclusions

The result of the present work demonstrates the possibility of successfully classifying psychiatric disorder using a multilayer feedforward ANN that accepts signs and symptoms as the feature input. The classifier uses the same set of signs and symptoms to classify all patients thereby reducing the chance of misclassification due to lack of information. As the proposed algorithm provides an accuracy of about 96.5% and as no sophisticated computational systems are required for its execution, the proposed work could be useful in clinical application.

References

1. Ahuja, N.: A Short Textbook of Psychiatry. JAYPEE Publication, India (2002)
2. Mitchell, A.J.: Principles of Psychiatric Classification. University Department of Psychiatry, Leeds
3. Ozyilmaz, L., Yildirim, T.: Artificial Neural Network for Diagnosis of Hepatitis Disease. IJCNN (2003)
4. Jain, V., Sahambi, J.S.: Neural Network and Wavelets in Arrhythmia Classification. In: Manandhar S., Austin, J., Desai, U., Oyanagi, Y., Talukder, A. (eds.): Applied Computing. Lecture Notes in Computer Science, Vol. 3285. Springer-Verlag, Berlin Heidelberg New York (1997) 415–438

Multilevel Neural Network to Diagnosis Procedure of Traditional Chinese Medicine*

Zhanquan Sun, Jianqiang Yi, and Guangcheng Xi

Key Laboratory of Complex Systems and Intelligence Science,
Institute of Automation, Academy of Sciences, 100080, Beijing, China
{zhanquan.sun,jianqiang.yi,guangcheng.xi}@mail.ia.ac.cn

Abstract. In Traditional Chinese Medicine (TCM), it is difficult to interpret how to judge syndrome according to clinic information in scientific means because TCM depends mostly on experience and subjective judgment. Now there are more and more people begin to study TCM with modern techniques in many countries. Neural network techniques have developed quickly during these years and many complex problems are solved with them. It is feasible to discover the hidden mechanism of TCM with neuron network. In this paper a multilevel neural network model is developed to dispose TCM problem. In this model, Principle Component Analysis network is used to preprocess symptoms and back-propagation network is used to give out syndromes and Boltzmann machine is used to give out prescription. After training, the network can realize the whole diagnosis procedure. The feasibility is illustrated through simulation example with stroke patients' clinical data.

1 Introduction

Neural network is an advanced multidisciplinary technique, which relates with biology, computer, mathematics, physical knowledge and so on. It has been widely applied in many kinds of fields and many complex problems have been resolved with it successfully. Traditional Chinese Medicine is our country's traditional culture, which plays an important role in history. With the entrance of western medicine, the role of TCM was impacted. Most people prefer to western medicine but to TCM. The main cause is because TCM lacks of scientific basis. But many diseases, which outreach western medicine, are cured by TCM markedly. It attracts more and more people to study. The main task is how to discover the general rule of TCM according to large number of clinical data and history literature. Many modern methods have been introduced into TCM now. Statistical method is the most important one. But the result obtained through statistical method is not ideal. Neural network was introduced into the process of differential diagnosis only several years before and obtained better results[1]-[4]. But most works were limited in using neural network individually. All the symptoms are considered as the input component of the network during the process of differential diagnosis. In fact, not all the symptoms play the same role in the process. Some information is redundancy. It will take long training time if all symptoms are treated equally. Furthermore it will not produce a correct network model meeting with our requirement with small sample size. In this paper we develop a

* The research was supported by National Basic Research Program of China (973 program) under grant No. (2003CB517106).

multilevel neural network model to realize the diagnosis procedure of TCM. PCA (Principle Component Analysis) neural network is used to preprocess the symptoms, which will transfer symptom space into principle component space with lesser dimension. It can remove redundancy information and disturbed information efficiently. Backpropagation neural network is used to determine the syndromes with those principle component data obtained from PCA network. It can improve the efficiency of the network. The output neurons of the BP network are binary state "have" and "no". Thus we can encode them with 1 and –1 respectively. In TCM, prescription is given according to syndromes. The mean-field Boltzmann machine is adopted in the third network to realize it. At last, an example is given to illustrate the feasibility of the network model.

2 Diagnosis Procedure of TCM

There lies great difference between TCM and western medicine. TCM aims at finding the syndromes and western medicine aims at finding diseases. TCM's treatment focuses on the well-being of the entire person, not simply on the physical complaints and symptoms. Differentiation of syndromes is a method of understanding and diagnosing disease by the theories of traditional Chinese medicine. The diagnostic procedure involves an analysis of the clinical data regarding symptoms, physical signs, and disease history, together with information obtained from an application of the four diagnostic methods. Only correct differentiation can give a correct diagnosis and thus a suitable prescription. Therefore the four diagnostic methods are the basis of the differentiation of syndromes, and the differentiation of syndromes is the basis of prescription. Furthermore, the effect of prescription is the criteria of judgment for the correctness of the deduced differentiation. These three aspects form an organic connection-a basic law of diagnosis and treatment in traditional Chinese medicine. The diagnosis. process of TCM can be described as Fig. 1

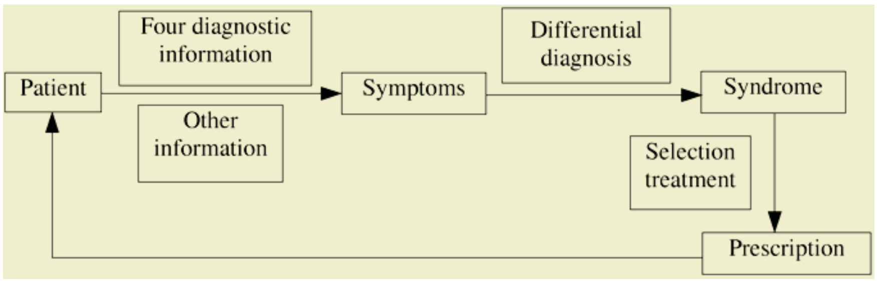


Fig. 1. Diagnostic procedure of TCM.

3 Neural Network Architecture

For realizing the diagnosis process with neural network, a multilevel neural network is developed. The neural network architecture is shown in Fig. 2. The first network is to generate principle component of symptoms, which is realized by PCA network. The second network is to generate syndrome according to the principle component of symptoms, which is realized through a backpropagation network. The last network is to give out prescription according to the syndromes, which is realized through a mean-field Boltzmann machine. All the realizations will be described at follows.

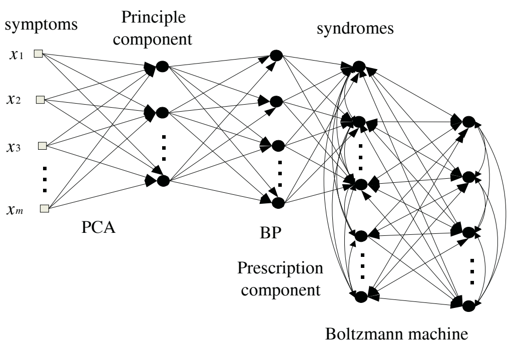


Fig. 2. Neural network architecture.

3.1 PCA Network

The first network will realize preprocess of symptoms. Many symptoms are not important for determining syndrome. Principle component analysis is a powerful transformation encoding method. In the PCA technique, the data are linearly transformed into the feature data of smaller dimension where the variance of the feature data is maximal. It is suitable to adopt PCA as the first network, which is described as follows[5]:

1) Initialize eigenvector (synaptic vectors) $w_{ij}(0)$ and learning parameter η_i.
2) The output vectors $y_j(n)$ of PCA linear neural network:

$$y_j(n) = \sum_{i=1}^{m} w_{ij}(n) x_i(n) \tag{1}$$

where m is the dimensionality of input vector x of PCA linear neural network.

3) Update the synaptic vectors using Sanger's generalized Hebbian algorithm:

$$\Delta w_{ij}(n) = \eta[y_j(n)x_i(n) - y_j(n)\sum_{k=1}^{j} w_{kl}(n) y_k(n)] \tag{2}$$

4) Iterate (2) and (3) until w_{ij} become stable.

The result of output $y_i, i = 1,2,\cdots,k$ is the principle component. The synaptic weight of neuron j converges to the i th component of the eigenvector associated with the j th eigenvalue of the correlation matrix of the input vector $\mathbf{x}(n)$.

3.2 Backpropagation Algorithm

The second network is to generate syndrome according to the principle components generated by PCA network. Three layers backpropagation neural network is adopted.

Backpropagation neural network is widely applied in many fields. It is based on gradient descent method which minimizes the sum of the squared errors between the actual and the desired output values. The concrete steps are described as follows[6].

1) Initialization: Pick the synaptic weights and thresholds from a uniform distribution randomly. Present the network with an set of training examples. For each example, perform the sequence of forward and backward computations.
2) Forward computation: Let the input vector $x(n)$ and bias vector $b(n)$. The local field $v_j^{(l)}(n)$ for neuron j in layer l is $v_j^{(l)}(n)=\sum_{i=0}^{m_0} w_{ij}^{(l)}(n)y_i^{(l-1)}(n)$. The output of neuron j in layer l is

$$y_j^{(l)} = \varphi_j(v_j^{(l)}(n)) \tag{3}$$

where $\varphi_j(.)$ is activation function. The output of the network $o_j(n)=y_j^L$. The error signal is

$$e_j(n)=d_j(n)-o_j(n) \tag{4}$$

where $d_j(n)$ is the j th element of the desired response vector $\mathbf{d}(n)$.

3) Backward computation. Compute the δs of the network, defined by

$$\delta_j^{(l)} = \begin{cases} e_j^{(L)}(n)\varphi_j^{'}(v_j^{(L)}(n)) & \textit{for neuron j in output layer L} \\ \varphi_j^{'}(v_j^{(l)}(n))\sum_k \delta_j^{(l+1)}(n)w_{kj}^{(l+1)}(n) & \textit{for neuron j in hidden layer l} \end{cases} \tag{5}$$

where the prime in $\varphi_j^{'}(.)$ denotes differentiation with respect to the argument. Adjust the synaptic weights of the network in layer l according to the generalized delta rule:

$$w_{ji}^{(l)}(n+1)=w_{ji}^{(l)}(n)+\alpha[w_{ji}^{(l)}(n-1)]+\eta\delta_j^{(l)}(n)y_i^{(l-1)}(n) \tag{6}$$

where η is the learning-rate parameter and α is the momentum constant.

4) Iteration. Repeat 3 and 4 until the stopping criterion is met.

The output of the network is syndrome vector $s_j, j=1,2,\cdots,m$. Each syndrome has binary state. The state "have" is encoded as 1 and "no" is encoded as -1.

3.3 Mean-Field Boltzmann Machine

The third network is to give out prescription according to the syndromes. Because the prescription's components have two states, the Boltzmann machine is the best choice to realize it. Boltzmann machine can obtain global minimum solution in theory. The shortcoming is that it needs large quantity computation. Some good methods have been introduced into Boltzmann machine to improve its speed. The mean-field Boltzmann machine is the fastest one currently. So mean-field Boltzmann machine is chosen here. The realize process is described as follows[7].

1) Set T to any positive big number. Initialize each free unit's mean-field $\langle x_T(k)\rangle$ and connection weight w_{kl} randomly, where $w_{ii} = 0$.
2) Iteration
   (a) Take any unit and calculate mean-field value

$$\langle x_T(k)\rangle = \tanh(u_k / T) \qquad x_T(k) \in \{1,-1\} \tag{7}$$

   where $u_k = \sum_l w_{kl} x_T(l)$, $w_{ii} = 0$. Repeate it until Eq.(7) change little.

   (b) Decrease T and repeat (a) until reach T_{final}.
3) Operate plus phase and minus phase with above process respectively.
4) Weight adjustment

$$\Delta w_{ji} = c(\langle x_i x_j\rangle^+ - \langle x_i x_j\rangle^-) \tag{8}$$

   where $\langle x_i\rangle^+$ and $\langle x_i\rangle^-$ are plus phase and minus phase respectively. Plus phase means input units and output units are clamped while the hidden units are free-running. Minus phase means only input units are clamped while output units and hidden units are free-running.

4 Example

There are 300 stroke patients' case histories. The clinical data were obtained from China Academy of Traditional Chinese Medicine. There are six different syndromes among these patients: wind syndrome, pathogenic fire syndrome, phlegm syndrome, syndrome of blood stasis, syndrome of deficiency of qi, hyperactivity of yang due to yin deficiency syndrome. Some patients take on two syndromes or more. 200 samples are chosen as training samples and the left are used to test.

In the sample, symptoms are scored with positive number according to the serious level. Each sample has 75 symptoms, such as pulse condition, palpitation, tinnitus, dizziness, stuttering, hemi-anesthesia, vomiting, twitch and so on. The number of principle component number is set 20. Different syndrome corresponds to different prescription components. There are total 30 components, such as red sage root, radix notoginseng, gastrodia tuber, scutellaria root and so on. Each network's parameters are set as follows.

In PCA network, input layer has 75 units and output layer has 20 units. The truncate error is set as 0.001.

In the backpropagation network, we choose sigmoid activation function in the second layer and choose linear activation function in the third layer. The input layer is the output layer of PCA. We select hidden unit number 12, learning rate 0.02, error 0.001, echo number 100000.

In the Boltzmann machine, hidden unit number is set 10. Input layer is the output layer of backpropagation network. Output has 30 neurons denote different prescription component. Initial temperature is set 20 and adjustment formulation is $T = T / \log(r+1)$ where r is the iteration number.

After training, we obtain acceptable simulate result. More than 75 percent samples can obtain correct results. The results have great improvement than the results obtained with individual neural network.

5 Conclusions

Neural network is used widely in many fields and many difficult and complex problems have been solved successfully. It is introduced into TCM field only recent years and is not applied widely. In fact, many TCM problems can be resolved through neural network because of its advantage. It is likely to discover the general rule of TCM with neural network. A multilevel neural network architecture is developed to realize the diagnostic procedure of TCM. The method is illustrated feasibility in applications. In the model, Boltzmann machine is only used as a memory. In fact, there lies complicate relation between syndrome and prescription. Fuzzy set theory should be introduced into the model to deal with the fuzziness of TCM. These are our further study.

References

1. Qin, B., He, Y. M., et al.: Neural Network Model of TCM Syndromes Based on MFBP Learning Algorithm. Chinese J. Basic Med TCM, **7** (2001) 66–69
2. Hu, J. N., Yan, S. C., et al.: An Intelligent Traditional Chinese Medicine Pulse Analysis System Model Based on Artificial Neural Network. Journal of China Medical University, **26** (1997) 134–137
3. Fan X. P., Peng, Z., et al.: Study on Depression Classified System Based on Multilayer Perceptrons Artificial Neural Network. Computer Engineering and Application, **40** (2004) 205–208
4. Pang, B., Zhang, D.: Computerized Tongue Diagnosis Based on Bayesian Networks. IEEE Transactions on Biomedical Engineering, **51** (2004) 1803–1810
5. Clifford C., Harry W.: Color Image Compression Using PCA and Backpropagation Learning. Pattern Recognition, **33** (2000) 1555–1560
6. Simon H.: Neural Networks: a Comprehensive Foundation (M). New York: Prentice hall (2001)
7. Xi G. C.: Neural Network Based on Mean-Field Theory Approximation. ACTA Electronic SINICA, **23** (1995) 62–64

An Automated Blowing Control System Using the Hybrid Concept of Case Based Reasoning and Neural Networks in Steel Industry

Jonghan Kim[1], Eoksu Sim[2], and Sungwon Jung[2]

[1] Production Management Research Team, RIST, San 32, Hyoja-Dong, Nam-Gu, Pohang City, Kyungbuk 790-330, Korea
kjh@rist.re.kr

[2] Department of Industrial Engineering, Seoul National University, San 56-1, Shillim-Dong, Kwanak-Gu, Seoul 151-742, Korea
{ses,jsw25}@ultra.snu.ac.kr

Abstract. Temperature adjustment is one of the critical tasks affecting the quality of manufactured steel. This is controlled by the Basic Oxygen Furnace's (BOF) blowing procedures. As many factors influence variations in temperature, it is often difficult to predict the blowing quantity necessary to achieve a required temperature. In this study, we assume the framework used by the intelligent blowing control system uses the Case Based Reasoning (CBR) and Neural Network (NN) to predict the appropriate blowing quantity in the BOF. Our proposed framework consists of three steps. First, we retrieve the similar cases for a new order requirement using CBR. Next, we train the NN engine with the selected case set. Finally, we predict the appropriate blowing quantity using a trained neural network. Experimental results show that the proposed framework performs more effectively than the framework without using CBR process.

1 Introduction

Numerous researchers have investigated ways promoting technological advancements through all processes within the steel industry with many researchers and practitioners focusing on large investment and interest to reduce costs, develop high value-added products, to increase productivity and to meet small lot size orders. Thus attempts to optimize operations using a variety of perspectives have been made [1]. Recently, some steel manufacturing processes have accomplished efficient productions due to the automated process control [2]. However, some important processes (e.g. oxygen blowing process) still rely on the experts' knowledge and not on an automated system. It results from the existent of some indefinable factors and relations within those processes. So many expert systems, using artificial intelligence, have been suggested [3],[4],[5]. But those systems were unable to achieve desirable results as they were incapable of filtering the noise. In this study, we suggest a refined expert system framework to monitor and control the oxygen blowing process in steel manufacturing. One of the most distinctive features of the proposed framework is the preprocessing procedure using a CBR engine. The prediction based on the preprocessed data set improves the predictability significantly.

The rest of the paper is organized as follows: Section 2 describes the problems' background. The proposed system framework is described in Section 3. Section 4 reports on the computational experiments. Finally, Section 5 provides concluding remarks and suggests a future research direction.

2 Problem Background

The production of steel follows the sequence described in Fig 1; as iron making, steel-making, continuous casting and rolling process. In the iron-making process, iron ore and coke are fed into a blast furnace and transformed into a molten iron. In the steel-making procedure, the hot metal is charged into a basic oxygen furnace and transformed into pure liquid steel ready to be formed into semi-finished products. Through the continuous casting and rolling process, this pure steel is turned into solid steel and, finally, transformed into a variety of shapes as steel plates, pipes, etc.

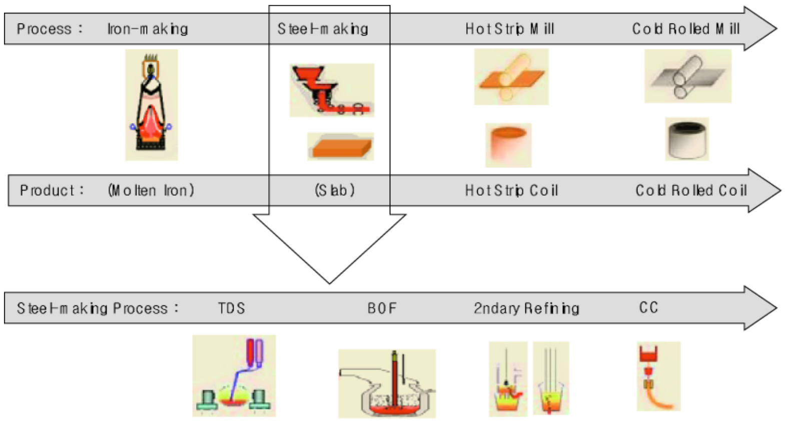


Fig. 1. Production process in steel industry.

This paper focuses on the oxygen blowing procedures in steel-making process. As mentioned above, the hot metal tabbed from the blast furnace in the iron-making process is carried over to the steel-making process. In the steel-making process, this hot metal is charged into a BOF (basic oxygen furnace). A high-pressure stream of pure oxygen is then injected into the hot metal, transforming impurities into gases and slag. The purpose of this oxygen blowing procedure is to reduce impurities, such as carbon, phosphorus and manganese, in molten iron in order to satisfy the required steel quality. The quantity of the oxygen blown in the BOF may vary with the required steel properties, present composition of chemical elements in molten iron and operational conditions such as the present furnace temperature. Hence, in practice, the prediction of the appropriate oxygen blowing quantity has relied on the experts' experience. However, there is no framework to predict the oxygen blowing quantity without involving human resource necessary for the automated control of the BOF process.

3 Framework for the Intelligent Blowing Control System in BOF

In this study, we propose a framework based on the CBR and NN. The overall framework for the intelligent blowing control system is illustrated in Fig. 2. When a molten iron arrives at the steel-making process, the data and operational conditions are gathered by the monitoring system; data such as the composition of chemical elements in the molten iron. Based on the information from the monitoring system and the required steel properties, a CBR engine retracts similar cases from a database containing the previous blowing quantity for each case. Using this selected previous case set, the neural network engine is trained to predict the blowing quantity required for that specific case. The blowing quantity for the case is determined by the trained neural network engine, and this decision is subsequently inserted into the database.

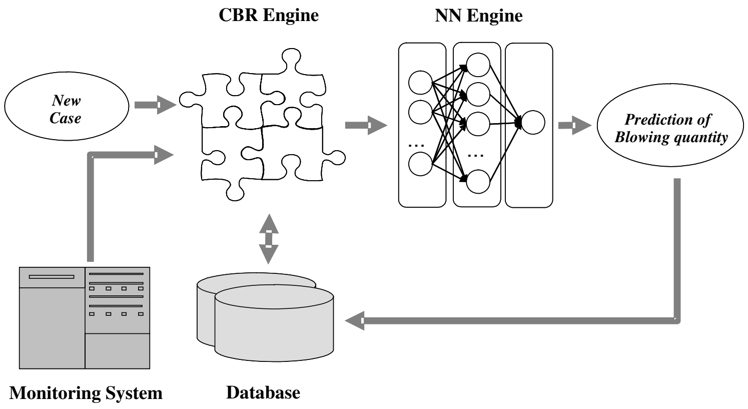


Fig. 2. The framework using CBR and NN.

3.1 Critical Factors for Prediction of Oxygen Blowing Quantity

Based on metallurgical knowledge and experts' experiences, all factors affecting the decision of the oxygen blowing quantity in the BOF process are investigated. The factors can be categorized into four groups related to refining, casting, ladle, and species of steel. Only 12 out of 38 influencing factors are deemed significant, based on the correlation coefficient between each factor and temperature drop. In addition, each factor's function type is determined using a scatter plot between the factors and oxygen blowing quantity. For example, a cubic expression of the arriving temperature is expected to predict the oxygen blowing quantity.

3.2 Selection of the Similar Case Set Using CBR

When molten iron for a new order arrives at the BOF, the CBR engine selects the most similar cases considering the order requirement and the current operational environment. Our CBR engine uses the nearest neighbor search method for retrieval. It can be expressed as equation (1) where P and C represent the selected critical factor for current and previous orders respectively.

$$Sim(P,C)=\sqrt{\sum_{i=1}^{n} w_i \times sim(p_i,C_i)}$$

3.3 Training the Neural Network Engine

Neural networks are composed of simple parallel elements inspired by the nervous system. Similar to nature, the network function is determined largely by the connections between elements. Usually, neural networks are adjusted or trained such that a particular input leads to a specific target output. Based on a comparison of the output and the target, the network is adjusted until the network output matches the target.

To predict the oxygen blowing quantity, the development of the neural network model is attempted based on the Matlab environment. As the most popular network architecture in use today, multilayer perceptrons are adopted to predict the temperature drop during the continuous casting procedure. To compare the analysis methods, the neural network model is developed with the detailed architectural structure as follows:

- ♦ The input node is made up of 12 factors.
- ♦ The number of hidden layers and nodes in each layer is set to 2 and 16, respectively. The complexity of the network is also selected considering the danger of over-fitting.
- ♦ The log sigmoid function is adopted as the transfer function at each node.
- ♦ The input layer is connected to the output layer through hidden layers.
- ♦ For the training algorithm, the back propagation algorithm is selected. In back propagation, the gradient vector of the error surface is calculated. This vector points along the line of the steepest descent from the current point. Moving along it for a short distance will cause the error to decrease. A sequence of such moves makes the actual response of the network move closer to the desired response.
- ♦ The neural network model based on the Matlab environment is developed.
- ♦ Back propagation is performed based on the gradient descent method.

4 Computational Experiments

To evaluate the proposed framework for the prediction of the oxygen blowing quantity, we perform the simulation experiment using the real industry data. First, we gather over 5000 data. Among them, we randomly select 100 data and regard this as future order data. Subsequently, two prediction methods are used to predict the oxygen blowing quantity for the case. One is our proposed method and the other is the prediction by the neural network without a preprocessing procedure using CBR. We classify the structure of the neural network for the experiment as simple, medium and complex. Simple size of the neural network consists of one hidden layer with ten hidden nodes. Medium size of the neural network is composed of two hidden layers with ten hidden nodes in each layer. Large size of the neural network is made up of the three hidden layers with ten hidden nodes in each layer. The performance measure is the difference between the predicted oxygen blowing quantity and the actual blowing oxygen quantity determined by the experts. If the absolute value of the If the absolute value of the difference between those the predicted and actual values is small, we can say that the estimator shows a good performance.

One of the simulation results for the model which adapt the complex structure of neural network without preprocessing process is shown in Fig. 3.

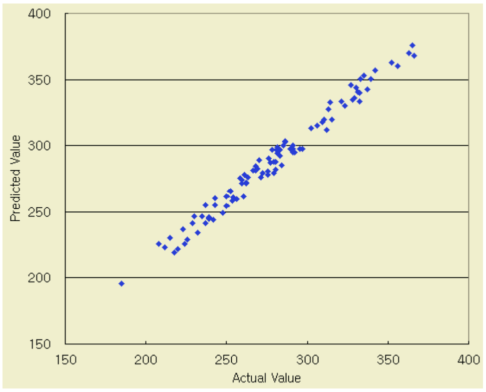


Fig. 3. The scatter plot for the predicted and actual value.

Table 1 shows the result of the average difference between the prediction value and the actual value for all six models. In all of the cases the adaptation of the preprocessing procedure using the CBR improved the prediction quality by approximately 10~25%.

Table 1. The result of the simulation experiment.

Structure of NN Preprocessing	Complex	Medium	Simple
No	25.6	25.2	22.1
Yes	19.3	20.6	19.7
% of the improvement	25%	18%	10%

We also mention that the case which uses the preprocessing procedure shows the stable performance regardless with the structure of the neural network, while the performance of the case without using the preprocessing procedure worse with the complexity of the neural network structure.

5 Conclusions

In this study, we propose an artificial prediction system for the automation of the BOF process. In practice, the prediction of the oxygen blowing quantity in the BOF process is highly dependant upon the experts' skill and experience. Therefore, the development of an intelligent prediction system for the oxygen blowing quantity in the BOF is necessary for the automated BOF control system. One of the most distinctive features of the proposed framework is the preprocessing procedures using the CBR engine. The prediction based on the preprocessed data set improves the predictability significantly. The simulation experiment shows that the adaptation of the preprocessing procedures using CBR significantly improves the prediction accuracy.

References

1. Yoo, B.D.: Introduction to the Steel Making. Inha University Press (2004)
2. Saxen, H., Sillanpaa, M.: A Model for Decision Support in Continuous Steel Casting. Modeling and Simulation in Materials Science and Engineering, **2** (1994) 79-98

3. Millman, M.S., Thornton, G.: New Technologies in Steelmaking. Rev Metall-Paris, **95** (1998) 477-486
4. Hatanaka, T., Takenaka, M., Oka, Y.: Development of Blowing Control System in BOF Applying AI Technology (1991) 48-53
5. Yamane A., Yamane, H., Miyahara, K., Iwamura, T.: Computer Integrated Steelmaking at the No.2 BOF Shop of the Mizushima Works. Proceedings of the Sixth International Iron and Steel Congress (1990) 47-54

Neural Networks Based Multiplex Forecasting System of the End-Point of Copper Blow Period

Lihua Xue[1,4], Hongzhong Huang[1,2], Yaohua Hu[3], and Zhangming Shi[4]

[1] Key Lab. for Precision and Non-traditional Machining Technol. of Ministry of Education, Dalian University of Technology, Dalian, Liaoning 116023, China
xuelih@hotmail.com

[2] School of Mechatronics Engn, University of Electronic Science and Technology of China, Chengdu, Sichuan 610054, China
hzhhuang@dlut.edu.cn

[3] College of Information Engineering, Dalian Maritime University, Dalian, Liaoning 116023, China

[4] School of Energy and Power Engineering, Central South University, Changsha, Hunan 410083, China

Abstract. The neural network and the experiential evaluation method are introduced into the industrial converting process forecast, and a multiplex forecast system is proposed at the end-point of copper blow period in a matte converting process. The fuzzy clustering analysis method is used to identify the number of hidden nodes. Results show that the forecast is highly dependable and capable of providing effective guidance to the production process.

1 Introduction

Converting is an important process in copper pyro-metallurgy. The product of matte converting process is copper. The aims and keys for the process are to guarantee the quality of the copper and prevent over-blowing as well as under-converting. At present, the end-point judgment of matte converting is mainly dependent on experience. Matte converting is a complicated process, which is related not only to chemical reactions, but also to fluid flow and to heat and mass transfer. Since the materials fed into the furnace differ greatly and the process is influenced by various factors, it is difficult to control the process online. In this work, based on studying the industrial converting process, a multiplex forecasting system is proposed to provide judgment of the end-point of copper blow period in matte converting process.

2 Matte Converting Process

Matte converting is commonly finished in horizontal P-S converters. In the converting process, the air is blasted into the matte to oxidize and remove the iron, sulfur and other harmful impurities to get blister copper. At the same time, the precious metals can be gathered into blister copper. The thermodynamic property difference of oxidizing the FeS and Cu_2S makes FeS oxidize firstly into FeO, which will react with the inputting silica flux to generate slag. When there is no FeS in the matte, Cu_2S will begin to be oxidized into Cu_2O, and Cu_2O will react with the unoxidized Cu_2S to generate copper [1]. So the matte converting is composed of slag blow and copper blow processes. Main chemical reactions in matte converting process are as follows:

Reactions in slag blow period are as follows.

$$2FeS+3O_2=2FeO+2SO_2$$

$$2FeO+SiO_2=2FeOSiO_2$$

Reactions in copper blow period are as follows.

$$2Cu_2S+3O_2=2Cu_2O+2SO_2$$

$$Cu_2S+2Cu_2O=6Cu+SO_2$$

These reactions happen partly at the end of the slag blow period. The converting in the copper blow period will continue until all Cu_2S is converted into copper.

3 Multiplex Forecasting Model Based on Neural Networks

3.1 Feedforward Neural Network

Neural networks (NNs) are a new information processing technique. In general, a neural network is typically composed of interconnected units organized in layers which serve as model neurons [2]. Neural networks are mathematical models of theorized mind and brain activity, which provide a great degree of robustness or fault tolerance due to the massive parallelism in their design. Neural networks are used in the situations where only a few decisions are required from massive amount of data, or a complex nonlinear mapping needs to be learned [3].

A multi-layer feedforward neural network is an effective tool for modeling nonlinear relationships between input and output data [4]. The structure of feedforward neural networks consists of an input layer, not less than one hidden layer and an output layer. The nodes at two adjacent layers are connected fully, but no connection exists between nodes in the same layer.

The BP algorithm is an iterative gradient algorithm designed to minimize the errors between the actual output of a neural network and the desired output [5]. In this work, the following Sigmoid function is selected as the activation function in the supervised learning process:

$$f(x)=\frac{1}{1+e^{-x}} \quad . \tag{1}$$

A complete description of BP algorithm can be seen in the reference [6].

In this work, a typical three-layer feedforward neural network is employed (See Fig. 1). This neural network consists of an input layer with l input nodes, a hidden layer with m nodes and an output layer with n output nodes.

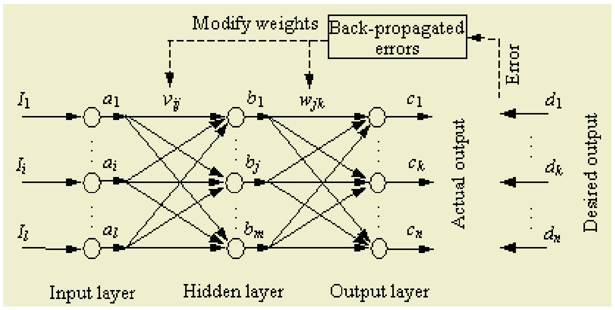


Fig. 1. Topological structure of a three-layer feedforward neural network.

3.2 Fuzzy Clustering Analysis of Identifying the Number of Hidden Nodes

A main difficulty in the application of neural networks is how to rationally identify the number of hidden nodes. In this work, the fuzzy clustering analysis is used to identify the number of hidden nodes in the feedforward neural network. The clustering process is as follows:

(1) Construction of new sample sets
Let the input sample be

$$\begin{aligned} &X=[X_1,X_2,\cdots,X_m]^{\mathrm{T}} \\ &X_j=[x_{j1},x_{j2},\cdots,x_{jn}] \quad j=1,2,\cdots,m \end{aligned} . \tag{2}$$

and the output sample be

$$\begin{aligned} &Y=[Y_1,Y_2,\cdots,Y_m]^{\mathrm{T}} \\ &Y_j=[y_{j1},y_{j2},\cdots,y_{jq}] \quad j=1,2,\cdots,m \end{aligned} . \tag{3}$$

where x_{jk} is the kth input variable of the jth sample, y_{jk} is the kth output variable of the jth sample, m is the number of samples.

Connecting the sample inputs and outputs to construct new samples

$$\begin{aligned} &Z=[Z_1,Z_2,\cdots,Z_m]^{\mathrm{T}} \\ &Z_j=[z_{j1},z_{j2},\cdots,z_{jn},z_{j(n+1)},z_{j(n+2)},\cdots,z_{j(n+q)}] \; . \\ &j=1,\ 2,\ \cdots,m \end{aligned} \tag{4}$$

where Z_i is the ith sample, z_{ij} is the jth input variable of ith new sample.

(2) Similarity calculation
The angle-off cosine method is used to calibrate the similarity coefficient [7].

$$r_{ij}=\frac{\left|\sum_{k=1}^{n+q} z_{ik}z_{jk}\right|}{\sqrt{\sum_{k=1}^{n+q} z_{ik}^2 \sum_{k=1}^{n+q} z_{jk}^2}} \qquad r_{ij}\in[0,1] \; . \tag{5}$$

(3) Construction of fuzzy similar matrix

$$\tilde{R}=(r_{ij})_{m\times m} \; . \tag{6}$$

(4) Clustering analysis
The direct clustering method is used [7]. Let the threshold $\lambda\in[0,1]$ (λ=0.989~0.991), if $r_{ij}\geq\lambda$, Z_i and Z_j belong to the same cluster. The number of all the clusters is the number of hidden nodes in the feedforward neural network.

3.3 Multiplex Forecasting Model Based on Neural Networks

To assure the accuracy of the forecast, a multiplex system is set up to provide supplement information on judgment of the end-point of copper blow period in the converting process, in which the forecast is made by combining two different models of the neural network and the experiential evaluation, and the result is verified with the value of the oxygen utilization ratio calculated in material balance computation.

According to different targets of the two neural networks, different structures of neural networks are used to construct the forecasting models.

Input Layer. Main factors that influence the end-point of copper blow period consists of three classes: dynamic, thermodynamic and operating factors. Their influence mainly embody in the production situation of flash furnaces, the input and output of material, the ingredient differences of cold dopes, the fluctuation of production situations in converters and the operating habit, etc. The influential factors of two neural network models are as follows respectively.

Model I: The input variables are the mass of matte, the grade of matte, the volume of needed oxygen in slag blow period, the mass of cold dopes in slag blow period, the mass of slag, the copper loss in slag, the iron/silica ratio in slag, the high copper cold dopes in slag blow period, the low copper cold dopes in slag blow period. So the number of input nodes is 9.

Model II: Apart from the 9 input variables in Model I, another variable, the average volume of air in copper blow period, is added in Model II. So the number of input nodes is 10.

Output Layer. In this work, the numbers of output nodes in two neural networks are all 1.

Model I: the volume of oxygen in copper blow period

Model II: the converting time of copper blow period

Hidden Layer. The above-mentioned fuzzy clustering analysis is used to identify the number of hidden nodes. To every converter of the matte converting system in a smelting plant, the recommended numbers of each model can be seen in Table 1.

Table 1. Numbers of hidden units of every neural network model in a smelting plant.

	Converter 1	Converter 2	Converter 3	Converter 4	Converter 5	Converter 6
Model I	25	30	35	33	27	28
Model II	15	40	24	27	31	26

The results with better prediction from the two models will be the output results. When the two results are not in the valid range of the end-point, the converting end-point will be evaluated through the experiential formula,

$$T_b^* = 1.25 \times \left(W + \alpha \times W_1^{\mathrm{S}}\right) \times \left(540 - g_m\right) / q_b \times \left(\frac{21}{\eta_b}\right) + \varepsilon \ . \tag{7}$$

and the evaluated value will be the final results.

Let T be the needed time of duration in copper blow period and [T-7, T+3] be the range of end-point.

4 Actual Effect of Multiplex Forecasting System

Industrial tests of multiplex forecasting system were made on Converter 1, 3 and 4 in a smelting plant from December, 2002. The actual effect is shown in Fig. 4.

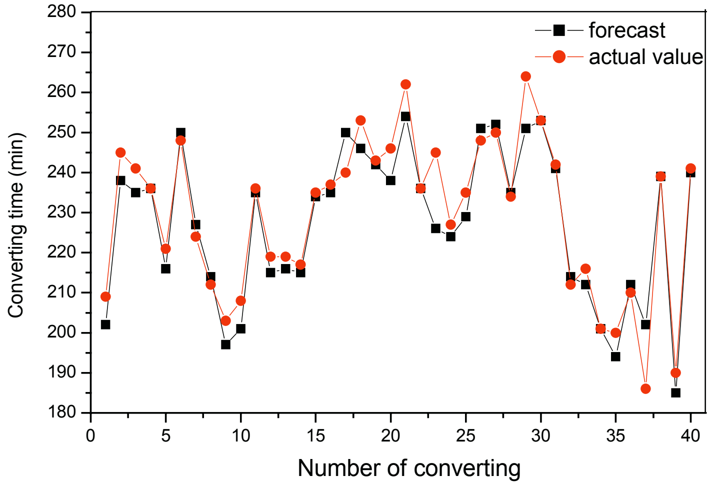


Fig. 2. Comparison between forecast and true value of end-point.

It is easy to see that the multiplex end-point forecasting system can predict the end-point of the copper blow period according to the actual production. the forecasting accuracy of the system has reached over 87.5%.

5 Conclusions

On the basis of studying the industrial converting process as well as through comprehensive application of the neural networks with BP algorithm and the experiential evaluation method, a multiplex forecast system is proposed at the end-point of copper blow period in a matte converting process. The fuzzy clustering analysis method is also used to determine the number of hidden nodes in the neural networks. The combination of the two technologies in different fields improves the accuracy and adaptability of the system, and gives an impulse to the improvement of the converting technology and managing level. The forecasting system can prevent over- and under-converting and decrease the production accident greatly. It is also of important academic significance and great practical value in guiding normal operation, ensuring high production quality and raising the whole level of operations of P-S converters.

Acknowledgement

This research was partially supported by the National Excellent Doctoral Dissertation Special Foundation of China under the contract number 200232.

References

1. A Handbook for Extractive Metallurgy of Nonferrous Metals Committee: A Handbook for Extractive Metallurgy of Nonferrous Metals: Copper-Nickel Volume. Metallurgical Industry Press, Beijing (2000)
2. de Oliveira, K.A., Vannucci, A., da Silva, E.C.: Using Artificial Neural Networks to Forecast Chaotic Time Series. Physica A. **284** (2000) 393-404
3. Bahrami, A., Lynch, M., Dagli, C.H.: Intelligent Design Retrieval and Packing System: Application of Neural Networks in Design and Manufacturing. International Journal of Production Research. **33** (1995) 405-426
4. Simpson, P.K.: Artificial Neural Systems: Foundations, Paradigms, Applications, and Implementations. Pergamon Press, Oxford (1989)
5. Sun, J., Kalenchuk, D.K., Xue, D., Gu, P.: Design Candidate Identification Using Neural Network-Based Fuzzy Reasoning. Robotics and Computer Integrated Manufacturing. **16** (2000) 383-396.
6. Liu, H.L., Bao, H.: Artificial Intelligence Optimization in Chemical and Metallurgical Process. Metallurgical Industry Press, Beijing (1999)
7. Han, L.Y., Wang, P. Z.: Applied Fuzzy Mathematics. Capital University of Economics and Business Press, Beijing (1998)

Modeling and Prediction of Electric Arc Furnace Based on Neural Network and Chaos Theory

Fenghua Wang, Zhijian Jin, and Zishu Zhu

Department of Electrical Engineering, Shanghai Jiaotong University,
Huashan Road 1954, Shanghai 200030, China
{fhwang7723,zhijianjin,zishuzhu}@sjtu.edu.cn

Abstract. Electric arc furnace is commonly used in iron and steel industry to produce quality steel by melting iron and steel scraps using electric arc. It represents one of the most disturbing loads in the subtransmission or transmission electric power systems. Therefore, it is necessary to build a practical model to descript the behavior of electric arc furnace in the simulation of power system. The electrical fluctuations in the electric arc furnace have proven to be chaotic in nature. This paper deals with the problem of electric arc furnace modeling using the combination of chaos theory and neural network. The radial basis function neural network is used to predict the arc voltage of arc furnace with one-step and multi-step ahead. The results can be applied to simulate the EAF load in power system and estimate the future state of arc furnace for control purpose.

1 Introduction

The electric arc furnace (EAF) is widely used in iron and steel industry. The basic operating principle of EAF is that an electric arc is struck from one or more electrode to a metallic charge. The thermal energy dissipating by the arc is used to melt and refine the iron and steel raw material to produce quality steel. It is energy efficient and economical.

However, the operation of EAF brings about severe power quality problems on power system, such as harmonics, inter-harmonics, voltage flicker, current and voltage imbalances etc. Therefore, it is necessary to build a practical model for the EAF employed in the simulation of whole plant. Due to the complex nature of EAF operation, the seemingly random arc behavior makes it difficult to formulate an explicit model. Up to now, such a circuit model to represent the EAF is still not available.

Historically, there have been two general approaches to the problem of EAF modeling. One is the time domain analysis method [1] and the other is frequency domain analysis method [2]. Stochastic ideas can also be found in the previous studies to model the behavior of an arc furnace due to the fact that frequency modulation of the supply voltage that is less than 0.5% can cause voltage flicker if the frequency of the modulating signal lies within the 6-to-10-Hz range [3]. In [4], Cassie equation and Mayr equation are applied to model the three-phase EAF. The effects of arc furnace transformer tap changer and flexible cables mutual inductances are considered.

Recently, the electrical fluctuations in the EAF have proven to be chaotic in nature [5]. Consequently, chaos theory has become a modeling issue for EAF. The Lorenz chaotic model and the combination of Lorenz model and Logistic model are used to represent the highly varying behavior of currents in an EAF [6], [7]. A low-frequency chaotic signal generated by the Chua chaotic circuit is modulated with the voltage obtained by using a dynamic, multi-valued voltage-current characteristic of electric arc to represent the output of the proposed EAF model [8]. Those above-mentioned methods adopt proper chaotic models to generate time series data, which are statistically similar to the EAF data and really obtain some satisfactory results. However, chaos theory is still a relatively new subject and has not been fully understood yet. Up to now, available chaos model are very few. It is sometimes difficult to fit a suitable model and parameter to the EAF data.

From an electrical network point of view, the EAF model can be represented either a variable load or a controlled voltage or current source. In this paper, a single-phase EAF is modeled as a controlled voltage source, using the output of Radial basis function neural network in a reconstructed phase space based on the chaotic phenomenon in EAF system. In addition, the arc voltage is regulated by the electrode control system and its variations are closely related to production efficiency and power quality problem. The results of multi-step ahead prediction can also be used to improve the operation of electrode control system.

2 Description of Electric Arc Furnace

The schematic diagram of EAF is shown in Fig.1. The voltage of Point of Common Coupling (PCC) is 110kV. The EAF is connected to the electric net by means of a high-voltage/medium-voltage (HV/MV) transformer (T1) and is fed by a MV/low-voltage (LV) transformer (T2). The furnace side of this transformer has adjustable voltage from 335 to 220V in order to vary the furnace power. The lead impedance represents the impedance of the connection between the secondary bus of the furnace transformer and the electrodes of the EAF.

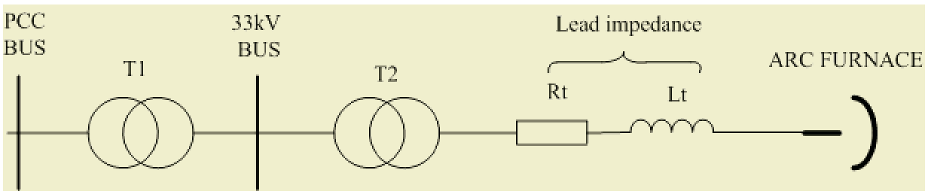


Fig. 1. One line diagram of electric arc furnace plant.

Fig.2 shows the typical EAF voltages and current waveforms. The arc current waveform is basically similar to the sine wave. The arc voltage is aperiodic and irregular with a trapezoidal shape. It is stochastically changing due to the metal-scrap adjustments, electromagnetic forces and arc-electrode variable displacement during the steel-making process. When investigated the chaotic behavior of some system, Lyapunov exponents are widely used as a standard diagnostic tool to estimate the amount of chaos in a signal. A distinctive feature of chaotic system is the high sensitivity to small changes in initial conditions. A positive Lyapunov exponent indicates a net average divergence from initial conditions. This in turn implies sensitive dependence on initial conditions, and therefore the presence of chaos. For the voltage signals shown in Fig.2, the largest Lyapunov exponent is evaluated and results in 2.54. The positive number is a further sign of the chaotic nature of the arc phenomenon.

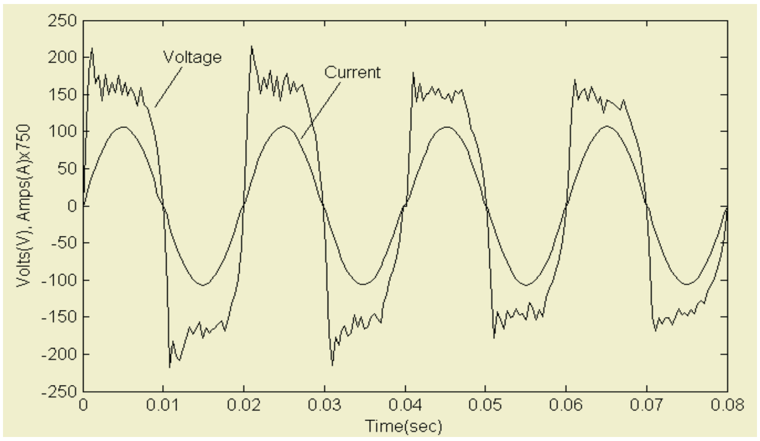


Fig. 2. Typical arc voltage and current waveforms.

3 Modeling and Prediction of Electric Arc Furnace

The basis of one-step and multi-step prediction of chaotic time series is the embedding theorem proposed by Takens. It deems that the evolution of an arbitrary component in the system depends on the other relative components. The information of these components is hidden in the developing process of an arbitrary component. When reconstruct a state space, a new system with the same dynamic property to the original system can be produced if delay time and embedding dimension of the phase space are selected reasonably.

Supposed that an observed time series is $x(i) = x(t_i)$ $t_i = t_0 + i\Delta t$ $i = 1,2,\cdots,N$, herein, Δt is the sampling time interval. The k vectors with m-dimension can be constructed as follows.

$$\begin{aligned} X(1) &= [x(1), x(1+\tau), \dots, x(1+(m-1)\tau)] \\ X(2) &= [x(2), x(2+\tau), \dots, x(2+(m-1)\tau)] \\ &\dots\dots \\ X(k) &= [x(k), x(k+\tau), \dots, x(k+(m-1)\tau)] \end{aligned} \tag{1}$$

where $N = k + (m-1)\tau$ is the length of time series, m is embedding dimension, τ is delay time. All the k vectors form a reconstructed phase space.

According to the embedding theorem, for a reconstructed phase space in (1), there exists a smooth mapping $f : R^m \to R$

$$y(n+p) = f[y(n), y((n-\tau)), \cdots, y(n-(m-1)\tau)] \tag{2}$$

where p is the prediction step. Consequently, the prediction problem is transformed to reconstruct the phase space (attractor) and solve the mapping from the attractor to the prediction value in the known time series. Embedding theorem does not specify the form of mapping f and the methods to evaluate the delay time τ and embedding dimension m, which are necessary to estimate in practical application. As we know, neural networks process high approximate capability for a smoothing continuous function after the appropriate training. Therefore, neural networks can be used to realize the approximate of foregoing mapping to obtain the prediction purpose.

3.1 Selection of Delay Time

Delay time τ is an important value in the process of phase space reconstruction. Almost every value for the delay time should work. However, it should not be too small because the attractor would be restricted to the diagonal of the reconstructed phase space and it should not be too large since the components would become uncorrelated. Although there are many suggestions to select the delay time, it is not sure which method is suitable for all the problems. At present, autocorrelation coefficient method is a common method to evaluate the delay time. It determines delay time according to the computation of autocorrelation coefficient of the time series. Generally, the delay time τ is determined as delay time when the autocorrelation coefficient attenuates to a rather small value. However, this value should not be too small which corresponds to very long delay time. Reference [9] presents that a reasonably empirical value is $1/e$, approximately 0.4.

Fig. 3 shows the result of delay time of one of the measured arc voltage waveform. When delay time is equal to 12, autocorrelation coefficient attenuates to 0.4 nearly. Therefore, $\tau = 12$ is selected as delay time of the reconstructed phase space.

3.2 Selection of Embedding Dimension

For an n-dimensional dynamical system, the embedding theorem indicates that the sufficient condition on the embedding dimension m is given by

$$m \geq 2d + 1 \tag{3}$$

where d is the correlation dimension of attractor. The selection of m is also important. If m is too small, attractor can not be spread out completely. On the contrary, if m is too large, a lot of unnecessary calculations of invariants, such as Lyapunov exponent will be added. G-P arithmetic is a common method to calculate the embedding dimension and correlation dimension. The main steps are illustrated in [9] and we don't explain here.

Fig.4 shows the curve of $\ln C(r,m)$ and $\ln r$ when the embedding dimension increases from 1 to 11. In the figure, when embedding dimension is 8, the slopes of the linear parts of the curves do not change any more. Therefore, the embedding dimension is approximately 8. Then the correlation dimension is 2.6 ± 0.3.

Fig.4 shows the curve of $\ln C(r,m)$ and $\ln r$ when the embedding dimension increases from 1 to 11. In the figure, when embedding dimension is 8, the slopes of the linear parts of the curves do not change any more. Therefore, the embedding dimension is approximately 8. Then the correlation dimension is 2.6 ± 0.3.

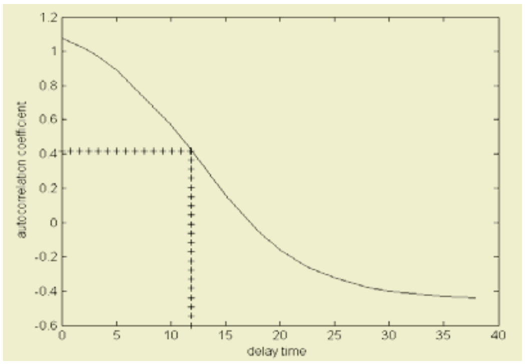


Fig. 3. Delay time determination.

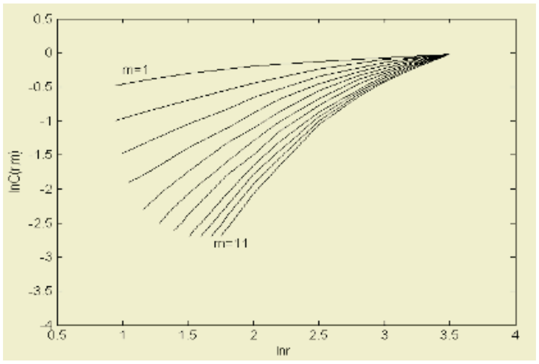


Fig. 4. Relation between ln $C(r,m)$ and ln r.

3.3 Structure of Neural Network

In recent years, artificial neural networks have been widely used to model the system for which mathematical models involve complex, multi-variable process with time-variant parameters. As a nonlinear time-varying load in power system, EAF belongs to both of those above categories, which makes it possible to learn the dynamic behavior of EAF by virtue of neural networks. Among the several kinds of neural networks, radial basis function (RBF) network has fast learning ability and high approximate and classification capability. Therefore, RBF network is used to model and predict the arc voltage in the paper.

The learning problem of the RBF network is to select the appropriate function centers and widths and then solve for the output weights using available techniques. In the paper, the method proposed by Chen et al [10] is used which is of the type where the centers and widths of Gaussian functions are first selected and then the network is solved to find the output layer weights. The orthogonal least squares algorithm is implemented which provides a systematic approach to the selection of RBF centers. It would have been selected randomly when using forward regression procedure. Matlab Functions from the Neural Networks Toolbox are used to carry out the above-mentioned process [11].

4 Results and Discussion

Several groups of arc voltage data in the whole steel-making process have been measured in Shanghai Baosteel Group Corporation. The sampling time interval is 2.0×10^{-5} second. Here we just randomly select one of them to show our results. First the erroneous data are removed and then the measured arc voltage data in 0.2 second in number 1000 points are used as experiment points. The first 800 points are used to construct initial phase space and the rest 200 points are used as predicted points to confirm the result of one step prediction.

According to the equation (2), the problem of prediction corresponds to the mapping from the reconstructed high dimension space to one dimension space. Consequently, the number of input nodes of RBF network should equal to the embedding dimension. The number of hidden nodes is varied to obtain the minimum number of neurons that will yield an adequate approximation in describing the dynamic behavior of arc voltage and we select 12 as the number of hidden nodes. Consequently, a RBF neural network with $8\times12\times1$ topology is applied to yield a one step prediction for the arc voltage. The mean square error (MSE) is used as performance indices to measure the accuracy of this method quantitatively.

$$MSE = \frac{\sum_{i=1}^{K}(\hat{f}(i) - f(i))^2}{K} \tag{4}$$

where $\hat{f}(i)$ is the predicted value, $f(i)$ is the actual value, K is the number of data.

Fig.5 is a plot of predicted output and system output and the relative error curve between the two signals. As can be seen, the signals are virtually identical and the network appears to have generalized well for the case. The MSE is 5.12%. The network does a good job of predicting the arc voltage except at small number of points.

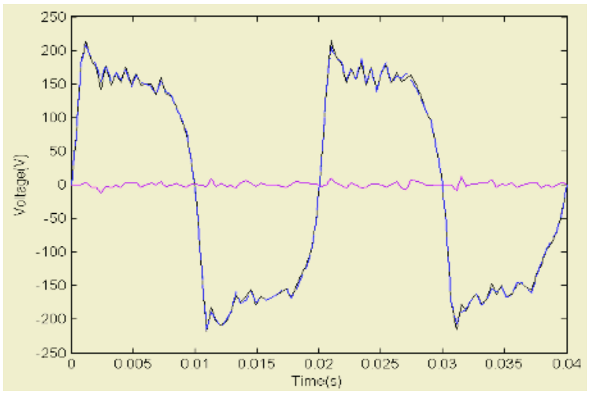


Fig. 5. One step prediction result using RBF network and phase space reconstruction theory. The black line is the measured value. The dotted blue line is the predicted value. The purple line is the difference between the two signals.

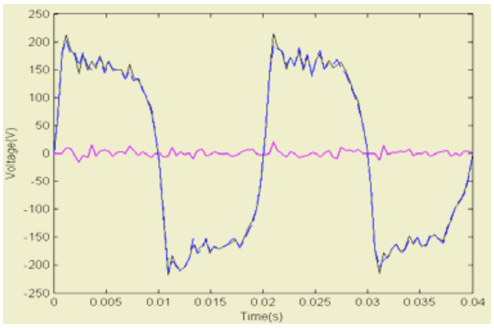


Fig. 6. One step prediction result using RBF network alone. The black line is the measured value. The dotted blue line is the predicted value. The purple line is the difference between the two signals.

In order to illustrate the effectiveness of the proposed method, RBF network alone is also used to predict the arc voltage using the normalized voltage signals. The inputs to the network consist of the arc voltages at times t-1, t-2, t-3, t-4 and t-5. These values are propagated to a 10 node hidden layer. The result for one step prediction for the arc voltage and the relative error curve is shown in Fig.6. The MSE is 6.54%. It is easily to seen that the accuracy of prediction results using RBF network alone is low compared with the results using phase space reconstruction and RBF networks.

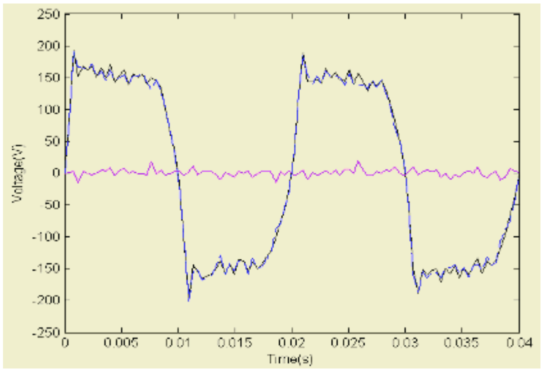


Fig. 7. Five times prediction result using RBF network and phase space reconstruction theory. The black line is the measured value. The dotted blue line is the predicted value. The purple line is the difference between the two signals.

In practical applications for the purpose of controlling the electrode, the next step ahead prediction may not be sufficient. Therefore, the case for multi-step ahead prediction is also calculated using the proposed method in the paper. Fig.7 shows the prediction results for five times ahead steps for a similar type of waveform using phase space reconstruction and RBF networks. The MSE is 6.67%. The results still appear to perform relatively well except at small number of points. We also do the prediction for more than five times ahead steps. However, the result does not match well with the measured arc voltage waveform from the sixteen times ahead steps on. We do not show the result here.

5 Conclusions and Future Works

When viewed the electric arc furnace as a controlled voltage source in electrical network, the study of modeling work of EAF is focused on the characteristic of arc voltage. Based on the chaotic property of arc voltage time series, this paper introduces RBF network and phase space reconstruction theory to predict the arc voltage signal. It can be applied to model the EAF load and optimize the production process by adjusting the arc voltage in advance. The result of one-step ahead prediction and multi-step ahead prediction does match well with the measured arc voltage signal, which confirms the validity of the proposed method.

When electric arc furnace is simulated in power system simulation software, such as EMTP, there are still several problems needed to solve. One of the problem is the synchronize issue between electric arc furnace model and the rest of the electrical network. Implementation of the electric arc furnace model in EMTP is part of on-going work. We will also do some research for controlling the arc furnace state based on the prediction results in order to increase the heat transfer efficiency and reduce the adverse effects of DC arc furnace to the power system.

References

1. Varadan, S., Elham B. Makram, E.B., Girgis, A.A.: A New Time Domain Voltage Source Model for An Arc Furnace Using EMTP. IEEE Trans. Power Delivery, **11** (1996) 1685-1691
2. Alonso M.A.P., Donsion, M.P.: An Improved Time Domain Arc Furnace Model for Harmonic Analysis. IEEE Trans. Power Delivery, **19** (2004) 367-373
3. Montanari, G.C., Loggini, M., Cavallini, A.: Arc Furnace Model for The Study of Flicker Compensation in Electrical Networks. IEEE Trans. Power Delivery, **8** (1994) 2026-2036
4. Mokhtari, H., Hejri, M.: A New Three Phase Time-Domain Model for Electric Arc Furnaces Using MATLAB. IEEE/PES Transmission and distribution Conference and Exhibition, **3** (2002) 2078-2083
5. King, P.E., Ochs, T.L., Hartman, A.D.: Chaotic Response in Electric Arc Furnace. Journal of Applied Physics, **76** (1994) 2059-2065
6. O'Neill-Carrillo, E., Heydt, G.T., Kostelich, E.J.: Nonlinear Deterministic Modeling of Highly Varying Loads. IEEE Trans. Power Delivery, **14** (1999) 537-542
7. Jang, G., Wang, W., Heydt, G.T.: Development of Enhanced Electric Arc Furnace Models for Transient Analysis. Electric Power Components and Systems, **29** (2001) 1061-1074
8. Ozgun, O., Abur, A.: Flicker Study Using A Novel Arc Furnace Model. IEEE Trans. Power Delivery, **17** (2002) 1158-1163
9. Kantz, H., Schreiber, T.: Nonlinear Time Series Analysis. Cambridge University Press (1997)
10. Chen, S., Cowan, C. F. N., Garnt, P. M.: Orthogonal Least Squares Learning Algorithm for Radial Basis Function Networks. IEEE Trans. Neural networks, **2** (1991) 302-309
11. Wen, X., Zhou, L., Wang, D.: Design Application of Neural Network Using MATLAB (in Chinese). Science Press Beijing (2002)

Modeling and Prediction of Violent Abnormal Vibration of Large Rolling Mills Based on Chaos and Wavelet Neural Networks

Zhonghui Luo[1,2], Xiaozhen Wang[1], Xiaoning Xue[1], Baihai Wu[2], and Yibin Yu[1]

[1] College of Engineering, Zhanjiang Ocean University, Zhanjiang, Guangdong, 524025 China
[2] Mechanical and Electronic Engineering College, Guangdong University of Technology, Guangzhou,Guangdong 510090, China
Lzh2382002@163.com

Abstract. This paper analyses the chaotic characteristics of violent abnormal vibration signals of a large rolling mill, and studies phase space reconstruction techniques of the signals. On this basis, the vibration model of wavelet neural networks and the model of backpropagation neural networks are set up, respectively, through inversion methods. The properties of these two models are tested and compared with each other. The result shows that the wavelet neural networks have an advantage over the backpropagation neural networks in rapid convergence and high accuracy.

1 Introduction

Violent abnormal vibration often occurs in a large rolling mill when rolling thin plates are below 2.0mm. The violent vibration results in ‘Vibratory Lines’ on the surface of the working rollers, and also on the surface of thin rolled steel plates. The roar at low frequencies arises because of varying vibration intensities. The violent vibration of the rolling mill speeds up the wear of the rollers, and affects the productivity of the rolling mill and the quality of products.

Many scholars at home and abroad have extensively studied the vibration of rolling mills, and established a variety of theories and methods [1], [2], [3]. In their papers, they have analyzed the mechanism of vibration of rolling mills from different aspects by establishing mathematical models of rolling mills. However, these theories are not fit for the abnormal vibration of rolling mills. Since a rolling mill is a complicated nonlinear system which consists of electromechanical and hydraulic equipment, and its working load is heavy and impact, it is difficult to establish a accurate mathematical model, and to solve the mathematical model. At present, the rapid advance in chaos provides ways to analyze a complicated nonlinear system. This paper presents a new method for modeling and prediction of violent abnormal vibration of large rolling mills based on chaos and wavelet neural networks.

2 Test for Violent Abnormal Vibration Signals of a Rolling Mill and Their Chaotic Characteristics

The violent abnormal vibration signals of a rolling mill have been sampled by using a multi-channel data acquisition when the rolling mill was rolling steel plates 1.6mm thick. Fig.1(a) shows the horizontal vibration signals of a working roller(the maximum amplitude signals). The sample frequency is 512Hz, and the sample size is 1024. Fig.1(b) shows frequency spectrum of the signals after fast Fourier transform(FFT). It is shown in Fig.1(b) that the frequency spectrum is a broad-band distribution, which indicates that the signals are non-periodic and non-stationary.

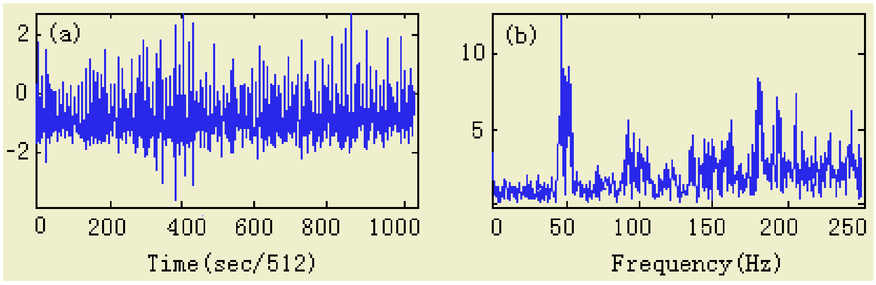


Fig. 1. Violent abnormal horizontal vibration signals of a working roller of a large rolling mill: (a) Time history, (b) FFT results.

The phase plane tracks of the signals show that the phase portrait is in a part area of the phase plane and is distributed in disorder. The maximum Lyapunov exponent of the signals is positive. These show that the vibration of the rolling mill has chaotic characteristics. So it is possible to inverse the kinetic model of the rolling mill system by means of phase space reconstruction based on chaotic theory, and to predict the future vibration in terms of the model [4].

3 Phase Space Reconstruction of Vibration Time Serial Signals

Studies on the chaotic characteristic of a system from time series started with the theory of phase space reconstruction put forward by Packed and Takens [5]. In the theory, it is considered that the evolution of any component of a system depends on other associate components. The information of these components is contained in the development of each of them. When a state space is reconstructed, it is necessary to consider only one of the components, and to use its delay points of some fixed time as new dimensions. By reasonably choosing the delay time and the dimension of phase space, it is possible to obtain a new system that has the same dynamic properties as the original dynamic system.

The phase space reconstructed by time serial signals x(1), x(2), ···, x(N) can be written as the form of a vector space:

$$J_i(X_{i,m}) = [x(i), x(i+\tau), x(i+2\tau), \cdots, x(i+(m-1)\tau)]^T. \tag{1}$$

where $X_{i,m}$ is called phase space vector, i=1, 2, ···, L, L=N-(m-1) τ, N is the sample size of the signals, L is the number of vectors of the reconstructed phase space, τ is delay time, and m is the embedding dimension. The process of reconstructing phase space is equivalent to the projection from one dimensional time series into m-dimensional Euclid's space. It should be emphasized that the choice of delay time(τ) and embedding dimension (m) is critical to the quality of phase space reconstruction. This reconstruction retains the evolution information of the original dynamical system, and they have the same dynamical properties.

3.1 Choice of Delay Time

In the process of reconstruction, if delay time(τ) is too large, any two adjacent delay coordinates of time series are not correlated at all, which does not reflect the characteristics of the whole dynamical system. On the other hand, if delay time is too small, any two adjacent delay coordinates of time series are close together, which leads to redundant data. Since a self-correlation function can provide a measure from superfluity to non-correlation between a signal itself and its delay, this paper uses a self-correlation function to determine the delay time, choosing the first zero-cross time as the delay time(τ). By using this method, the delay time(τ) of the vibration signals shown in Fig.1(a) has been found out, which is equal to 2.

3.2 Choice of Embedding Dimension

In 1980 Takens demonstrated that embedding dimension(m) should be $m \geqslant 2d+1$, where d is the space dimension of attractor of the original state space. At present, the ways commonly used to choose embedding dimension are saturation of system eigenvalue and singularity value decomposition [6]. This paper synthesizes the two methods. In this paper, the embedding dimension is initially determined by singularity value decomposition, then proved by saturation of system eigenvalue. This will make the results reasonable. By using this method, the embedding dimension (m) of the vibration signals shown in Fig.1(a) has been found out, which is equal to 24.

4 Modeling and Prediction of Wavelet Neural Networks for Vibration of Rolling Mills

In chaotic theory, it is considered that a chaotic time series is predictable in a short period. For a chaotic time series $\{x(t), t=1,2,\cdots,N\}$, the projection exists: $R^m \rightarrow R$

$$x[(k+1)+(m-1)\,\tau] = f\,(x(k), x(k+1), \cdots, x(k+(m-1)\,\tau)). \tag{2}$$

where m is the embedding dimension, and τ the delay time. The essence of modeling and prediction is how to gain a good approximation to f(•) and then how to predict the future phase space points in terms of the constructed model. Today, two main methods for modeling and prediction exist. One of them utilizes the nonlinear projecting capability of neural networks to establish a predictor of neural networks [7]. The other method is the local linear prediction model based on chaos [8]. This paper uses local nonlinear modeling and prediction based on wavelet neural networks. A wavelet neural network is a feed-forward network based on wavelet analysis. The original signals can be represented in terms of a wavelet expansion by shifting and scaling, thus having the properties of function approximation of wavelet decomposition. In the structure, a wavelet neural network is a neural network that just changes a traditional activation function f(t) into a wavelet function ψ(t). Wavelet neural network is a new and important branch of neural networks. It is developed through combining the wavelet theory with the artificial neural networks. The combination of wavelets and neural networks provides better performance for processing data at multiple scales and for adaptively adjusting wavelet parameters during data processing [9]. So the properties of wavelet neural networks are much better than those of the traditional BP networks.

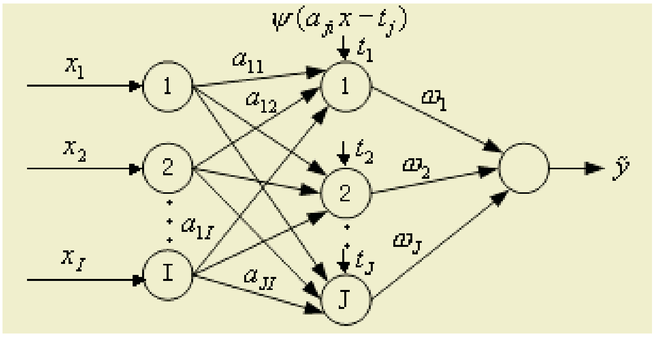


Fig. 2. Structure of a wavelet neural network.

The structure of a multi-input single-output wavelet network is showed in Fig.2, where the activation function of hidden neuron is not Sigmoid function, but is wavelet function ψ(t) (Gauss wavelet is used as a mother wavelet in this paper). In Fig.2, ω_i is the connection weight between the No.i neuron in the hidden layer and the output neuron, α_{ji} is the connection weight between the no. j neuron in the hidden layer and the No.i neuron in the input layer, which indicates the scale factor of the mother wavelet activation function ψ(t), and t_j is the shift factor of the mother wavelet. The output of the network is linear combination of the outputs of the hidden layer. According to the above, the reconstruction parameter of phase space of the vibration signals(shown in Fig.1) of a rolling mill is chosen as 24. So the parameters of the wavelet neural network can be chosen as: 24 nodes in the input layer, 25 nodes in the hidden layer and one node in the output layer.

The steps for modeling and prediction are as follows:

(1) Getting the phase space $J_i(X_{i,24})$ (i=1,2,····,L) by phase space reconstruction of the time serial signals.

(2) For example, if the first phase space vector $J_{k+1}(X_{k+1,24})$ after the vector $J_k(X_{k,24})$ is needed to be predicted, choose P(in this paper, P=100) vectors closest to the no. k vector as the input of a sample set, and choose the next step components of each of the P vectors as the output of the sample set.

(3) Using the input and output sample sets to train the wavelet neural network on line(training goal is 0.005), thus establishing the model of the wavelet neural network.

(4) Making a prediction of next vector by using the trained model of the wavelet neural network to obtain the predicted values.

According to the above, this paper programs the modeling of a wavelet neural network and making predictions step by step by using Matlab. In the program, the first 800 samples of the vibration signals in Fig.1(a) are used as the training sample set for modeling, and the last 200 samples are used as the test set. Fig.3(a) shows the predicted values and the measured values(in order to display clearly, 40 values are shown.).

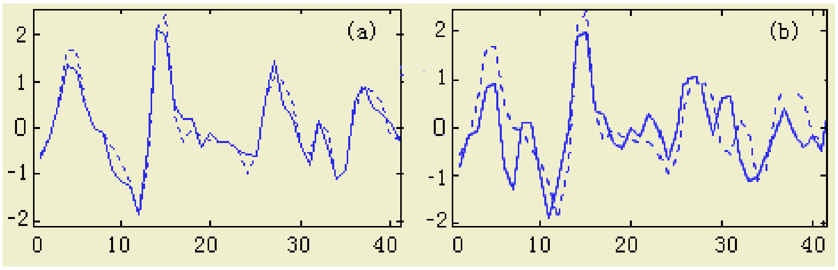


Fig. 3. The predicted signals of two models: (a) the model of Wavelet neural network, (b) the model of BP neural network. Herein, solid-line is the predicted signals, dashed-line is the real signals.

The prediction capability and the calculation speed of a wavelet neural network model are the major qualities for evaluating the model. To show the capability of the wavelet neural network, this paper has compared its model with the traditional BP network model. Fig. 3(b) shows the predicted signals of BP network model and the measured signals. Obviously, as shown in Fig.3(a), the prediction by means of the wavelet neural network model is more accurate. In order to evaluate the prediction accuracy quantitatively, mean error(MER) and mean square(MSQ) are used for the indices of evaluating.

According to the formulation of MER and MSQ, the prediction errors of the two prediction models have been calculated, respectively, the results of which are shown in No.1 in Table 1. The MER of the prediction of the wavelet neural network model is only 3.42%, which is less than half of that of the BP network model. The ratio of MER of these two kinds of models is 1:2.1, and the ratio of MSQ is 1:3.5. The calculation speed of the wavelet network is more than 1.3 times as fast as that of the BP network. In order to demonstrate the validity of the model and prediction presented in this paper, two other sets of abnormal vibration data have been tested, the results of which are shown in No.2 and No.3 in Table 1, where it is shown that the results of the wavelet neural network model are better than those of the BP neural network model.

Table 1. Comparison of errors of two prediction models.

	The wavelet neural networks			The BP neural networks		
Signals	Calculating time (sec.)	MER	MSQ	Calculating time (sec.)	MRE	MSQ
No.1	3.509	3.42%	0.024	4.692	7.38%	0.084
No.2	3.781	3.69%	0.029	4.971	7.65%	0.095
No.3	3.428	3.31%	0.021	4.538	7.41%	0.089

5 Conclusion

This Paper has dealt with a new method for modeling and prediction of violent abnormal vibration of rolling mills based on phase space reconstruction and wavelet neural networks by means of chaos, thus giving a successful solution to the problem of modeling and prediction of abnormal vibration. It has been shown by tests that the result of prediction of wavelet neural networks is better than that of BP networks.

References

1. Cheng, Y., Liao, G., Shi, T., et al.: Friction-type Chatter on 4-h Cold Rolling Mills. Journal of Huazhong University of Science & Technology, **31** (2003) 105-107
2. Tang, H., Li, X., Zhong, J.: Analysis Torsion Self-excited Vibration in Rolling Mill. Journal of Xiangtan Mine Institute, **17** (2002) 21-24
3. Wang, Q., Tan, J.: A Study of the Vertical Vibration of the Plate and Strip Rolling Mill Based on the Gap Dynamics. Mechanical Science and Technology, **22** (2003) 721-723
4. Teng, L., Liu, T., Tong, D., et al.: Application of Correlation Dimension to Machinery Condition Monitoring. Journal of Vibration Engineering, **15** (2002) 399-403
5. Packard, N. H., Crutchfield, J. P., Farmers, J. D., et al.: Geometry From a Time Series. Phys Rev Letter, **45** (1980) 712-716
6. Cao, L.: Practical Method for Determining the Minimum Embedding Dimension of a Scalar Time Series. Physic D, **110** (1997) 43-50
7. Wang, Y., Lin, C., Runge-Kutta: Neural Network for Identification of Dynamical Systems in High Accuracy. IEEE Trans. on Neural Networks, **9** (1998) 294-307
8. Xiang, X.: Prediction of Chaotic Time Series Based on Chaotic Theory and Radical Basis Function Neural Networks. Journal of Fuzhou University, **31** (2003) 401-404
9. Zhang, J., Walter, G., Miao, Y.: Wavelet Neural Networks for Function Learning. IEEE Trans. on Signal Processing, **43** (1995) 1485-1497

Neural Grey Box Model for Power Estimation in Semiautogenous Mill

Tito Valenzuela[1], Karina Carvajal[1], Gonzalo Acuña[1], Max Chacón[1], and Luis Magne[2]

[1] Departamento de Ingeniería Informática, Facultad de Ingeniería,
Universidad de Santiago de Chile,USACH, Avda. Ecuador 3659, Santiago, Chile
{tvalenzuela,klcarvajal,gacuna,mchacon}@diinf.usach.cl
[2] Departamento de Ingeniería Metalúrgica, Facultad de Ingeniería,
Universidad de Santiago de Chile, USACH, Avda. Ecuador 3659, Santiago, Chile
lmagne@lauca.usach.cl

Abstract. Technology advances in semiautogenous milling have not been accompanied by advances in the operation of these mills, mainly in the estimates of load levels and the power of the mill. This article presents a grey box model that improves estimate of power. The obtained results are satisfactory and demonstrate that both the phenomenological model and the neural network strengthen each other, and the results are better than those obtained individually.

1 Introduction

Milling in the mining industry has advanced significantly, specifically with regard to the technology of autogenous and semiautogenous (SAG) mills. In SAG mills this technological advance has not been accompanied by an advance in their operation because there has neither been a correct interpretation of the variables [1], nor a comprehension of internal laws that are involved in this process [2]. Two lines of research have arisen, one for predicting the load level through the use of sensors [3] and the other for control of energy through the online registry of the utilized power.

There are various attempts to develop equations to allow the estimation of the power in SAG milling, such as the equations by Austin, Taggart, Lukasiewicz, Mishra and Rajamani [4], [5]. Nonetheless, in a number of these models, the assumptions have led to a very simplified product, which is only useful in the analysis of low variability systems such as conventional ball mills [1].

The present work proposes an improvement of the power estimate that these models work with, using only a grey box model [6]. In this case, the power estimation models use a calibration factor K. The purpose of this work is to take advantage of neural networks in the modeling of complex functions [7] in order to find a function *f(K),* which can be added to the phenomenological model to improve its estimation.

The present article is divided as follows: the first section is a brief description of semiautogenous milling, the second section is a definition of grey box models which is followed by the used methodology and the obtained results ending with the conclusions.

2 Semiautogenous Milling

The purpose of Concentration Processes in mining is to recover particles of valuable substances which are found in mineralized ore. The process of concentration of copper is divided into three phases: Crushing, Milling and Flotation.

Semiautogenous (SAG) milling is a variety of the autogenous milling process in which metallic milling elements are added to the mill. The term autogenous indicates that milling occurs due to the falling of the mineralized rock from a height that is close to the diameter of the mill.

When working with these mills the ideal is to work under conditions that imply the use of the maximum installed power. However, this implies working at an unstable operational point because any increase in the load level of the mill beyond the point of maximum consumption leads to an overload condition and a loss in efficiency (see Figure 1).

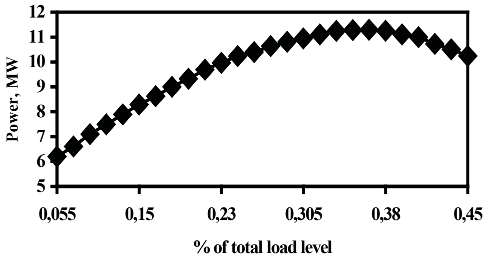


Fig. 1. Typical variation of power consumption with load expressed as a percent of total mill capacity.

The value of maximum power that a SAG mill can consume is not constant and depends primarily on the density of the internal load, the distribution of the size of the feedstock and the state of the internal lining. The load level is related to the amount of grinding media and the movement of the material being ground. Because of this, SAG mills operators must juxtapose these factors in order to first stabilize operations and then seek to improve them.

There have been numerous attempts to develop equations that allow a prediction of the amount of power that will be consumed. However, many of these models are too simple given the assumptions involved, and are only useful for systems with low variability such as conventional ball mills.

3 Grey Box Models

A Grey Box Model is a combination of two models, one that is based on available first principles about the different processes to be modeled, and another that can be used to identify parameters that are difficult to model with the a priori knowledge available.

In a Neural Grey Box Model, a priori knowledge corresponds to the phenomenological part of the model which is represented by the various differential equations which allow the modeling of the process under study. The empirical part corresponds, in this case, to a neural network [8]. It has been demonstrated that this combination allows an improvement over the predictions that a neural network alone can deliver, trained on scarce and noisy data [9].

Grey Box Models have significant advantages over the standard neural network model (Black Box Model); they can interpolate and extrapolate functions with far greater precision, analysis and interpretation are easier, and finally they require significantly less training data [10].

Grey Box Models can be classified into the following two types [9]: (1) Grey Box Models in series and (2) Grey Box Models in parallel. The Figure 2 shows the grey box model in series than is used in this work.

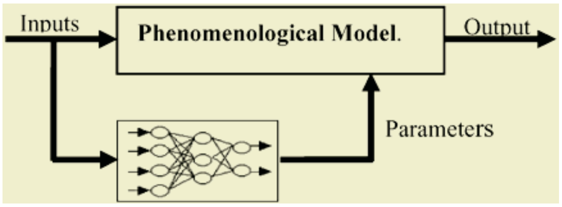


Fig. 2. Grey Box Model in series.

In MISO (multi-input single-output) systems, it has been demonstrated that the series scheme generates grey box models with improved performance in comparison to parallel models [11].

4 Description of the Methodology

For estimating of power in SAG milling the results of eight models were compared: two phenomenological, two black box models and four grey box models.

In the black and grey box models, 4466 registries were used, with six input and one output variables, all of which were measured during the operation of the mill. The output variable for all of these models is power. The input variables are: ball load (as a % of mill volume- Jb), total load (as a % of mill volume-Jc), water flow (Fa), mill rotation speed (N), pebble flow (Fp), and mineral flow (Fm).

The different models to be compared are: Austin's Model, Sastri-Rao's Model, Neural Model with six input variables, Neural Model with two input variables, after considering a sensitivity analysis, Grey Box Model with six and two input variables considering Austin's Model and a Grey Box Model with six and two input variables considering Sastri-Rao's Model.

4.1 Phenomenological Models

At the following, two models made by different authors are presented to estimate of power of rotatory mills. These models were chosen because of their simplicity and the availability of variable values involved in each model.

Austin's model for estimate of power is represented in equation (1).

$$P = KD^{2.5} L\rho_c J_c (1 - AJ_c)\phi_c \left(1 - \frac{0.1}{2^{9-10\phi_c}}\right) \tag{1}$$

Austin is based in Bond's model for power estimation in conventional mills [5], [6], [7], [8], [9], [10], [11], [12], where K is a constant (9.77); D is internal diameter of mill, m; L is length of cylindrical regions, m.; ρc is load density, t/m^3; Jc is total load, as a % mill volume, ϕc is percent critical velocity; A is a constant (0.9375) and P is power.

Equation (2) shows a simple power model, developed by Sastri and Rao [4], where total power is given by the power without load (first addend) plus the net power for the loaded mill (second addend), where V is mill volume, m^3; D is internal diameter of the mill, m; Jc is total load, as a % mill volume; ρc is load density, t/m^3; K is a constant (3.3) and P is power.

$$P_G = 2\left[VD^{0.5}\right]^{0.8} + KV\rho_c D^{0.5} J_c{}^{0.8} \tag{2}$$

4.2 Black Box Model

Two groups of data were considered for the neural network (one for training and the other for testing). Training is carried out with the cross validation method and is repeated 10 times in order to ensure the trustworthiness of the results.

For the neural model, the output variable is power, while the input variables are Jb, Jc, Fa, N, Fp and Fm.

During the search for the best configuration various architectures were considered which used between 2 and 25 neurons in the hidden layer. The configuration chosen was that which gave the least mean and least deviation in the error between the desired power and that given by the model.

The two configurations finally chosen were: (1) a neural network with six inputs, one hidden layer with 8 neurons and a hyperbolic tangent sigmoid transfer function in both layers. The second configuration is (2) a neural network with two inputs (Jb, Jc, obtained after a sensitivity analysis) and one hidden layer with 11 neurons.

4.3 Neural Grey Box Models

The phenomenological models used as a basis for the Neural Grey Box Models are those proposed by Austin [5], [6], [7], [8], [9], [10], [11], [12] and by Sastri-Rao [4]. The outputs of the neural network, included in the Grey Box Model, correspond to the value of the parameter K, which minimize the error for the calculation of power. This corresponds to the so-called direct training method. [8].

The final configurations are four, two for each of the power models. In Austin's model, the first of the grey box configurations is made up of equation (1) as a phenomenological model and a neural network with six inputs, one hidden layer with 15 neurons. The second configuration for Austin's model is analogous to the first, but with only two inputs (Jb and Jc) and 18 neurons in the hidden layer.

For the Sastri-Rao model, the first grey box configuration is composed of equation (2) as a phenomenological model and a neural network with six inputs, one hidden layer with 17 neurons. The second grey box configuration for this model is analogous to the first, but with two inputs (Jb and Jc) and 14 neurons in the hidden layer. The neural network's output, in both models, is calibration factor K that is then added to the model shown in equation (2).

Each model used a hyperbolic tangent sigmoid transfer function in each layer.

5 Results

Table 1 shows the results of the estimates obtained with each of the models presented above. The first column shows the model used for estimating of power, the second shows the type of model, the third shows the mean difference between the desired power and the power estimated by the different models, and the last column specifies the particular configuration for each model. For this last column, in the case of the phenomenological models the values of K are shown, and in the case of black box and grey box is shown the number of inputs into the network (I), the number of neurons in the hidden layer (H), and the number of outputs (O) in the order I-H-O.

As can be seen in Table 1, the black box models show a marked improvement over the phenomenological models in terms of power estimate. This improvement is even more noticeable when looking at grey box models.

The obtained results show a 1% error of the mean of power consumed (5950.5 kw) by the mill which means that it is possible to work at an operation point where milling conditions are more stable and more efficient.

This shows the advantages of using grey box model instead of neural or phenomenological model that means using the combination of both models.

Table 1. Results of different models.

Model	Type	Error (Kw)	Configuration
Phenomenological	Austin	92.14	K= 9.77
	Sastri-Rao	359.53	K= 3.3
Black Box	Neural Network	74.20	6-8-1
	(NN)	69.61	2-11-1
Grey Box	NN + Austin	44.77	6-15-1
		61.89	2-18-1
	NN + Sastri-Rao	36.10	6-17-1
		163.76	2-14-1

6 Conclusions

This work shows that the fusion between black box models and phenomenological models, also known as grey box models, is an alternative that improves the results obtained by the first two individually. On the other hand, these improved results imply the use of a larger number of parameters, for both black and grey box models, than the used ones in the phenomenological models.

The obtained results show an error on the order of 1% of the mean of power consumed (5950.5 kw) by the mill which means that it is possible to work at an operation point where milling conditions are more stable and more efficient.

The importance of estimating of power with a minimum error is that it makes it possible to estimate other variables obtaining small errors in these estimates as well. On the other hand, an improvement of 1% in the estimates of the variables that affect the process of semiautogenous milling implies a reduction of US$10 million in operational costs per year.

Given the good results obtained in this work, future work will include estimates with grey box models in parallel.

Acknowledgements

We would like to thank the partial financing provided for this work by the Chilean Government through Fondecyt 1040208 and Fondef D02I1077 projects.

References

1. Magne, L., Titichoca, G., Campi, A.: Semiautogenous Mill Plant Operation: A Tough Reality. Part 2. The Compsumtion of Power, VI SHMMT/XVIII ENTMME, Brasil (2001)
2. Magne, L., Améstica, R., Barría, J. and Menacho, J.: Dynamic Modeling of Semiautogenous Milling Based on A Simplified Phenomenological Model. Metal Madrid , **31** (1995) 97-105
3. Van Dyk, W., Stange, W., Hatch Africa (Pty) Ltd.: Optimum Control of the Leeudoorn Semiautogenous Milling Circuit. CIM Bulletin, **93** (2000) 106-110
4. Sastri, S.R.S., Rao, K.K.: Simple Empirical Equations for Prediction of Mill Power Draw. Mineral Processing Extractive Metallurgy, (1997) 91-94
5. Austin, L.G., Klimpel, R.R., Luckie, P.T.: Process Engineering of Size Reduction Ball Milling. American Institute Mining Engineer, New York, U.S.A. (1984)
6. Cubillos, F.Y., Lima E.: Identification and Optimizing Control of A Rougher Flotation Circuit Using An Adaptable Hybrid-Neural Model. Minerals Engineering (1997) 707-721
7. Hornik, K., Stinchcombe, M. Y White H.: Multilayer Feedforward Networks are Universal Approximators. Neural Networks, **2** (1989) 359-366
8. Acuña, A., Cubillos, F., Thibault, J., Latrille, E.: Comparison of Methods for Training Grey-Box Neural Networks Models. Computers and Chemical Engineering Supplement, **23** (1999) 561-564
9. Thompson, M., Kramer, M.: Modeling Chemical Processes Using Prior Knowledge and Neural Networks. Computer & Chemical Engineering, **40** (1994) 1328-1340
10. Psichogios, D., Ungar, L.: A Hybrid Representation Approach for Modeling Complex Dynamic Bioprocesses. Bioprocess Engineering, **22** (1992) 547-556
11. Van Can, H, Braake, H., Dubbelman, S., Hellinga, C., Luyben, K., Heijnen, J.: Understanding and Applying the Extrapolation Properties of Serial Gray-Box Models. AIChE Journal, **44** (1998) 1071-1089
12. Austin, L.G., et al.: A General Model for Semiautogenous and Autogenous Milling. South Africa institute Mining Metallurgy, Johannesburg, South Africa (1987) 107-126

Neural Network Based On-Line Shrinking Horizon Re-optimization of Fed-Batch Processes

Zhihua Xiong[1], Jie Zhang[2], Xiong Wang[1], and Yongmao Xu[1]

[1] Department of Automation, Tsinghua University, Beiijng 100084, China
{zhxiong,wangxiong,xym-dau}@tsinghua.edu.cn
[2] School of Chemical Engineering and Advanced Materials, University of Newcastle, Newcastle upon Tyne, NE1 7RU, UK
jie.zhang@ncl.ac.uk

Abstract. Neural network is used to model fed-batch processes from process operational data. Due to model-plant mismatches and unknown disturbances, the off-line calculated control policy based on the neural network models may no longer be optimal when applied to the actual process. Thus the control policy should be re-optimized. Based on the mid-batch process measurements, on-line shrinking horizon optimization is carried out for the remaining batch period. The iterative dynamic programming algorithm based on neural network models is developed to solve the on-line optimization problem. The proposed scheme is illustrated on a simulated fed-batch chemical reactor.

1 Introduction

Interests in fed-batch processes are concerned with determining the feeding policy to the reactor that will give the maximal yield of the desired product [1]. It is very important for optimal control of fed-batch processes to obtain an accurate model that is capable of providing accurate long range predictions. To overcome the difficulties in building mechanistic models, empirical models based on process input-output data can be utilized. Neural networks (NNs) have been shown to be able to approximate any continuous nonlinear functions and have been applied to nonlinear process modeling and control [2]. If properly trained and validated, these NN models can be used to predict steady-state and dynamic process behavior reasonably well, hence, leading to improved process optimization and control performance [3]. Using NN models to predict the long-range dynamic behavior of a fed-batch process, the optimal control policy for the fed-batch processes can be calculated off-line [4].

In general, optimization of fed-batch process can be described as an end-point optimization problem of a non-linear control-affine batch process [5]. Computational methods of this kind of problems usually have to cope with problems such as numerical evaluation of derivatives and feasibility issues. Iterative dynamic programming (IDP) [6], as one of direct search methods, has been applied successfully to solve the optimization problem of fed-batch processes [4],[7]. It has the high possibility of obtaining the global optimum and avoids such problem as singular control. But the main disadvantage of IDP is the low efficiency and computationally time-consuming, resulting from the necessity of solving the non-linear differential-algebraic equations (DAEs) of the mechanistic models in each iteration. The replacement of the mechanistic model by an equivalent NN model in IDP takes the advantage of high speed processing, since simulation with an NN model involves only a few non-iterative algebraic calculations [4].

Due to unknown disturbances during the operation of the batch and model-plant mismatches, the off-line calculated optimal control profile based on the NN models can be no longer optimal when applied to the actual process. There is a need to re-optimize the process on-line using updated process information [8]. During on-line re-optimization, both control horizon and prediction horizon for fed-batch processes shrink as the process progresses [9]. In this study, the focus is on overcoming the problem of model-plant mismatches and improving the computation efficiency of the on-line shrink horizon re-optimization control strategy.

2 Fed-Batch Process Modeling Using Neural Networks

Many fed-batch processes can be considered as a class of control-affine non-linear systems, mathematically formulated as [5]

$$\dot{x} = f(x)+g(x)u, \quad x(0)=x_0. \tag{1}$$

where $x \in R^p$ is a vector of states, $u \in R^q$ is the feeding rate, and f and g are non-linear functions. In fed-batch processes, the maximal yield of the desired product is obtained by employing an optimal feeding policy to the reactor while the volume of reactor and feeding rate are constrained. This leads to a constrained end-point optimal control problem of a non-linear system, which can be mathematically formulated as [5]

$$\min_{u(t)} J(u(t)) = \phi(x(t_f)) \, . \tag{2}$$

$$s.t. \quad \gamma(x, u) \le 0. \tag{3}$$

where ϕ is a scalar constrained function, $\gamma(x,u)$ are constraint functions, and t_f is the final time (at the end of a batch). To reduce the complexity of the optimization problem, the feeding policy is here parameterized as a piecewise constant function.

As shown in Eq(2), the performance index only includes the end-point values of product variables x. Neural network models can be used to represent the non-linear relationship between product variables x and feeding policy u of fed-batch processes. In this study, feed-forward neural networks (FNN) [2] are used to model the fed-batch process, and an FNN model can be described as

$$y(k+1) = f(y(k),\dots,y(k-n_y+1),\, u(k),\dots,u(k-n_u+1)). \tag{4}$$

where $y=x$, n_u and n_y are the maximal lags in the input and output of the NN model respectively, known as the orders of discrete-time system, and subject to $(n_y+n_u) \ge n$. Given the initial conditions and the feeding policy of a fed-batch process, an FNN model can predict recursively the product at the end of a batch and obtain long range or multi-step-ahead predictions. The FNN can be trained by using the Levenberg-Marquart optimization algorithm with regularization [4]. Based on the trained FNN model for a fed-batch process, the optimal feeding policy can be calculated off-line.

3 On-Line Shrinking Horizon Re-optimization

An important issue in fed-batch process optimization is that the optimization effort can be seriously hampered by the presence of unmeasured disturbances and/or model-plant mismatches [4]. The off-line calculated optimal control profile may not be "optimal" when applied to the actual process. This issue can be overcome by using on-line re-optimization or the midcourse correction policy [8], where on-line information during the earlier stages is used to re-calculate the control profile of the remaining batch stages.

During on-line re-optimization, the batch length is divided into several equal stages. An initial "optimal" control profile containing piecewise constant control actions over all stages is first calculated off-line based on the NN model, and the control action for the first stage is implemented. At next sampling time, the actual process products are measured and compared with the NN model predictions, and their differences are used to verify the existence of model-plant mismatches and/or disturbances. If model-plant mismatches/disturbances exist and the original "optimal" control profile is still used for the remaining batch stages, the process products at the batch end will be significantly different from the desired products. So there is a need to re-optimize the control profile for the remaining batch stages. Based on the currently measured process operating conditions, the "optimal" control profile for the remaining batch stages is calculated again and the first part of this new control profile is then implemented. The re-optimization procedure is repeated at the next stage until the final stage of the batch. In this procedure, both control horizon and prediction horizon for fed-batch processes shrink as the process progresses [9]. The shrinking horizon terminology stems from the fact that the number of control actions remaining to be chosen decreases as a fed-batch progresses. Under the strategy of on-line re-optimization, the optimal control performance can be gradually improved.

In this study, the on-line re-optimization problem is solved by IDP algorithm. The original IDP algorithm [6] uses continuous-time mechanistic models in the form of differential equations. In some cases, the computation time is very large, so it is very difficult to use IDP for on-line optimization and control. The authors have developed a modified IDP algorithm replacing the mechanistic model by an equivalent NN model [4], where some steps in the original IDP algorithm have been modified. First, the grid points of IDP algorithm has to change from the state variables of mechanical model to all the input variables of the NN model, as shown in Eq(4). Second, the NN model is applied to predict the product at each time stage and at each iteration instead of the rigorous mechanistic model. Prediction based on an NN model only involves a few non-iterative algebraic calculations, thus the time cost decreases significantly.

4 Simulation on a Fed-Batch Reactor

This example involves reactions taken place in an isothermal semi-batch reactor, whose schemes are $A + B \xrightarrow{k_1} C$ and $B + B \xrightarrow{k_2} D$. The differential equations describing the reactions are given by [10]. The objective is, through addition of reactant *B*, to convert as much as possible of reactant *A* to the desired product, *C*, in a specified time t_f=120 min. The performance index in this fed-batch reactor is to maximize the amount of final product, $[C](t_f)V(t_f)$, while limiting the amount of the final undesired species, $[D](t_f)V(t_f)$, where *V* is the liquid volume of the reactor. The manipulated variable is the feed rate of reactant *B* and is constrained by $0 \leq u \leq 0.01$, and the volume of the reactor is limited by $V(t_f) \leq 1$.

In this study, we use an FNN to model the non-linear relationship among [C], [D] and u, while $V(t_f)$ is obtained by integrating u. Several batches of the reactor operation under different feeding policies were simulated from the mechanistic model to produce the data set for FNN modeling. The feeding policies are chosen as a random sequence from a uniform distribution in [0, 0.01]. The generated data set was divided into training data, consisting of 40 batches, testing data of 20 batches, and unseen model validation data of 1 batch. Networks were trained using the Levenberg-Marquardt algorithm with regularization and an "early stopping" mechanism to overcome the problem of over-fitting [4]. The appropriate network topology was determined by examining their sum of squared errors (SSE) on the testing data. The selected FNN has the structure: 5-10-5-2, which is described as

$$\begin{bmatrix} y_1(k+1) \\ y_2(k+1) \end{bmatrix} = f_{NN}[y_1(k), y_1(k-1), y_2(k), y_2(k-1), u(k)] \quad (5)$$

$$V(N) = \sum_{k=0}^{N-1} u(k)h + V(0)\,. \quad (6)$$

where y_1=[C], y_2=[D], $y_1(0)=0$, $y_2(0)=0$, $V(0)=0.5$, N=10, and $h=t_f/N$ is the interval time. After FNN training, long-range predictions of [C] and [D] from the FNN model on the unseen validation batch are shown in Fig 1. It can be seen that these predictions are quite accurate. Using the penalty function technique and taking into account the final state inequality constraint [4], the augmented performance index was chosen as

$$\min_{u(k)} J(u) = -[w_1 y_1(N)V(N) - w_2 y_2(N)V(N)] + \lambda \sum_{k=1}^{P} p(k)\,. \quad (7)$$

$$s.t.\quad V(N) \le 1,\quad 0 \le u(k) \le 0.01. \quad (8)$$

$$p(k) = \begin{cases} 0, & \text{if } \ (V(k) \le 1) \\ V(k)-1, & \text{if } \ (V(k) > 1) \end{cases}. \quad (9)$$

where w_1 and w_2 are weighting factors and both set to 1, λ is a penalty function factor and set to 0.1. To investigate the performance of the proposed strategy, three cases are studied: Case1 – off-line optimization based on mechanical model; Case 2 – off-line optimization based on NN model; Case 3 – on-line shrinking horizon re-optimization based on NN model. The IDP algorithm is used to solve the non-linear optimization problems in all three cases, where the following parameters in the IDP algorithm were chosen: time stage P=10, a reduction factor α=0.7, initial control policy u^0= 0.005, an initial control region size $r(0)$=0.005, the number of grid points Q=25, allowable control values M=9, and iterations T=10. All the calculations were carried out using MATLAB running on a Pentium 800 personal computer, and the MATLAB command ODE45 was used for integration of the DAEs in the mechanistic model.

In Case 1, the performance index value is 0.0439 but it costs 36.95 minutes CPU time because of solving the DAEs of the mechanistic model. It is much longer than the control interval time h (12 minutes), so it is impossible to be used on-line. On the contrary, in Case 2, based on the NN model, the off-line optimization only takes 2.38 minutes CPU time, a reduction of 15.5 times compared with Case 1, making it possible to be implemented on-line. The actual performance index in Case 2 is 0.0362, a 17.5% reduction compared to Case 1, due to model plant mismatches. The feeding

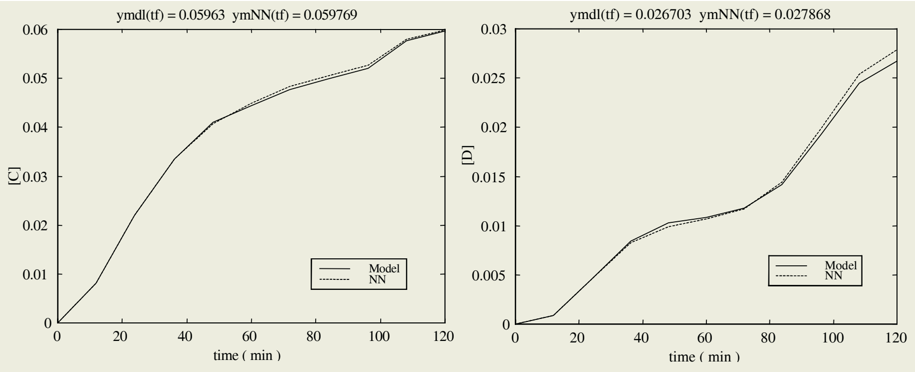
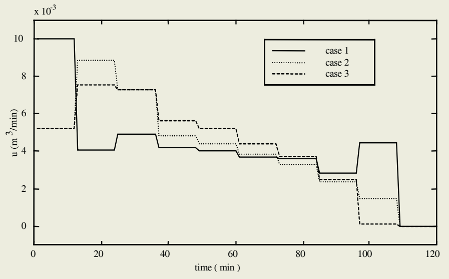


Fig. 1. NN model predictions of [C] and [D] on a validation batch compared to its true values.



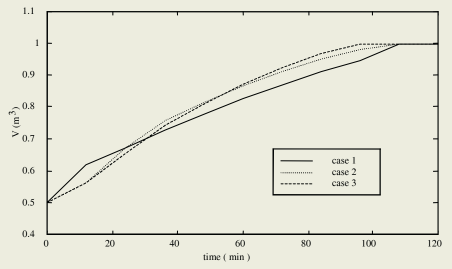


Fig. 2. Optimal feeding policies in all cases.

Fig. 3. Volume V under optimal control profiles.

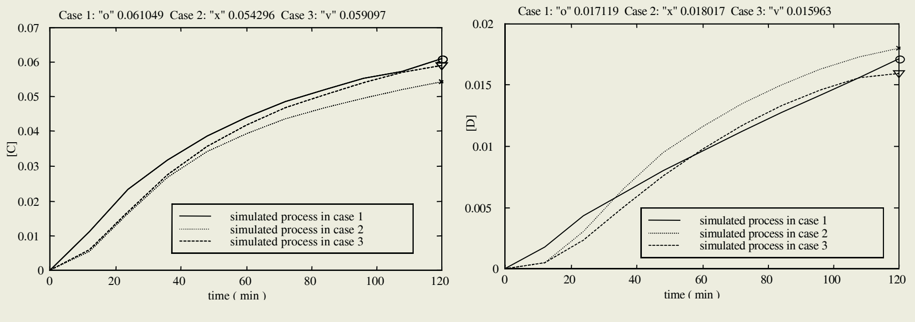


Fig. 4. Actual and predicted values of [C] and [D] in Case 3 compared to Case 1 and Case 2.

policy in Case 2 was used as the initial feeding policy in Case 3. After on-line shrinking horizon re-optimization in Case 3, the actual performance index is improved to 0.0431, only 1.8% less than that in Case 1. The entire computation time for the ten shrinking feeding policies is only 8.91 minutes. The piecewise constant optimal feeding policies of all three cases are shown in Fig 2. Fig 3 shows the volume of the reactor and the final volumes in all cases are 1.0 exactly. This demonstrates that the final state constraints are satisfied using the penalty function technique. In Fig 4, the actual values of products [C] and [D] based on the NN model in Cases 3 are compared with those in Case 1 and Case 2. From the above simulation results, it can be observed that the problem of model plant mismatches is significantly alleviated by on-line re-optimization strategy.

5 Conclusions

Neural network models are developed from process operational data to address the difficulties in developing mechanistic models, and the IDP algorithm based on neural network is presented to solve optimization problems. The off-line calculated control policy may not be optimal when applied to the actual process. To overcome the effects of model-plant mismatch, on-line shrinking horizon re-optimization is carried out to find the optimal control actions for the remaining batch periods. The proposed strategy is demonstrated effectively on a fed-batch chemical reactor.

Acknowledgements

The authors greatly appreciate the financial support of the NSFC (No. 60404012).

References

1. Bonvin, D.: Optimal Operation of Batch Reactors – a Personal View. J. Process Contr, **8** (1998) 355–368
2. Sjoberg, J., Zhang, Q., Ljung, L., Benveniste, A. et al.: Nonlinear Black-box Modeling in System Identification: a Unified Overview. Automatica, **31** (1995) 1691–1724
3. Bhat, N., McAvoy, T.: Use of Neural Nets for Dynamic Modeling and Control of Chemical Process Systems. Comput. Chem. Eng, **14** (1990) 573–583
4. Xiong, Z.H., Zhang, J.: Neural Network Model-based On-line Re-optimization Control of Fed-batch Processes Using a Modified Iterative Dynamic Programming Algorithm. Chem. Eng. Process, **44** (2005) 477–484
5. Visser, E., Srinivasan, B., Palanki, S., Bonvin, D.: A Feedback-based Implementation Scheme for Batch Process Optimisation. J. Process Contr, **10** (2000) 399–410
6. Luus, R.: Optimal Control by Dynamic Programming Using Systematic Reduction in Grid Size. Int. J. Control, **51** (1990) 995–1013
7. Nascimento, C.A., Giudici, R., Guardani, R.: Neural Network Based Approach for Optimisation of Industrial Chemical Processes. Comput. Chem. Eng, **24** (2000) 2303–2314
8. Lyer, M.S., Wiesner, T.F., Rhinehart, R.R.: Dynamic Reoptimization of a Fed-Batch Fermentor. Biotechnol. Bioeng,**63** (1999) 10–21
9. Soni, A.S., Parker, R.S.: Closed-Loop Control of Fed-Batch Bioreactors: A Shrinking-Horizon Approach. Ind. Eng. Chem. Res, **43** (2004) 3381–3393
10. Terwiesch, P., Ravemark, D., Schenker, B., Rippin, D.W.T.: Semi-batch Process Optimiza-tion under Uncertainty: Theory and Experiments. Comput. Chem. Eng, **22** (1998) 201–213

Chip Speed Prediction Model for Optimization of Semiconductor Manufacturing Process Using Neural Networks and Statistical Methods

Tae Seon Kim

School of Information, Communications and Electronics Engineering
Catholic University of Korea, Bucheon, Korea
tkim@catholic.ac.kr
http://isl.catholic.ac.kr

Abstract. As the increase of device complexity, prediction of chip performance such as chip speed is crucial as much as yield prediction model in semiconductor manufacturing. In this paper, hybrid of circuit simulation and DOE (design of experiment) are applied to develop speed prediction models for high-speed microprocessor manufacturing process. Speed prediction models are developed through three steps; transistor ratio variation analysis step, electrical test data analysis step, and fabrication test data analysis step. Artificial neural networks are used to find relation between measured data and chip speed and network inputs are selected based on statistical analysis. Electrical test based modeling results showed only 1.2% of RMS error and fabrication level speed prediction model showed 83% of fitness results.

1 Introduction

Generally, chip yield is considered as one of the most critical index to demonstrate manufacturing performance as it greatly impacts to manufacturing cost, time, and profits [1]. However, chip yield is not sufficient to demonstrate manufacturing performance for recent highly innovative semiconductor. Especially for high performance logic device such as microprocessors, chip speed and reliability requirements are equally crucial to functional requirements. Conventionally, chip yield and performance are easily evaluated through measured ET (electrical test) data including transistor (Tr) source-drain saturation currents, Tr threshold voltages and breakdown voltages. However, with increase of chip complexity and reduced device scale, conventional ET data were not enough to describe overall electrical chip performance for high performance chips. As the device feature size continues to shrink and performance is elevated, the performance variations are also increased. Process engineer no longer find cause of ET data variation directly from process conditions. Therefore, systemic process optimization methods are required for semiconductor manufacturing. There are various efforts to optimize semiconductor manufacturing process. However, they are mainly focused only on maximizing manufacturing yield or optimizing specific unit process such as lithography step or plasma etching step and several research results showed relation between yield and reliability of ICs [2][3].

In this work, we focused on development of speed prediction model to optimize manufacturing of high-speed microprocessors. For high-speed microprocessor, speed is the most critical performance factor since it directly relates on production yield and

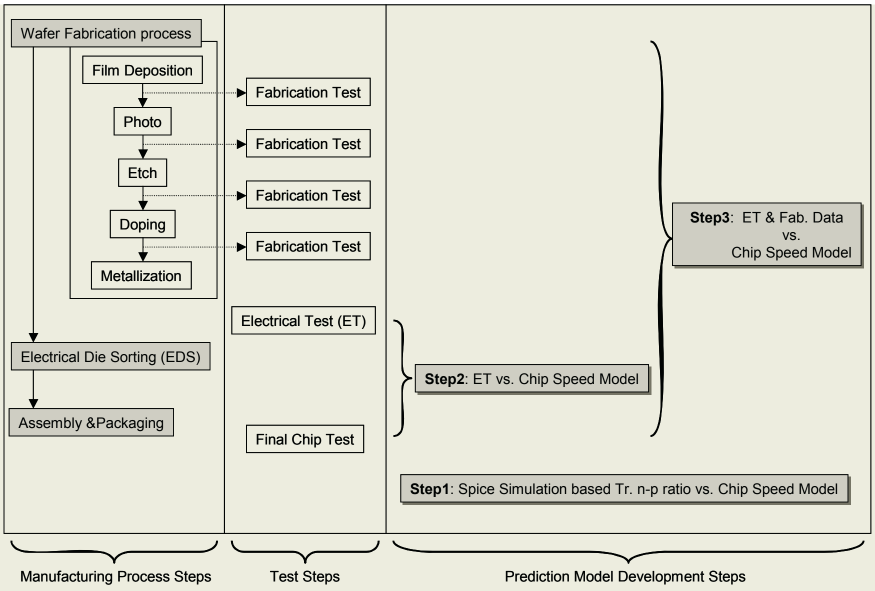

Wafer Fabrication process
Film Deposition
Photo
Etch
Doping
Metallization
Fabrication Test
Fabrication Test
Fabrication Test
Fabrication Test
Electrical Test (ET)
Final Chip Test
Electrical Die Sorting (EDS)
Assembly &Packaging
Step3: ET & Fab. Data vs. Chip Speed Model
Step2: ET vs. Chip Speed Model
Step1: Spice Simulation based Tr. n-p ratio vs. Chip Speed Model
Manufacturing Process Steps
Test Steps
Prediction Model Development Steps

Fig. 1. Overview of the semiconductor manufacturing process and three-step speed prediction modeling method.

sales price. In general, the market price of high-end microprocessors is exponentially proportional to device speed. However, for high-end microprocessor fabrication a process, device operation speed was not unique even every chips are fabricated based on same process recipes. If operation speed is below manufacturing specification, they have to scraped even they function correctly. Therefore, it is essential to find control parameter to optimize overall processor speed. For this, we developed three-step speed modeling approaches for process optimization using computer simulation analysis and DOE (design of experiment) data. At the first step, we find the optimal ET values using statistical SPICE simulation. Simulation based optimal ET values are used to optimize Tr n-p ratio. And then, correlation between chip speed and ET data are modeled at the second step, and finally the relation between ET data and fabrication data such as film thickness and line width of Tr gate oxide are modeled. Based on the results, new process conditions are set to improve chip speed and verified. Figure 1 shows the overview of semiconductor manufacturing process and developed three-step speed modeling method.

2 Three-Step Speed Prediction Modeling Method

2.1 Simulations for Tr n-p Ratio Analysis

To improve chip performance and manufacturing yield, semiconductor devices are modified using real wafer fabrication data. However, it is not desirable since it requires undesirable production data gathering period. Therefore, in term of time and cost, use of combination of simulation and DOE is much effective than use of only real processing data. In general, several modifications are made in manufacturing stages to improve chip performance and manufacturing yield through mask changes and device parameter changes. However, they also made unwanted changes on device properties. For example, increase of source-drain saturation current (I_{dsat}) increase device speed. But it may increase of leakage current, breakdown voltage variations, or even unknown changes. Therefore, it is not easy to find the optimal ET values in terms of device performance. In this work, we find optimal ET values using SPICE simulation. For this, we set the simple circuit which can represent overall processor performance such as power consumption and speed. The BSR (boundary scan register) chain was chosen as a critical circuit and this circuit delay time is considered as index of chip performance since chip speed is one of the most critical performance factors for high-end microprocessors. After determination of critical circuit and performance index, we set the experimental data to be simulated. Through this simulation, SPICE model parameter sets for experimental points are obtained and they are used to find optimal Tr n-p ratio which is one of important factor to design CMOS Tr.

2.2 Analysis of Correlation Between Chip Speed and ET Data

The simulation based optimized Tr n-p ratio results guide fabrication process to push either nMOS Tr or pMOS Tr for performance improvement. This is very prompt and effective way to improve the chip performances for CMOS process. However, various delays can be occurred on microprocessors from interference, interconnections and etc. Therefore, it is desirable to find highly correlated parameters to chip speed among ET data. For this, both of statistical methods and artificial neural network methods are used. For statistical way, correlation analysis and multiple regression method were used. Based this results, sensitivity analysis are performed to find relation between ET parameters and BSR delay. For neural way, error back-propagation (BP) neural networks are used. This speed prediction model can be used to scrap before package and assembly step. If lower speed devices can be effectively scrapped before packaging step, this can be a good alternative way to minimize undesirable manufacturing cost and time. In terms of cost, generally silicon package cost is around 50% of overall manufacturing cost for high-end microprocessors. In terms of time, it is desirable to scrap as quickly as possible if the final chip performance is below specifications. And also, ET data based speed prediction model gives crucial information to analysis of fabrication data to improve overall chip performance.

2.3 Analysis of Correlation Between ET Data and Fabrication Data

In general, fabrication data can be used to control manufacturing process. However, generally fabrication data based speed prediction model showed poor performance because of technical difficulties and testing cost. Manufacturing of high-end microprocessors requires several hundreds of sequential processing steps and the number of test parameters on fabrication level is more than 200 per wafer. Also, consecutive processing steps are highly correlated each other. Therefore, it is desirable to select key parameters that have highly correlation with speed. In manufacturing stages, only of fabrication data can be directly controlled. And even, among the fabrication factors, only of limited factors can be directly controllable because of limitation of equipment and technical difficulties. Therefore, we must find controllable fabrication parameters which have highly related to ET parameters that we need to change. This fabrication data based speed prediction model can be used to improve overall chip speed since it can guide controllable fabrication factors for speed improvement.

3 Modeling Results

In this work, BSR chain was selected as a critical circuit and its delay time is considered to represent final chip speed as a performance index. The final full chip performance values are measured from 627 production wafers. To set the experimental points to be simulated, we extract the 31 using experimental design method. For n-p ratio optimization, we choose source-drain saturation current variables (I_{dp} and I_{dn}) and threshold voltages (V_{tp} and V_{tn}) of nMOS and pMOS as input variables. SPICE simulation for this ten corner points, Tox, Vth0, U0, Lint, and Wint were used as BSIM3v3 MOS model parameters. The response surface design for four factors (V_{tp}, V_{tn}, I_{dp} and I_{dn}) gives optimal ET values. Figure 2 shows the contour plot of speed versus normalized Tr currents. In this figure, darker area represents faster chip speed and we can notice that increase of pMOS current is critical to chip speed and the design results showed that we need to change n-p ratio to 2.135. For speed prediction model using ET data, same circuit (BSR chain) was used as a critical circuit to represent final full chip speed. Total sixty kinds of electrical parameters are characterized in ET phase. They include basic I-V Tr characteristics for 3 different sizes, via resistances, capacitance between metal layers and etc. ET data were measured at every five points per wafer, and median of measured values are used. For relation analysis between chip speed and ET data, production data from 852 wafers are used. Statistical regression modeling method was performed using commercial statistical analysis tool, MINITAB. Statistical prediction model showed 92.2% of adjusted R-squared value. However, still it has some outliers at the edge of high and low speed regions. For neural prediction, three-layered BP network was used. Total of 60 ET parameters are used as network inputs and delay time of critical circuit was used as neural network output. The numbers of hidden neurons, learning rate and momentum values are heuristically optimized. Neural speed prediction model shows only 1.2% of RMS (root mean squared) error and it showed significant small number of outlier for entire speed ranges.

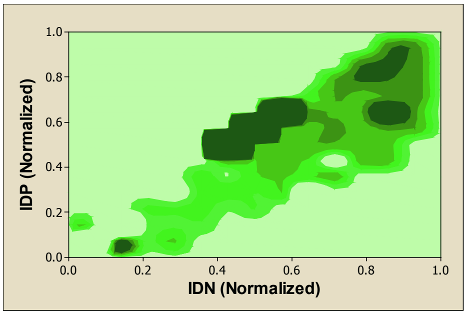
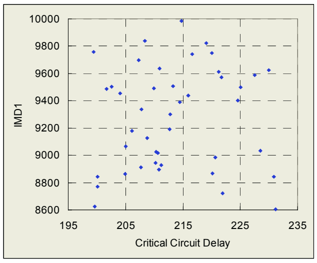


Fig. 2. Contour Plot of Speed versus Normalized Tr currents. The darker area represents faster chip speed.

Similar to ET data based speed prediction model, statistical regression modeling method was applied to model fabrication data based speed prediction. However, it shows only 62.0% of adjusted R-squared value with five key prediction parameters. Since the Tr gate CD (critical dimension) shows wide difference sensitivity to other fabrication parameters, it is desirable to remove effect of variation of Tr gate CD for analysis of other parameter data.



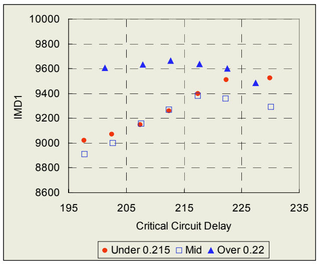


Fig. 3. Scatter plot of critical circuit delay versus IMD1. Left plot is transformed to right plot through separation and grouping scheme.

Fig. 3 shows one good example. Left plot of Figure 3 is the scatter plot of critical circuit delay and thickness of the first inter-layer dielectric film (IMD1). In this figure, we can't find any correlation between two factors. Right plot of Figure 3 is alternative representation of same data. For this plot, we used two schemes, separation and grouping. From right plot of Figure 3, we can see that thickness of IMD1 is proportional to circuit delay, but the thickness of IMD1 is not significant in our process range if the Tr gate CD is wider than 0.22 μm. Based on this separation and grouping method, seven highly correlated factors are selected as an input of prediction models and chip speed converted from delay time of BSR chain was selected as output. Similar to ET based prediction model, three-layered BP network was used and it showed 17% of fitness error (83% of fitness). Table 1 shows the comparison of prediction results between multiple regression methods and proposed three-step neural prediction method.

Table 1. Modeling RMS error compariosn between multiple resressions and three-step neural prediction modeling method.

	Multiple Regressions	3 Step Neural Prediction
ET vs. Chip Speed	7.8%	1.2%
ET & Fab. vs. Chip Speed	38%	17%

4 Conclusions

In this paper, neural networks based three-step chip speed prediction modeling method was developed for high-speed microprocessor manufacturing process. Speed prediction models are developed at transistor size optimization level, electrical test level, and fabrication test level. Compare to conventional statistical modeling methods, neural based three-step chip speed prediction model showed superior prediction results. Developed ET based prediction model can be used to package decision system and fabrication test based prediction model can be used to optimize fabrication parameter for speed improvement.

Acknowledgments

The author would like to thank the Catholic University of Korea for support of this research.

References

1. Kim, T.: Intelligent Yield and Speed Prediction Models for High-Speed Microprocessors, Proc. IEEE Electronic Components & Technology Conf., San Diego (2002)
2. Barnett, T., Singh, A., Nelson, V.: Extending Integrated-Circuit Yield-Models to Estimate Early-Life Reliability, Trans. IEEE Reliability, **52** (2003) 296-300
3. Melzner, H.: Statistical Modeling And Analysis of Wafer Test Fail Counts. Proc. IEEE/SEMI Conf. Advanced Semiconductor Manufacturing (2002) 266-271
4. May, G.: Manufacturing ICs the Neural Way. IEEE Spectrum, **7** (1994) 233-244

Using ANNs to Model Hot Extrusion Manufacturing Process

Kesheng Wang[1], Per Alvestad[1], Yi Wang[2], Qingfeng Yuan[3], Minglun Fang[3], and Lingiang Sun[3]

[1] Knowledge Discovery Lab., Department of Production and Quality Engineering, NTNU, N-7491 Norway
[2] Cardiff School of Engineering, Cardiff University, Queen's Buildings PO Box 925, Cardiff, CF24 0YF, UK
[3] Knowledge Discovery and Management Lab., CIMS & ROBOT Center, Shanghai University, Shanghai 200436, China

Abstract. In metal forming processes, automatic selection of forming tools is heavily depended on the estimation of forming forces. Due to complex relationships between processes parameters like die angle, co-efficient of friction, velocity of dies, and temperature of billet for forming products with sound quality and forming forces related, there is a need to develop approximate models to estimate the forming forces without complex mathematical models or time--consuming simulation techniques. In this paper, an Artificial Neural Networks (ANNs) model has been developed for rapid predication of the forming forces based on process parameters. The results obtained are found to correlate well with the finite element simulation data in case of hot extrusion.

1 Introduction

Metal forming is becoming more and more attractive in the industry today. An example of this kind of metal forming is hot extrusion. By applying a constant pressure to the work piece, you push it through a preheated die. This die is formed with an angel so that the work piece is converted.

In hot extrusion processes, automatic selection of forming tools is heavily depended on the estimation of forming forces. Due to complex relationships between processes parameters like die angle, co-efficient of friction, velocity of dies, and temperature of billet for forming products with sound quality and forming forces related, there is a need to develop approximate models to estimate the forming forces without complex mathematical models or time-consuming simulation techniques.

Artificial Neural Networks (ANN) are simulates physiological features of human brain and have been used for non-linear mapping by numerical approach. Many types of problems in engineering, such as modeling, classification, predication, clustering, mapping, pattern recognition and novelty detection can be perfectly and easily solved by ANNs. [6]

The ANNs behave as model free estimators, e.g., they can capture and model complex input-output relationships even without the help of mathematical models. [2] In this paper, an effort is made to utilize the function approximation capabilities of ANNs in modeling of metal forming processes, i.e. hot extrusion. The characterizations of hot extrusion process are (1). Multiple parameters which govern the process, and (2). the lack of adequate mathematical models which map these parameters. Alternative approaches of modeling the process are Finite Element Simulation (FES) or experimentation. Experimentation is expensive because of the immense number of experiments required. Finite Element Modeling (FEM) techniques also have several limitations. Through FEM is capable of giving detailed analysis of the problem at hand, the pre-processing and program execution consumes a lot of CPU time. A small change in a single process parameter requires a new simulation run to predict its effect on forging forces.

ANNs are able to give much more generalized models, which can predict a wide variety of process parameters. In this paper, a Back-propagation learning algorithm used as learning algorithm for the ANN model of hot extrusion processing. A development tool *NEUframe* [3] has been used to establish the ANN model. The results from the model are compared with data from a real simulation. The model that *NEUframe* created is usable for predicting forging loads for hot extrusion processes. We have done this without knowing the exact connection between the four parameters involved in the process. We do not know the mathematical formula for this kind of operation as well, but we manage to get good results.

In practical use, we can now use this model to predicate forging loads for hot extrusion processes. If we get an order that requires a new die angel for instance, we just put this into the ANN model and it will tell us which forging load to use. This will be both time- and moneysaving.

The rest of the paper is organized as follows. In section 2, the hot extraction and input-output parameters are described. In section 3, a brief introduction and design of Back-propagation ANN is given. A solution procedure using *NEUframe* s presented in Section 4. In section 5, we analyze and discuss the results of ANN comparing with the results of simulation. Conclusions are presented in Section 6.

2 Problem Description

Hot extrusion is a metal forming process of forcing a heated billet to flow through a shaped die opening [5]. It is used to produce long straight metal products of constant cross-section, such as bars, solid and hollow sections, and tubes, wires and strips from materials that cannot be formed by cold extrusion. Hot metal forming has become an attractive process in industry due to its ability to achieve energy and material savings, quality improvement and development of homogeneous properties throughout the component. However, the process is rather complicated, as it requires careful selection of parameters, and control and inspection to verify that the final component has the requisite mechanical properties. This, in turn, demands a precise simulation and analysis of the process.

A number of finite element simulations are performed for forward hot extrusion of pre-form for a transmission shaft with various die angles (15°, 30°, 45°, 60° and 75°) at temperatures varying from 1000°C to 1260°C under different punch velocities (168mm/s to 203 mm/s) using a FEM. This range of operating parameters is often used in industry for hot extrusion. The die is kept at a constant temperature (350°C). The geometry is axis-symmetric in nature so only one half of the part is simulated.

The forging force at 50% reduction in diameter for a few simulations at 203 mm/s die velocity and 1260°C with 0.4 friction is depicted as a forging load graph w.r.t displacement of upper die. The variation of forging force (180 data samples) with different die angle, punch velocities, coefficient of friction and temperature of billet are obtained.

3 Back-Propagation ANN

Artificial Neural Networks (ANNs) have been used extensively in design and manufacturing.[1],[6] ANN is a model that emulates a biological neural network. It consists of Processing Elements (PEs), called neurons, and connections between PEs, called links. Associated with each link is a weight. Multiple Layer Perceptron (MLP) ANN trained by Back-propagation (BP) algorithm is one of the most popular and versatile types of ANN. It can deal with nonlinear models with high accuracy. A three-layer supervised learning ANN consisting of input layer, hidden layer and output layer is used as an identifying model in this study.

The Process Elements (PEs) are organized in different ways to form the network structure. Each PE operates by taking the sum of its weighted input and passing the result through a nonlinear activation function, mathematically models as

$$y_i = f(a_i) = f_i(\sum\nolimits_{j=1}^{n} w_{ij} - s_i) .$$

The error is subsequently backward propagated through the network to adjust the weights of the connections and threshold, minimizing the sum of the mean squared error in output layer. The ANN computes the weighted connection, minimizing the total mean square error between the actual output of the ANN and the target output. The weight are adjusted in the presence of momentum by

$$\begin{aligned}(w_{ji})_{n+1} &= (w_{ji})_n + \Delta w_{ji} \\ \Delta w_{ji}(n+1) &= \eta \delta_j x_i + \alpha \Delta w_{ji}(n)\end{aligned} .$$

where η is learn rate, δ_j is an error value for node j and α is a momentum term. The momentum term is added to accelerate convergence. [4]

In this paper, ANN is used to find the relationship between forging loads and the four variables: die angel (a), coefficient of friction (f), velocity (v) and temperature (t). If the forging load is Y, we can get a function as $y = f(a,f,v,t)$. To find the function f mathematically is a very time consuming and difficult task. This is where we can see the big advantage of using ANN. By using ANN, we do not have to derive the physical relationships of there parameter at all. We can establish an ANN model using few data sample provided. So after testing the model created in ANN and verifying it accurate enough, we can implement the ANN model as a decision making tool in analyzing the manufacturing process in stead of function f.

Today's design and manufacturing problems are changing over time, and are both complex and chaotic [6]. Therefore it is very important to a tool that can help us to deal with these changes and ANN is such a kind of a tool.

4 Solution Procedure

To solve the problem described we have used Standard Back Propagation (SBP) mode in *NEUframe* to develop the ANN model. *NEUframe* is a commercial development tool for ANN [3] and the workspace are shown in figure 1.

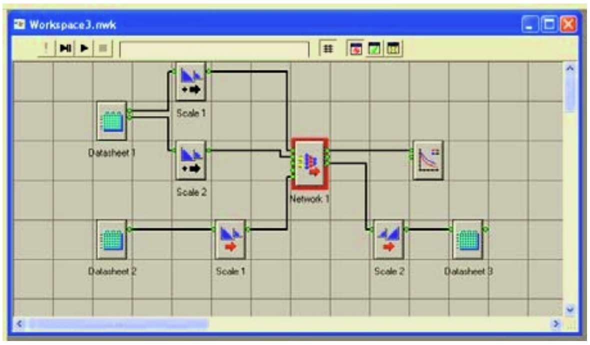
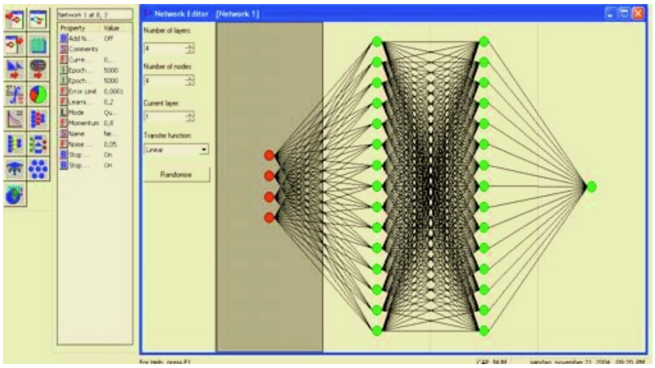


Fig. 1. Workspace created in *NEUframe.*

The four parameters: angel, coefficient of friction, velocity and temperature were put in datasheet one as input data. One also defined the target data in this datasheet. The target data was the appurtenant forging loads to the different inputs. 155 data were selected from 180 total data as training data. As in all BP networks we had one input layer consisting of 4 nodes, one output layer consisting of 1 node and two hidden layers in between with 15 nodes in each hidden layer. The first layer was set to be a purely linear layer and the rest of the layers were set to be sigmoid. Maximum epochs were considered to be 5000. The error goal was set at 0,0001 and the learning rate was taken as 0,2. It was tried several different values and combinations of these parameters mentioned here, but this setup gave the best result. We also tried only one hidden layer in the network, but it did not give as good results as this setup. Figure 2 shows the structure of the ANN in *NEUframe* window.



Property	Value
Add N...	Off
Comments	
Curre...	0,...
Epoch...	5000
Epoch...	5000
Error Limit	0,0001
Learni...	0,2
Mode	Qu...
Momentum	0,8
Name	Ne...
Noise ...	0,05
Stop ...	On
Stop ...	On

Number of layers: 4
Number of nodes: 4
Current layer: 1
Transfer function: Linear

Fig. 2. The network used in the simulation.

After the training process, the weights that *NEUframe* has calculated were fixed. Now it is time to test the model. To do this the model is presented with 25 query input values. This is 25 randomly chosen data from the total data set. They were removed from the total dataset before the training process, so that they did not contribute to the training process.

5 Results and Discussion

The results in datasheet three are the forging loads for the 25 data called query data in the previous section. What you do now, is to check the validation of the model. To do this you simply compare the data from the ANN model with the data you already had from the practical simulation. The results and the comparison are displayed in Table 1.

Table 1. Result and comparison.

No.	Simulation	ANN modeling	Errors
1	1618,09	1644,51	1,61 %
2	1614,32	1663,78	2,97 %
.	…	…	…
25	1679,73	1773,73	5,30 %

We can see that the results of the forging load provided by the ANN model are close to the practical values. There are three samples that stand out, i.e., sample No. 5 (17.82%), No. 7(29.80%) and No.18 (18,53%). Besides of these three the result is very satisfying. It can be improved if more training data could be used.

The ANN model that *NEUframe* created is usable for predicting forging loads for hot extrusion processes. We have not any knowledge about the exact mathematical relationship between the four input parameters and one output involved in the process, but still we manage to get good results through a few of data samples.

If we look at the errors they are within reasonably values, except in the cases of the three samples mentioned earlier. It is difficult to say what causes these errors. But it is important to remember that what *NEUframe* has created here are supposed to be a universal model that we can use with different values for the four parameters later. The ANN is an approximate model.

In practical use, we can now use this ANN model to calculate forging loads for hot extrusion processes. If we get an order that requires a new die angel for instance, we just plot this in to our model and it will tell us which forging load to use. This will be both time- and moneysaving.

6 Conclusions

ANN, and specially the BP algorithm is very suitable in simulation of a hot extrusion process based on practical experience. It is easy to set up a simulation in *NEUframe*, and it gives results that are very satisfying. Based on this study one can say that using ANN in general in different kinds of manufacturing operations is suitable.

Using ANN for these kinds of simulations and using ANN as a support in the manufacturing industry has proven to save a lot of time and money and one can meet the demands on flexibility and rapid changes. By doing so, company will be competitive in today's market.

Acknowledgements

The authors would like to thanks for the support from the National Natural Science Foundation of China under grant No.50375090.

References

1. Cakar, T., Cil, I.: Artificial Neural Networks for Design of Manufacturing Systems and Selection of Priority Rules. International Journal of Computer Integrated Manufacturing, **17** (2004) 195-211
2. Fausett, L.: Fundamentals of Neural Networks. Prentice Hall, Englewood Cliffs, NJ (1994)
3. NEUframe User's Guide, NCS Manufacturing Intelligence.
4. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: In Parallel Distributed Processing: Explorations in the Microstructures of Cognition. Eds.: Rumelhart, D.E. and McClelland, J.L., USA **1** (1986) 318-362
5. Semiatin, S. L.: ASM Handbook Forming and Forging. Metals Park, Ohio, USA 14 (1996)
6. Wang, K.: Applied Computational Intelligence in Intelligent Manufacturing. Advanced Knowledge Ltd. ,Australia (2005)

Application Research of Support Vector Machines in Condition Trend Prediction of Mechanical Equipment

Junyan Yang and Youyun Zhang

Theory of Lubrication and Bearing Institute,
Xi'an Jiaotong University,
Xi'an, Shaanxi 710049, China
{yangjunyan,yyzhang}@tlbi.xjtu.edu.cn

Abstract. Support vector machines (SVMs) are used for condition trend prediction of mechanical equipment. The influence of cost functions, kernel functions and parameters on prediction performance of SVMs is studied. Cost functions play a very important role on prediction performance of SVMs. Experiments show that the prediction performance of ε insensitive cost function is superior to that of least square cost function. At the same time, analysis of the experimental results shows that four kernel functions have very close prediction performance in short term prediction, and radial basis function kernel has better prediction performance than other kernels in long term prediction. In comparison with traditional Back Propagation (BP) neural network, Radial Basis Function (RBF) network and Generalized Regression Neural Network (GRNN), experiments show that SVMs, which implement the structure risk minimization principle, obtain the best prediction performance.

1 Introduction

With rapid development of scientific technology, mechanical equipment in modern industry is growing larger, more precise and more automatic. Their structures become more complex and their potential faults become more difficult to find. So, in the field of mechanical fault diagnosis, it is an urgent problem to exactly evaluate and correctly predict the running condition of the mechanical equipment [1].

Because of the noisy, non-stationary and chaotic characteristic of vibration signal of mechanical equipment, the prediction methods [2], which based on the stationary signal, such as $AR(P)$ model, $MA(q)$ model, $ARMA(p, q)$ model and multivariate regression model, can not be efficiently used to evaluate and predict the condition of mechanical equipment [3]. In recent years, neural networks have been successfully applied to evaluate and predict non-stationary time series [1], [3], [4]. The neural network prediction model has poor generalization ability because it only implements the Experience Risk Minimization (ERM) principle. Support Vector Machines (SVMs) a universal learning algorithm, which implement the Structure Risk Minimization (SRM) principle proposed by Statistical Learning Theory (STL), gradually become the hot research point in the field of artificial intelligence for its favorable generalization ability. In many practical applications, the SVMs outperform many traditional regression technologies in prediction performance [5], [6]. In this paper, the SVMs are used for condition trend prediction of mechanical equipment, and a comprehensive comparison between the neural network prediction model and SVMs is studied.

2 Support Vector Regression

SVMs, a novel learning algorithm, which come from an optimal separating hyperplane in case of linearly separable, were developed by Vapnik and his co-works. Its core idea is to map the original pattern space into the high dimensional feature space $\mathbf{Z}$ through some nonlinear mapping functions, and then construct the optimal separating hyperplane in the feature space. Thus, the nonlinear problem in low dimensional space corresponds to the linear problem in the high dimensional space. In order to generalize the results obtained for estimating the indicate functions (pattern recognition problem) to estimate real-valued functions (regression problem), ε insensitive cost function is introduced [7]. Like the pattern recognition problem, the input vectors are mapped into high dimensional feature space through nonlinear mapping function $\varphi(x)$. The linear function sets

$$f(x,\alpha) = (\omega \bullet \varphi(x)) + b \tag{1}$$

in feature space are used for estimating the regression function. To given training data set

$$(y_1, x_1), (y_2, x_2), \cdots, (y_l, x_l) \qquad x \in \mathbf{R}^n \qquad y \in \mathbf{R}$$

the corresponding constraint optimization problem is

$$\begin{gathered} \min \tfrac{1}{2}||\omega||^2 + C\tfrac{1}{l}\textstyle\sum_{i=1}^{l}(\xi_i + \xi_i^*) \\ \text{s.t.} \begin{cases} y_i - (\omega \bullet \varphi(x_i)) - b \le \varepsilon + \xi_i & i = 1, 2, \cdots, l \\ (\omega \bullet \varphi(x_i)) + b - y_i \le \varepsilon + \xi_i^* & i = 1, 2, \cdots, l \\ \xi_i, \xi_i^* \ge 0 & i = 1, 2, \cdots, l \end{cases} \end{gathered} \tag{2}$$

where the coefficient C is a penalty factor, and it implements a tradeoff between empirical risk and confidence interval. The coefficients ξ_i and ξ_i^* are slack factors.

Eq. (2) is a classical convex optimization problem. According to the Lagrangian multiplier theory, weight vector ω is equal to linear combination of training points:

$$\omega = \sum_{i=1}^{l}(\alpha_i - \alpha_i^*)\varphi(x_i) \tag{3}$$

where the coefficients α_i and α_i^* are Lagrangian multipliers. Combining Eq. (1) and Eq. (3), we can get the solution of the unknown data point x:

$$f(x) = \sum_{i=1}^{l}(\alpha_i - \alpha_i^*)(\varphi(x_i) \bullet \varphi(x)) + b \tag{4}$$

In Eq. (4), the inner product $(\varphi(x_i) \bullet \varphi(x))$ needs to be computed in feature space. In 1992, Boser et al [7] found that it is not necessary to compute the inner product explicitly in feature space. According to kernel function theory, we can use the kernel function $K(x_i, x_j)$ in input space, which satisfying the Mercer condition, to compute the inner product. So Eq. (4) can be expressed as:

$$f(x) = \sum_{i=1}^{l} (\alpha_i - \alpha_i^*) K(x, x_i) + b \tag{5}$$

The typical examples of kernel function are the polynomial kernel, the radial basis function kernel, the sigmoid kernel and the linear kernel.

3 Experiments

We continually monitored and recorded the vibration signal of a generator machine sets about 141 hours, and extracted the peak-peak value from the vibration signal every one hour. Thus, these history records about peak-peak value formed a time series, as shown in Fig. 1. In this time series, the first 117 points are selected as training sample and others as testing sample.

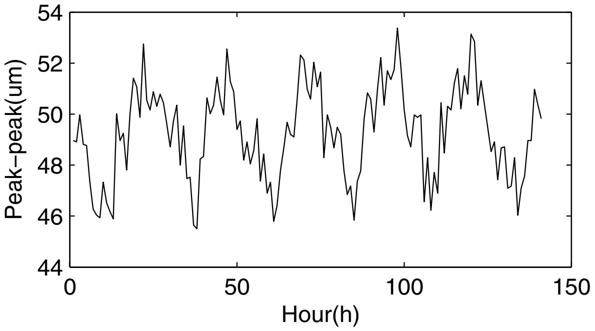


Fig. 1. History records of the machine sets about peak-peak value

To given time series $x_0, x_1, \cdots, x_{n-1}$, condition trend prediction refers to estimate the future value at time t_{n+k} with the history observation value of time series, where k is prediction step. The problem of predicting the x_n based on m previous value $x_{n-1}, x_{n-2}, \cdots, x_{n-m}$ can be expressed as:

$$x_n = f(x_{n-1}, x_{n-2}, \cdots, x_{n-m}) \tag{6}$$

where m is called prediction order.

The prediction performance is evaluated by using the Normalized Mean Square Error (NMSE).

$$\begin{aligned} \text{NMSE} &= 1/(\delta^2 n) \textstyle\sum_{i=1}^{n} (x_i - x_i')^2 \\ \delta^2 &= 1/(n-1) \textstyle\sum_{i=1}^{n} (x_i - \overline{x})^2 \end{aligned} \tag{7}$$

4 Analysis of Experimental Results

4.1 Results of Different Cost Functions

In this section, two cost functions, ε insensitive cost function and least square cost function, are used to evaluate the influence of different cost functions on prediction performance of SVMs. In 1999, J.A.K. Suykens [8] proposed a modified version of SVMs, Least Square SVMs (LS-SVM), where a least square cost function is used as the cost function of SVMs [9]. It involves equality constraints instead of inequality constraints, and its solution follows from a linear Karush-Kuhn-Tucker system instead of a quadratic programming problem. However, sparseness is lost in the LS-SVM case. The corresponding constraint optimization problem is given by

$$\begin{array}{c} \min \frac{1}{2}||\omega||^2 + \frac{1}{2}C\sum_{i=1}^{l}(\xi_i^2) \\ \text{s.t. } y_i = \omega \bullet \varphi(x_i) + b + \xi_i \qquad i = 1, 2, \cdots, l \end{array} \tag{8}$$

Eq. (8)'s solution can be obtained by solving the following linear equation:

$$\left[\begin{array}{c|c} 0 & y^T \\ \hline y & \Omega + C^{-1}I \end{array}\right] \left[\begin{array}{c} b \\ \hline \alpha \end{array}\right] = \left[\begin{array}{c} 0 \\ \hline 1_v \end{array}\right] \tag{9}$$

where $y = [y_1, y_2, \cdots, y_l]$, $1_v = [1; 1; \cdots; 1]$, $\alpha = [\alpha_1, \alpha_2, \cdots, \alpha_l]$, and $\Omega_{ij} = K(x_i, x_j)$ for $i, j = 1, 2, \cdots, l$.

Table 1. The result of two cost functions

Cost function	C	γ	m	NMSE
ε insensitive	1	0.125	17	0.238091
Least square	4	1	7	0.268277

Table 1 gives the result of two cost functions in 24-step prediction, where the RBF kernel is used as the kernel functions, and ε is fixed at 0.01. The experiments show that the NMSE of ε insensitive cost function on test set is less than that of least square cost function about 0.03. This indicates that the prediction performance of ε insensitive cost function is superior to that of least square cost function. So, in this paper, we adopt the ε insensitive cost function to research the SVMs prediction problem.

4.2 Prediction Performance of Various Kernel Functions in SVMs

In this section, four kernel functions are used to analyze the influence of kernel function type on prediction performance of SVMs. Four kernel functions are linear kernel $x \bullet x_i$, polynomial kernel $[(x \bullet x_i) + 1]^d$, radial basis function kernel $\exp(-||x - x_i||^2)$ and sigmoid kernel $\tanh[v(x \bullet x_i) + c]$. The grid search method based on K-fold cross validation [10] is used to select parameters of SVMs. The optimal parameters of SVMs are listed in Table 2, and ε is fixed at 0.01.

Table 2. The optimal parameters of four kernel function. 1S denotes the 1-step prediction error on test set, and 24S denotes the 24-step prediction error on test set

Kernel	m		C		others	
function	1S	24S	1S	24S	1S	24S
RBF	14	17	8	1	$\gamma = 1/128$	$\gamma = 0.125$
Sigmoid	14	15	4	4096	$v = 0.03$ $c = 2$	$v = 0.005$ $c = 0.5$
Poly	14	16	1	2	$d = 3$	$d = 3$
Linear	14	15	0.125	8192		

Fig. 2 gives the NMSE of SVMs on test set at different kernel functions. The figure shows that the NMSE of four kernel functions is very close in 1-step prediction. This indicates that kernel function has little influence to SVMs prediction performance in short term prediction. In contrast to 1-step prediction, Fig. 2 shows that kernel functions exert an important influence on SVMs prediction performance in 24-step prediction, and RBF kernel obtains the best prediction performance. As stated above, the RBF kernel outperforms other kernels in prediction performance. So, in present paper, we adopt the RBF kernel to research the prediction problem of SVMs.

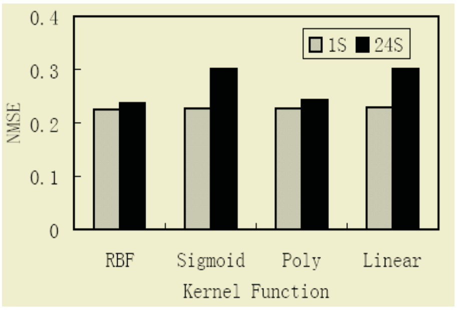


Fig. 2. Results of four kernel functions

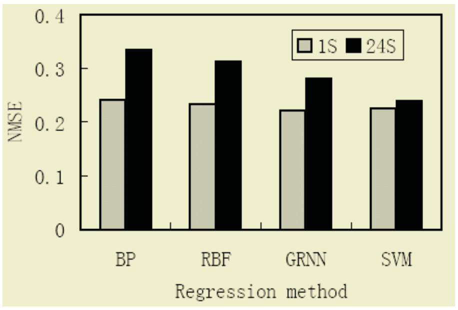
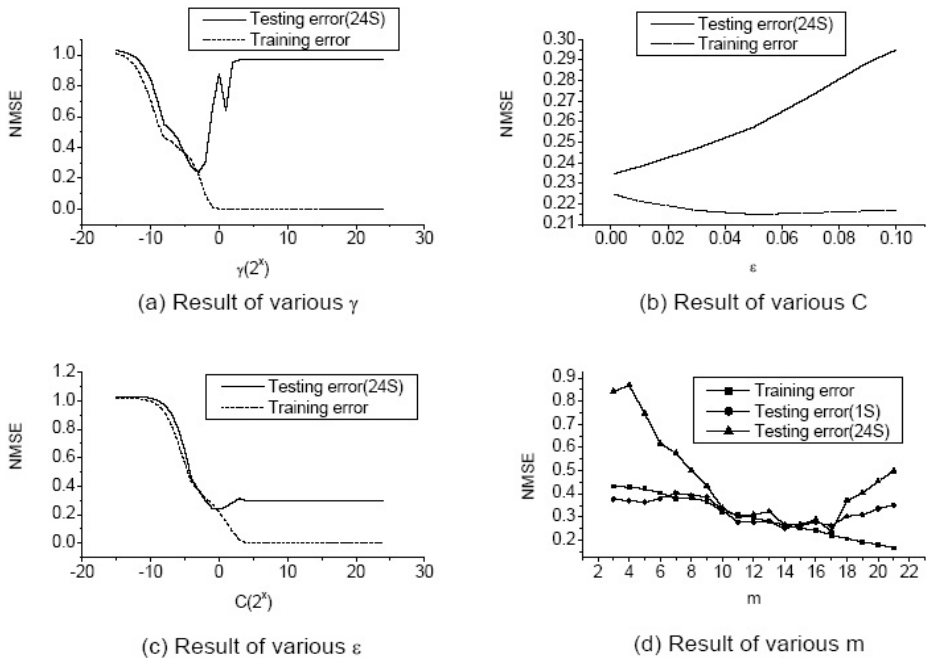


Fig. 3. Results of SVMs and three Neural network models

4.3 Sensitivity of SVMs to Parameters

In this section, we discuss the influence of parameters C, γ, ε and m on prediction performance of SVMs. Fig. 4(a) gives the results of SVMs at $\gamma = [2^{-15}, 2^{-14}, \cdots, 2^{15}]$, in which C, ε and m are, respectively, fixed at 1, 0.01 and 17. This figure shows that the NMSE on training set decreases with γ. On the other hand, the NMSE of 24-step prediction on test set decreases initially but subsequently increases as γ increases. This indicates that too large a value γ $(2^{-1} - 2^{15})$ causes SVMs to over-fitting the training data. An appropriate value for γ would be between 2^{-5} and 2^{-1}. It can be said that γ plays a very important role on generalization performance of SVMs.

Fig. 4(b) gives the influence of parameter C at $C = [2^{-15}, 2^{-14}, \cdots, 2^{14}, 2^{15}]$ on prediction performance of SVMs, where γ, ε and m are fixed at 0.125, 0.01 and 17. It



(a) Result of various γ (b) Result of various C

(c) Result of various ε (d) Result of various m

Fig. 4. Results of various parameters γ, C, ε and m

can be observed that the NMSE on training set decreases monotonically as C increases. In contrast, the NMSE on test set decreases as C increases from 2^{-15} to 2^0, and then increases as C increases from 2^0 to 2^5, and subsequently maintains a constant value at 0.308129 as C increases beyond 2^5. This indicates that with the increase of C from 1 to 2^{15}, the prediction performance of SVMs keeps steady basically, and compared to parameter γ, C do not emerge obviously over-fitting phenomenon.

Fig. 4(c) gives the result of various parameters ε ranging between 0 and 0.1, where C, γ and m are fixed at 1, 0.125 and 17. This figure shows that the NMSE on training set is very stable, and relatively unaffected by changes in ε. On the other hand, the NMSE on test set increases with ε, but the change scope of NMSE is only 0.05. We can see that the local change of ε has little influence on prediction performance of SVMs, but according to the SVMs theory, the number of support vector decreases as ε increases thus resulting in a speedup of testing speed.

Fig. 4(d) gives the influence of prediction order m on prediction ability of SVMs, in which C, γ and ε are fixed at 1, 0.125 and 0.01, and m range between 3 and 21. This figure shows that the NMSE on training set decreases as m increases, and the NMSE of 1-step prediction keeps steady basically. The NMSE of 24-step prediction on test set decreases initially but subsequently increases as m increases. As stated above, to short term prediction, SVMs are insensitive to prediction order m. On the other hand, because a small m includes less prediction information than that of a large m, it shows poor prediction ability in long term prediction. When m is too large, it includes many noises, and lead to deterioration in the prediction performance of SVMs. In that case, an appropriate m would be between 11 and 17.

4.4 Comparison Study Between SVMs and Neural Network Model

In this investigation, three-layer BP neural network, RBF neural network and GRNN neural network are used as benchmark algorithm to compare with SVMs. According to Sect. 4.1 and Sect. 4.2, the ε insensitive cost function and the RBF kernel are respectively used as cost function and kernel function of SVMs. The optimal parameters C, γ, ε and m are fixed at 1, 0.125, 0.01 and 17 because they provide the best possible result based on the grid search method. To three-layer BP neural network, there are 7 nodes in the input layer which is equal to the prediction order m, and 15 nodes in hidden layer and 1 node in output layer. The Bayesian regularity algorithm is adopted by three-layer BP neural network. To RBF and GRNN neural network, the spread of the radial basis function are respectively fixed at 4.5 and 0.35, and the prediction order m equal to 7. Fig. 3 gives the comparison among SVMs and three neural networks model. This figure shows that the four methods have very close NMSE on 1-step prediction, and SVMs obtain the least NMSE on 24-step prediction. This indicates that the prediction performance of SVMs outperform three neural networks model in long term prediction. Therefore, it can be concluded that SVMs provide a promising method in condition trend prediction of mechanical equipment.

5 Conclusions

This study used SVMs to predict the condition of mechanical equipment. In this paper, the effect of various cost functions and kernel functions was investigated. Experiments show that the ε insensitive cost function obtains better NMSE than least square cost function. Kernel function has little influence on prediction ability of SVMs in short term prediction, but in long term prediction, RBF kernel obtains the best prediction ability based on NMSE.

In addition, the influence of parameters C, γ, ε and m on prediction performance of SVMs was discussed in this study. Experimental results show that improper selection of parameters C and γ can cause over-fitting problem of SVMs and the prediction performance of SVMs is insensitive to the local change of ε. Prediction order m has great influence on prediction performance of SVMs. A small m includes less prediction information than that of a large m, so it shows poor prediction performance in long term prediction. When m is too large, it includes too many noises, and lead to deterioration in the prediction performance of SVMs. An appropriate m would be between 11 and 17.

In the end, this study compared SVMs with neural network prediction model. The experimental results show that SVMs and neural network prediction model have very close prediction performance in short term prediction, but in long term prediction, SVMs obtain better prediction performance than neural network prediction model for their favorable generalization performance.

Acknowledgment

Thanks are due to the anonymous referees for valuable suggestions. This work is supported by National Natural Science Foundation of China under Grant No. 50335030.

References

1. Zhang, X.N.: Research on the Condition Monitoring and Forecasting for Large Rotating Machine (in Chinese). Ph.D thesis of Xi'an Jiaotong University, Xi'an China (1998)
2. Yang, S.Z., Wu, Y.: Time Series Analysis in Engineer Application (in Chinese). Huazhong University of Science and Technology Press, Wuhan China (1999)
3. Shu, Y.P.: Forecasting Theory Based on Neural Network and Its Application (in Chinese). Ph.D thesis of Xi'an Jiaotong University, Xi'an China(1994)
4. Kin, T.Y., Oh, K.J., Kim, C., Do, J.D.: Artificial Neural Networks for Non-Stationary Time Series. Neurocomputing, **61** (2004) 439-447
5. Tay, F.E.H., Cao, L.J.: Application of Support Vector Machines in Financial Time Series Forecasting. Omega, **29** (2001) 309-317
6. Thissen, U., Brakel, R.V., Weijer, A.P.D., Melssen, W.J., Buydens, L.M.C.: Using Support Vector Machines for Time Series Prediction. Chemometrics and Intelligent Laboratory Systems, **69** (2003) 35-49
7. Vapnik, V.N.: Statistical Learning Theory (in Chinese). Xu J.H., Zhang, X.G. Publishing House of Electronics Industry, Beijing (2004)
8. Suykens, J.A.K., Brabanter, J.D., Lukas, L., Vandewalle, J.: Weighted Least Squares Support Vector Machines: Robustness and Sparse Approximation. Neurocomputing, **48** (2002) 85-105
9. Gestell, T.V., Suykens, J.A.K., Baestaens, D.E., et al.: Financial Time Series Prediction Using Least Squares Support Vector Machines within the Evidence Framework. IEEE Transactions on Neural Networks, **12** (2001) 809-821
10. Hsu, C.W., Chang, C.C., Lin, C.J.: A Practical Guide to Support Vector Classification . Technical Report, Department of Comptuer Science & Information Engineering, National Taiwan University, Taiwan China

Comparative Study on Engine Torque Modelling Using Different Neural Networks

Ding-Li Yu[1] and Michael Beham[2]

[1] Control Systems Research Group, School of Engineering
Liverpool John Moores University, UK
d.yu@livjm.ac.uk
[2] BMW AG E30, Munich, Germany

Abstract. In this paper, modelling the torque of a variable valve timing (VVT) spark ignition (SI) engine is studied using multilayer Perceptron (MLP), radial basis function (RBF) and local linear model tree (LOLIMOT) networks. Real data collection, model order selection, model test are discussed. The obtained model is finally tested in real time on-line mode by an engine test bench. The model output is compared with the process output and compared among different network models. Selected model performs well and can be used in the model-based engine control.

1 Introduction

To design and implement model-based fault detection and control, and also to simulate these systems during the design and implementation, accurate engine models for each sub-system of the engine is need to be developed. Up to now there are no spark ignition engine models based on first principle mathematical equations have been reported in the literature, which have a tolerable modelling error. Only partial engine areas can be modelled by a physical model in non real time operation mode, due to simplification in implementation.

Neural networks (NN) have been used to model complex dynamics systems including engines [1]. In this paper, three types of NNs, MLP, RBF and LOLIMOT are used to model engine torque of a VVT SI engine from real data, and the models are compared. Obtained model is evaluated in real time by an engine test bench to predict torque in parallel mode and the results are satisfactory.

2 Engine Description

For the investigation an internal combustion reciprocating piston engine according to the Otto process was used. Characteristically for this system is the external carburetion. The actuators, valve lift (VL), throttle position (TP), camshaft intake position (CIP), and camshaft exhaust position (CEP) control the air mass flow. The air to fuel ratio is controlled by injection timing (IT). This ratio influences the exhaust gas, CO, CO_2, HC and NO_X. The schematic of this engine is shown in Fig.1 with input-output variables list as follows, *Inputs:*

Engine Speed	(ES)	Camshaft Intake Position	(CIP)
Camshaft Exhaust Position	(CEP)	Valve Lift	(VL)
Ignition Angle	(IA)	Throttle Position	(TP)
Injection Timing	(IT)	Engine Temperature	(TE)
Differential Pressure	(DPS)	Intake Air Temperature	(IAT)

Outputs:

Torque	(T)

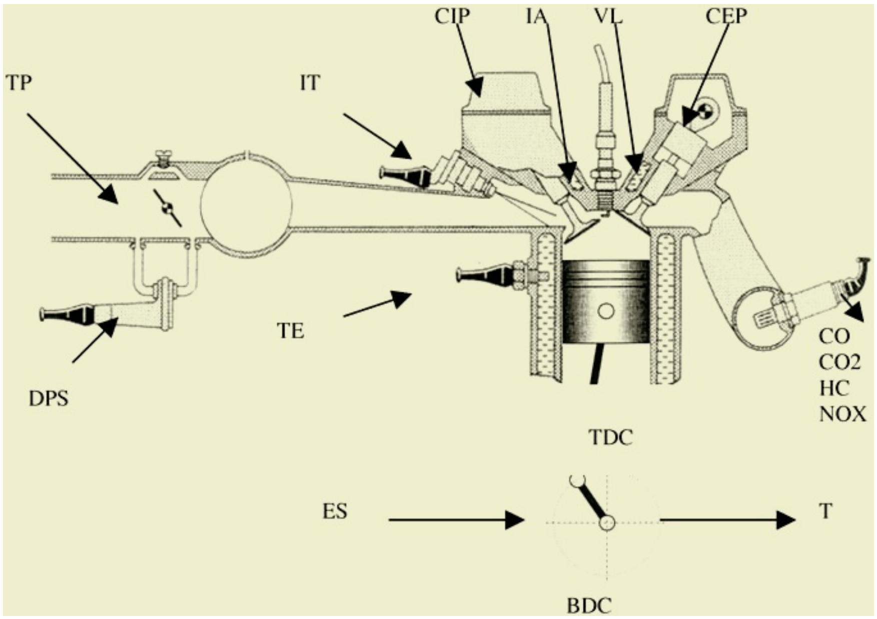


Fig. 1. Spark ignition engine.

The throttle in Figure 3 is used in normal operation condition for tank venting. In fail/save mode the air mass flow is controlled by the throttle position, like a conventional Otto engine. The air/fuel mixture is compressed. At an assigned ignition angle starts the ignition using external energy addition. Changes in load are mainly controlled by the valve opening time in respect of the CIP and the CEP. The torque is produced by rising pressure in the combustion chamber caused by the heat of the combustion. The exhaust gas composition is a function of the air to fuel ratio, determined by the injection timing, and the geometrical conditions of the combustion chamber.

3 Brief Review of the Neural Networks Used

MLP Network. To model the engine torque, a single output MLP network with one hidden layer is used. The non-linear mapping realized by this network can be described by the following equation.

$$\hat{y}(k) = o^T(k) * w^{[2]} + b^{[2]} . \tag{1}$$

where $w^{[2]} \in \Re^{nh}$ and $b^{[2]}$ is the weight vector and the bias between the hidden layer and the output layer, $o \in \Re^{nh}$ is the hidden layer output vector, the element of which is from the non-linear transformation,

$$o_i(k) = \frac{1}{1+e^{\xi_i(k)}} \ , \ i = 1, \cdots, nh \, . \tag{2}$$

where $\xi_i(k)$ is given by,

$$\xi(k) = W^{[1]} x(k) + b^{[1]} \, . \tag{3}$$

where $W^{[1]} \in \Re^{nh\,x\,n}$ and $b^{[1]} \in \Re^{nh}$ are input weight matrix and the input bias vector respectively, n_h is the number of the hidden layer nodes and n is the number of the network inputs, $x \in \Re^n$ is the network input vector and $\hat{y}$ the network output. The RBF network will be trained using the second order Newton method, Levenberg Marquardt algorithm.

RBF Network. The Gaussian RBF network is used in this research with the following equations.

$$\hat{y}(k) = w^T z(k) \, . \tag{4}$$

$$z_i(k) = \exp\left(-\frac{1}{\sigma_i^2} \left\| x(k) - c_i \right\|^2 \right), \ i = 1, \cdots, n_h \ . \tag{5}$$

where $z \in \Re^{nh}$ is the hidden layer output vector, $c_i \in \Re^n$ denotes the ith centre vector, $\sigma_i \in \Re$ denotes the ith width of the Gaussian function. Centres and widths are chosen using the K-means clustering method and the p-nearest centres method, while the weights are trained using the recursive orthogonal least squares (ROLS) algorithm [2].

LOLIMOT Network. The LOLIMOT network [3] is a RBF type network but based on a number of local linear models $z_i, i = 1, \cdots, M$, the weighted sum of which forms the output.

$$\hat{y} = z\phi \, . \tag{6}$$

where $z, \phi \in \Re^M$ are hidden layer output vector with each neuron realizes a local linear model and an associated validity function that determines the region of validity of the local linear model. The validity functions are chosen as normalized Gaussian.

$$\varphi_i = \frac{\mu_i}{\sum_{j=1}^{M} \mu_j}, \ \ \mu_i = \exp(-\frac{\left\| x - c_i \right\|^2}{\sigma_i^2}) \ . \tag{7}$$

The output of the local linear model is

$$z = Wx \, . \tag{8}$$

where $W \in \Re^{M \times n}$ denotes the input weight matrix and $x \in \Re^n$ is the input vector. In the design, M, c_i and σ are chosen according to error threshold using the so-called model tree algorithm [3]. The model tree algorithm splits the input space into linear partitions. The weights are trained the Least squares algorithm. The major advantage of LOLIMOT network is its low computational complexity.

4 Model Training

4.1 Model Order Selection

According to the knowledge and previous modelling experience of the engine, the maximal output order was determined as $n_y = 3$ and maximum input order was $n_u = 1$. The input time delay was chosen for the input variables as 0. Then, the following model structures were chosen:

$$\hat{y}(k) = f[y(k-1), u(k-1-d_u)]\,. \tag{9}$$

$$\hat{y}(k) = f[y(k-1), y(k-2), u(k-1-d_u), u(k-2-d_u)]\,. \tag{10}$$

$$\hat{y}(k) = f[y(k-1), y(k-2), y(k-3), u(k-1-d_u), u(k-2-d_u)]\,. \tag{11}$$

for first order, second order and third order model input respectively. For model validation, all process outputs will be replaced by the model output and therefore forms so called parallel models.

4.2 Real Data Collection

To collect data when the engine works in different conditions involving signals in different frequencies and different operating points, two different driving cycles, FZ3 and FZ4, were designed and produced by driving in different gears and velocity conditions. Fig.2 and 3 show the two driving cycles for the gear and velocity changes.

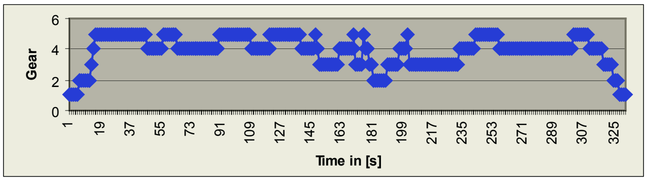


Fig. 2. Gear changes in drive cycle FZ3.

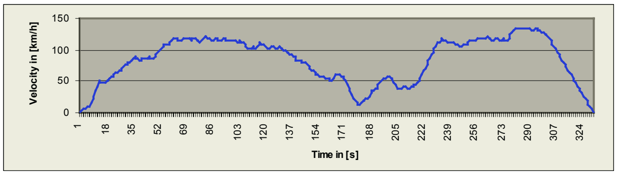


Fig. 3. Velocity changes in drive cycle FZ3.

4.3 Modelling Results

After training for different network models with different orders as shown in (9)-(11), the results in the mean squares errors are shown in table 1 for comparison.

Table 1. Modelling results.

		LOLIMOT			RBF				MLP			
Output		Train		Test	Train		Test		Train			
y	order	nh	MSE	MSE	nh	MSE	nh	MSE	nh	MSE	nh	MSE
T	1	18	8.4767	19.4738	16	4.6687	16	7.8287	15	3.4844	13	8.1194
T	2	24	7.4746	14.9260	18	2.3499	18	5.3207	15	1.3424	11	5.7145
T	3	21	3.5323	6.4095	20	2.2986	20	5.4735	15	1.2308	14	7.0230

It can be seen that the second order RBF network with 18 hidden layer nodes gives the best modelling results for test data. This is also shown in Fig.4

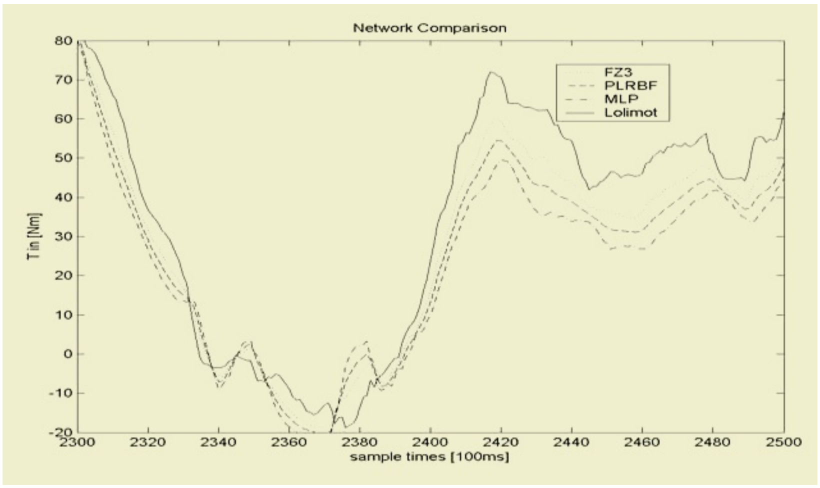


Fig. 4. Comparison of modelling with different networks.

5 Real-Time Model Evaluation

The selected best model from the trained different model structures was validated in real time on the test bench. The engine input still use the test cycles FZ3 for the reason that these signals present process dynamic behaviour at different frequencies and covers the whole operating region. The network models were tested in parallel mode (independent from the engine output). This guarantees the model accuracy for multi-step-ahead prediction in the later use of the model based-predictive control scheme. The neural network output is shown and compared with the process output in Fig.5. The corresponding residuals $e = |y - \hat{y}|$ are also displayed in the figure.

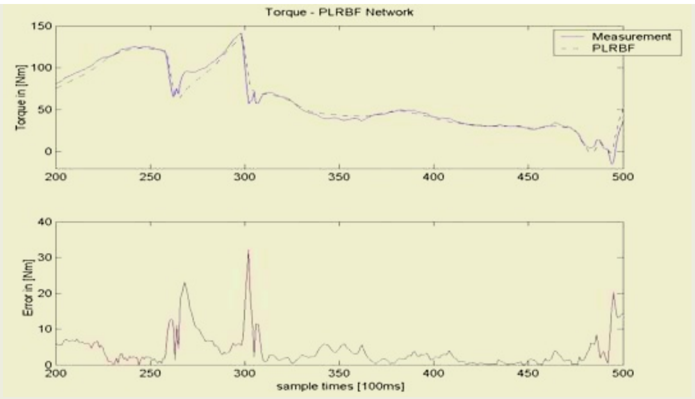


Fig. 5. Process and parallel model output for torque.

6 Conclusions

Three types of neural networks, MLP, RBF and LOLIMOT have been used to model engine torque of variable valve timing spark ignition engines. Models with different orders have been tried and real data modelling results show that an RBF model of second order gives a best model. The obtained model is also evaluated in real time on an engine test bench to independently predict engine torque. The on-line result is shown satisfactorily.

References

1. Holzmann, H., Halfmann, Ch., Isermann, R.: Neuro-Fuzzy Modelling of Automotive SI-Engine Characteristics, IFAC Workshop on Advances in Automotive Control. Mohican State Park, Ohio, USA, February 27 – March **1** (1998)
2. Yu, D.L., Gomm, J.B., Williams, D.: A Recursive Orthogonal Least Squares Algorithm for Training RBF Networks, Neural Processing Letters, **5** (1997) 167-176
3. Nelles, O.: Nonlinear System Identification, Springer-Verlag, Berlin, Heidelberg, New York (2001)

A Hybrid Intelligent Soft-Sensor Model for Dynamic Particle Size Estimation in Grinding Circuits*

Ming Tie, Heng Yue, and Tianyou Chai

Research Centre of Automation, Northeastern University,
Shenyang, Liaoning 110004, China
tiemingd@hotmail.com

Abstract. The purpose of this paper is to develop an on-line soft-sensor for dynamic estimation of the particle size distribution of hydrocyclones overflow in consecutive grinding process. The hybrid model based soft-sensor is based on the following model structures: 1. a neural net-based dynamic model of state space description for hydrocyclone with a neural net-based model for classifier and a population balance model for ball mill and sump, 2. an ANFIS-based model mainly for abnormal operating conditions, 3. a fuzzy logic coordinator for the final predictive result according to output values of aforementioned models. The fact that the soft-sensor performs well in particle size estimation demonstrates that the proposed hybrid intelligent soft-sensor model is effective for dynamic estimation of particle size distribution.

1 Introduction

In the mineral processing industry, it's an extremely important goal to achieve an optimal particle size distribution in grinding circuit's products. But the particle size analyzer has long measurement period, high price, expensive maintenance and frequent calibration. Infrequent sampling of the controlled process variables can present potential operability problems. As such, it's imperative for researchers to develop online particle size estimators which are both accurate and frequent enough.

Soft-sensors, which supply the estimation with a model based on technological knowledge and appropriate modeling approaches, have been employed in mineral processing industry since 1980s [4]. Chilean researchers developed particle size estimators for one of Codelco's concentrators by a static empirical model with linear dynamic compensation [1]. Canada researchers developed neural net-based particle size soft-sensors for dynamic estimation [2]. The above techniques are effective, but they are not appropriate for consecutive grinding process with a classifier, which is complex and is common in mineral industry. Zhang developed a neural net-based soft-sensor model for consecutive grinding process [10], but it was not appropriate for dynamic estimation and overlooked the products from 2nd mill.

This paper proposes a hybrid intelligent soft-sensor model based on population balance models and hybrid intelligent modeling approaches.

* This paper is supported by Chinese National Hi-Tech Development Program (2004AA412030).

2 Description of the Consecutive Grinding Process

The consecutive grinding process(Fig.1) we studied is composed of two closed grinding circuits. M_{75}, percent of solids whose diameter are under 75μm of solids in overflow products, is the ultimate controlled variable of the grinding process.

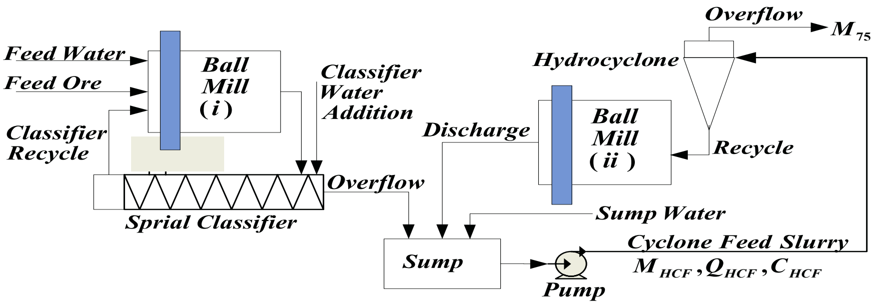


Fig. 1. Schematic diagram of consecutive grinding process.

Q_{HCF} is the hydrocyclone volumetric feed rate of slurry, C_{HCF} is the concentration of hydrocyclone feed, M_{HCF} is the percent of solids under 75μm in hydrocyclone feed.

3 The Hybrid Intelligent Particle Size Soft-Sensor Model

3.1 Hydrocyclone Model

According to the classify theory of the hydrocyclone [6], M_{75} can be determined by Q_{HCF}, C_{HCF} and M_{HCF}. Since $\dot{Q}_{HCF}$ is a constant because of the control strategy of the pump, a dynamic equation for M_{75} can be described as

$$\dot{M}_{75} = f_1(Q_{HCF}, C_{HCF}, M_{HCF}, \dot{C}_{HCF}, \dot{M}_{HCF}) \cdot \tag{1}$$

where f_1 is assumed to be a neural networks function. To get M_{HCF}, $\dot{C}_{HCF}$ and $\dot{M}_{HCF}$, the sump model should be developed.

3.2 Population Balance Model

Since the sump behaves as a single perfect mixer [6], C_{HCF} and M_{HCF} can represent the solids concentration of slurry and the particle size distribution in sump respectively. Thus its behavior can be described as the following population balance equations:

$$\dot{V}_S = Q_W + Q_{SPO} + Q_2 - Q_{HCF} \cdot \tag{2}$$

$$\frac{dV_S C_{HCF}}{dt} = Q_{SPO} C_{SPO} + Q_2 C_2 - Q_{HCF} C_{HCF} \cdot \tag{3}$$

$$\frac{dV_S C_{HCF} M_{HCF}}{dt} = Q_{SPO} C_{SPO} M_{SPO} + Q_2 C_2 M_2 - Q_{HCF} C_{HCF} M_{HCF} \cdot \tag{4}$$

where V_S represents the volume of slurry in the sump, Q_W is the volumetric feed rate of water added to the sump, Q_2 is the slurry discharge rate from 2nd mill, C_2 is the slurry concentration in 2nd mill, M_2 is the percent of solids under 75μm in discharge from 2nd mill, Q_{SPO} is the feed rate of the spiral classifier's overflow, C_{SPO} is the solids concentration of the spiral classifier's overflow, M_{SPO} is the percent of solids under 75μm in the spiral classifier's overflow.

Since V_2 is a constant for overflow ball mills[6], C_2 can be get from

$$V_2\dot{C}_2 = Q_{HCF}C_{HCF} - Q_{HCO}C_{HCO} - Q_2C_2. \tag{5}$$

where C_{HCU} and C_{HCO} are the solids concentration of hydrocyclone overflow and underflow respectively, Q_{HCO} is the flow rate of hydrocyclone overflow. Q_2 can be get from the difference of Q_{HCF} and Q_{HCO}.

M_2 can be get from the following ball mill population balance equation:

$$V_2\frac{dC_2(t)M_2(t)}{dt} = Q_2(t)C_{HCU}(t)M_{75}(t-\tau) - Q_2(t)C_2(t)M_2(t) + s_1(t)V_2C_2(t)M_2(t) \tag{6}$$

where τ is a constant for a given grinding circuit. s_1 is the size selection function, which is proportional to $P/(C_2V_2)$ [5]. P represents the power draft of the mill for a given material, which can be derived from one of the Bond equations [7].

Q_W, Q_{SPO}, C_{SPO}, Q_{HCF}, C_{HCF}, Q_{HCO}, C_{HCO} are on-line measurable variables. M_{SPO} comes from the following classifier model.

3.3 Classifier Model

According to the classification theory, the correlative variables of M_{SPO} are Q_{SPO}, C_{SPO} and Q_{SPR}, the classifier recycle slurry flow rate. Since Q_{SPR} has approximate nonlinear relationship with N_{SP} [8], the classifier shaft power, the classifier model is described as

$$M_{SPO} = f_2(N_{SP}, Q_{SPO}, C_{SPO}). \tag{7}$$

where f_2 is assumed to be a neural networks function.

3.4 Neural Networks Modeling Approach for f_1 and f_2

The radial basis function (RBF) networks are selected for f_1 and f_2. The num of hidden layer nodes are determined with the rival penalized competitive learning method [9].

In this paper, the mapping of the two RBF networks is described as

$$\begin{gathered} f_i(X_i) = w_i^0 + \sum_{j=1}^{M_i} w_i^j \exp\left[-\frac{M_i}{(d_i)^2}\|X_i - c_i^j\|^2\right] \qquad i = 1,2 \\ X_1 = [Q_{HCF}, C_{HCF}, M_{HCF}, \dot{C}_{HCF}, \dot{M}_{HCF}]^T, \quad X_2 = [Q_{SPO}, C_{SPO}, N_{SP}]^T \end{gathered} \tag{8}$$

where M_i is the num of nodes of hidden layer, $\|.\|$ denotes the Euclidean norm, d_i is the largest distance of centers, c_i^j are called RBF center, w_i^j are the tap weights. The standard LMS algorithm is selected as the learning algorithm.

3.5 ANFIS Model for Abnormal Operating Conditions

The ANFIS model is used mainly for abnormal operation, such as the malfunction of some measurement meters, or excessive recycle slurry of the classifier etc.

The input variables of the ANFIS model are $F_{ORE}(t-\tau_1)$, the fresh ore feed rate, $F_{WM}(t-\tau_1)$, the water feed rate at 1st mill entrance, $Q_{WSP}(t-\tau_2)$, the classifier water addition rate, $Q_W(t-\tau_3)$ and Q_{HCF}. The output variable is M_{75}. For the grinding process in this paper, τ_1, τ_2 and τ_3 can be presented with 5, 3, 1 respectively. Fuzzy variables u_1~u_5 represent the 5 input variables respectively. There are 3 fuzzy sets for u_i, defined as "Big", "Middle", "Small". The structure of the ANFIS model is as shown in Fig.2, And its learning algorithm is Back propagation algorithm.

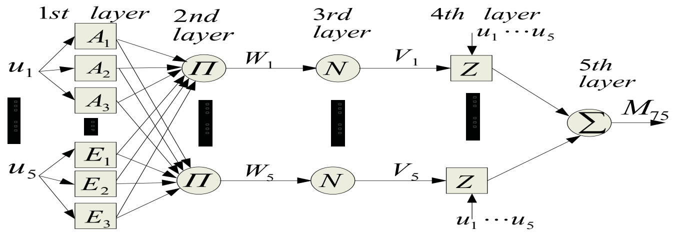


Fig. 2. Structure of the ANFIS Model.

3.6 Fuzzy Logic Coordinator

To combine the results from the above ANFIS model and the hydrocyclone model, a coordinator based on fuzzy inference is designed. N_{SP}, C_{SPO}, and ME_ER(a logic variable to represent malfunction in measurement meters) are inputted to determine the weight values for the results from the ANFIS model and the hydrocyclone model. F is assumed to be a fuzzy set which represents the result from hydrocyclone model is suitable. The membership function for N_{SP} and C_{SPO} in F are described as

$$\mu_i = \begin{cases} \dfrac{L_3^i - x_i}{L_3^i - L_2^i} & x_i \in [L_2^i, L_3^i] \\ 1 & x_i \in [L_1^i, L_2^i] \\ \dfrac{x_i}{L_1^i - x_i} & x_i \in [0, L_1^i] \\ 0 & others \end{cases} \qquad (i = 1,2) \tag{9}$$

where $x_1 = N_{SP}$, $x_2 = C_{SPO}$, μ_1 and μ_2 are values of their membership.

From the first-principle and history data, $[L_1^1, L_2^1, L_3^1]^T$ and $[L_1^2, L_2^2, L_3^2]^T$ can be presented with $[0,5\%,90\%]^T$ and $[5\%,70\%,90\%]^T$ respectively.

The final result of the fuzzy logic coordinator, y_{FINAL} is described as

$$y_{FINAL} = y_1\,(1\text{-}ME\_ER)\cdot\mu + y_2(1\text{-}\mu) \qquad (\mu=\min\{\mu_1, \mu_2\}) \tag{10}$$

where y_1 and y_2 are the results from the cyclone model and the ANFIS, respectively.

3.7 The Hybrid Intelligent Model

Based on the aforementioned models, a hybrid intelligent model is presented as Fig.3. Z=$[M_{HCF}, Q_{HCF}, C_{HCF}, \dot{M}_{HCF}, \dot{C}_{HCF}]^T$, U=$[F_{ORE}(t\text{-}5), F_{WM}(t\text{-}5), Q_{WSP}(t\text{-}3), Q_W(t\text{-}1), Q_{HCF}]^T$
Z_2=$[N_{SP}, C_{SPO}, Q_{SPO}]^T$, Z_1=$[Q_W, Q_{SPO}, C_{SPO}, Q_{HCF}, C_{HCF}, Q_{HCO}, C_{HCO}]^T$

In normal situation, the final predictive result mainly depend on the output from the cyclone model. The database is used to change parameters of population balance model according to the predictive error. According to different operation states of grinding system, the fuzzy logic coordinator presents the final predictive result with the inputs from the outputs of the ANFIS model and the hydrocyclone model.

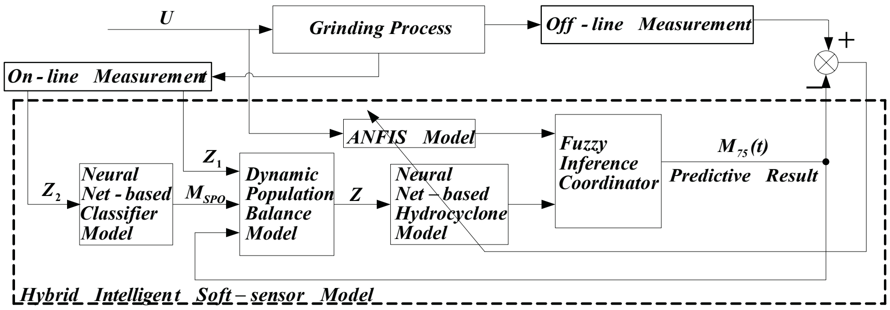

U
Grinding Process
Off - line Measurement
On - line Measurement
+
−
ANFIS Model
Z_1
Z_2
Neural Net - based Classifier Model
M_{SPO}
Dynamic Population Balance Model
Z
Neural Net − based Hydrocyclone Model
Fuzzy Inference Coordinator
$M_{75}(t)$
Predictive Result
Hybrid Intelligent Soft − sensor Model

Fig. 3. The structure of hybrid intelligent particle size soft-sensor model.

4 Results and Discussion

4.1 Dynamic Simulator Design

To generate data for the soft-sensor development and test, two dynamic simulation systems are developed for the grinding circuits shown in Fig.1.

Simulator1:
The ball mill is modeled as *N* mixers in series, which are the same as Radhakrishnan's models[5]. The sump model is the same as the model of Rajamani [6]. The hydrocyclone is modeled with empirical modeling approach proposed by Lynch and Rao [6]. The classify characteristic of classifier is different from hydrocyclone, and is modeled with the industrial spiral classifier model [8].

Simulator2:
The model presented by Dubé [3],where the slurry flow rate in mill is time-varied.

4.2 Simulation Results

800 groups data from the above simulators are used to generate training data sets. The training precision is 0.01, all the Gauss-shaped parameters are assumed to be 1.

Fig.4 presents a comparison between the soft-sensor predictions and true value, which are different from training data.

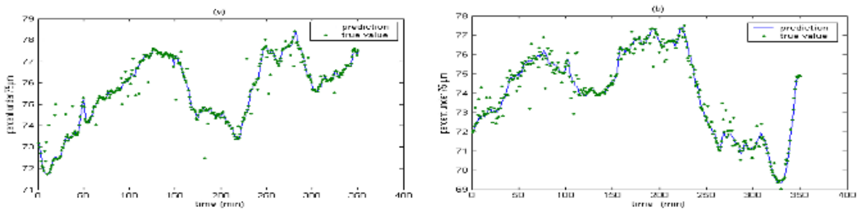


Fig. 4. Hybrid intelligent soft-sensor model predictions. (a) result with simulator1 (b) result with simulator2.

From the simulation results, it can be seen that the dynamic characteristic of the consecutive grinding process is described correctly with the soft-sensor of this paper.

5 Conclusions

In complicate industrial processes, measurements may become unavailable because of failure, removal or economic reasons, and the physical models or empirical models are not effective or precise enough for estimation, the soft-sensor based on physical models and hybrid intelligent modeling approaches may give effective solution.

References

1. Casali, A., González, G.D., Torres, F., Vallebuonam, G.: Particle Size Distribution Soft-Sensor for a Grinding Circuit. Powder Technology, **99** (1998) 15-21
2. Del, V.R.G., Thibault, J.: Development of a Soft-Sensor for Particle Size Monitoring. Minerals Engineering, **9** (1996) 55-72
3. Dubé, Y., Lanthier, R.: Computer Aided Dynamic Analysis and Control Design for Grinding Circuit. CIM Bulletin, **80** (1987) 65-70
4. González, G.D.: Soft Sensors for Processing Plants. IEEE Conf (1999)
5. Radhakrishnan, V.R.: Model Based Supervisor Control of a Ball Mill Grinding Circuit. Journal of Process Control, **9** (1999) 195-211
6. Rajamani, K., Herbst, J.A.: Optimal Control of a Ball Mill Grinding Circuit. Chemical Engineering Science, **46** (1991) 861-870
7. Tangsathitkulchai, C.: Effects of Slurry Concentration and Powder Filling on the Net Mill Power of a Laboratory Ball Mill. Powder Technology, **137** (2003) 131-138
8. Thibalt, J.: Modeling and Control of Mineral Processing Plants Using Neural Networks. IFAC on Expert Systems in Mineral and Metal Processing, Finland (1991)
9. Xu, L.: Rival Penalized Competitive Learning for Clustering Analysis, RBF Net, and Curve Detection. IEEE Trans. on Neural Networks, **4** (1993) 636-649
10. Zhang, X.D.: Research of Neural Network Particle Size Soft-Sensor for Concentration Process. Control Theory and Applications, **19** (2002) 85-88

Application of Artificial Neural Networks in Abrasive Waterjet Cutting Process*

Yiyu Lu[1], Xiaohong Li[1], Binquan Jiao[1], and Yong Liao[2]

[1] Key Lab of Ministry of Education for the Exploitation of Southwestern Resources & Environmental Disaster Control Engineering, Chongqing University, Chongqing, 400044, China
Luyiyu@hotmail.com, xhli@cqu.edu.cn, jbq_lm@163.com
[2] College of Mechanical and Energy Engineering, Zhejiang University, Zhejiang 310027, China
Lyong666@163.com

Abstract. This paper presents and discusses the application of artificial neural networks (ANN) in the abrasive waterjet (AWJ) cutting process. Backpropagation networks are chosen for the proposed network, which is written using the programming package MAT-LAB. The overall results are observed and compared with the experimental data for the performance of the networks. Based on the application it was found that the ANN is able to learn the complicated relationships between the main abrasive waterjet input parameters and cutting speed with necessary cutting quality. The proposed prediction model for certain AWJ system can be used for parameter optimization and numerical simulation of AWJ cutting process.

1 Introduction

As a unique cold high-energy beam process, the abrasive waterjet (AWJ) cutting technology has been proved to be very effective means for shape cutting of hard-to-machine engineering materials, which it is often not technically or economically feasible to cut with conventional machining. One of the principle deficiencies of AWJ cutting process is the characteristic striated surface finish, which limited the potential applications of this technology. In AWJ cutting, as the jet penetrates into the workpiece, the kinetic energy of the particles decreases continuously and the jet starts deflecting, moreover, the kinetic energy of abrasive particles wavily distributes across the jet section [1]. Thus typically, the surface produced by AWJ has two distinct zones along the cutting kerf, a smooth upper zones (cutting wear zone) and a regular oblique striated low zone (deformation wear zone). Considerable researches have been carried out to study model, mechanism, as well as surface characteristics of AWJ cutting process [2]-[6]. Due to the particular properties of the high-speed liquid-solid flow, the AWJ cutting process is a complicated and unstable process influenced by many factors and phenomena involved in material cutting are very complex and often not well understood. It is very difficult to establish an effective theoretical model for the AWJ cutting process. Until now, the AWJ cutting process parameters for required cutting quality are empirically determined generally. The empirical parametric models cannot give reliable predictions for its valid range of parameters.

* Grant sponsor: NSFC; Grant number: 50334060.

The multi-layer feed-forward network with the back-propagation algorithm is most commonly used in optimizing practical nonlinear problems. Neural network has been successfully used in prediction of waterjet depainting process [7] and in concrete structures [9].

In this paper, the back-propagation network is used to develop a sufficiently accurate, reliable and intelligent numerical prediction model of AWJ cutting process. Trained by experimental database for several typical materials, the network with definite structure and parameters can formulate a good approximation to complex nonlinear relationships between the cutting speed and the inputs parameters (material, water pressure and required cutting quality).

2 ANN Model of AWJ Cutting Process

2.1 Topological Structure of ANN Model

According to practical requirements, three main parameters: workpiece material type X_1, waterjet pressure X_2 and desired cutting quality index X_3 are set as input parameters of the network, while cutting speed Y is set as output of the network. According to the number of input and output parameters, the network architecture used in modeling of AWJ cutting process is shown in Fig.1. It is a standard three-layer feed-forward neural network with one hidden layer, three input neurons and one output neuron. The tan-sigmoid activation transfer function in hidden layer and linear activation transfer function in output layer are used respectively. The number of neurons in the hidden layer will be determined by the accuracy and calculating time of training network.

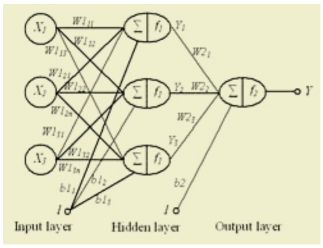


Fig. 1. Topological Structure of Network.

2.2 Experimental Procedure for Database

In these experiments, the effects of workpiece material and thickness, waterjet pressure, and cutting speed on AWJ cutting surface quality are investigated to prepare pattern database for the neural network. The experiments are carried out with a two-axis NC AWJ system in which waterjet pressure varies from 80 to 220MPa, traverse speed varies from 10 to 2500mm/min, and abrasive flow rate varies from 95 to 148 g/min. The other parameters keep constant as following: standoff distance=2mm, diameter of water orifice=0.28mm, diameter of mixing tube=0.8 mm, length of mixing tube=76mm, and jet impact angle=90degrees. Abrasive is 80# garnet. The following materials are selected as test specimens: 3.0, 4.5 and 11mm thick steel plates, 4.0 and

10mm thick aluminum plates, 7.0mm thick ceramic tiles, 4.2mm thick glass plates, 8.0mm thick PVC plates, 8.0mm thick plexiglass plates and 10mm thick marble plates.

An experimental database consists of 300 input/target pairs representing the AWJ cutting performance. The database is normalized between [0, 1] and divided into two data sets: training and checking, which are used to train and check the neural network. The dimensionless cutting quality level index X_3 is defined as the ratio of maximum cutting depth to workpiece thickness. X_3=1 represents criteria for separation cutting corresponding to the maximum cutting speed, X_3=2 represents rough surface finish with striation marks at the lower half surface, X_3= 3 represents smooth/rough transition criteria with slight striation marks, X_3=4 represents a smooth surface finish without striation for most cases, and X_3=5 represents best surface finish.

2.3 Training of ANN Model

Once the network weights and biases have been initialized by random values, the network is ready for training. as shown in Fig.1, for a training pattern vector, output of hidden layer $\boldsymbol{Y_1}$ is:

$$Y_{1j} = f\left(\sum_{i=1}^{3} W_{1ij} X_i + b_{1j}\right) \quad . \tag{1}$$

Where $\boldsymbol{X}_\mathrm{i}$ are inputs of input layer, $\boldsymbol{W}_{1\mathrm{ij}}$ are weights between neurons of input layer and hidden layer, $\boldsymbol{b}_{1\mathrm{j}}$ are biases of hidden layer, i=1~3 is index of the input layer neurons, j=1~n is index of the hidden layer neurons. The tan-sigmoid activation transform function in hidden layer is:

$$f(x) = \frac{1}{1+e^{-x}} \quad . \tag{2}$$

The actual outputs of output layer $\boldsymbol{Y_2}$ are:

$$Y_2 = \sum_{j=1}^{n} W_{2j} Y_{2j} + b_2 \quad . \tag{3}$$

Where, $\boldsymbol{W}_{2\mathrm{j}}$ are weights between neurons of hidden layer and output layer, $\boldsymbol{b}_2$ is bias of output layer. The network error performance function compared with target value $\boldsymbol{Y}$ (experimental value) for pattern k is expressed as:

$$E_k\left(W_{1ij}, W_{2j}, b_{1j}, b_2\right) = \frac{1}{2}\left(Y_2^k - Y^k\right)^2 \quad . \tag{4}$$

The standard gradient descent BP algorithm has a slow convergence rate and is easily trapped in local minimal error. In this work, the Levenberg-Marquardt algorithm based on numerical optimal methods is used for faster and more accurate network learning. The weight and bias vector $\boldsymbol{U}$ is adjusted as following iterative process:

$$U^{(m+1)} = U^{(m)} - \eta^{(m)}\left(H^{(m)} + \lambda^m I\right)^{-1} J^T E \quad . \tag{5}$$

Where $\boldsymbol{\eta}$=(0,1) is learning rate, λ is the adaptive value, $\boldsymbol{I}$ is the identity matrix, $\boldsymbol{E}$ is a vector of network errors, and $\boldsymbol{H}$ is the Hessian matrix, which is second derivatives of the network errors with respect to the weights and biases:

$$H_{ij} = \partial E^2 / \left(\partial x_i \partial x_j \right) \quad . \tag{6}$$

As it is complex and expensive to compute the Hessian matrix for feed-forward neural networks, the Levenberg-Marquardt algorithm is designed to approach Hessian matrix as:

$$H = J^T J \quad . \tag{7}$$

The Jacobian matrix ***J***, which contains first derivatives of the network errors, can easily be computed through a standard back-propagation technique.

In the primary iteration with large λ, this algorithm becomes gradient descent with a small step size. The scale λ is decreased gradually to zero with approaching to optimum point. This algorithm is faster and more accurate near an error minimum.

The network training finishes when the sum-square error of all training patterns *P* is less than expected value *ε*:

$$E_t = \frac{1}{2P} \sum_{k=1}^{P} \left(Y^k - t^k \right)^2 \leq \varepsilon \quad . \tag{8}$$

3 Results and Discussion

Figure 2 shows the training convergence of the neural network for AWJ cutting process. The number of neurons in hidden layer is 12, the training of this network takes only 26 cycles, and the final sum-square error is 0.001. The results of the cutting speed prediction using ANN model are presented in figures 3-12. The maximum relative error between prediction results and experimental results is 5.2%. A trained and checked network, which builds an inferable model, can be applied in AWJ parameter optimization and process prediction, etc. When a new or detailed analysis is required, a comparatively modest additional database and training will be necessary.

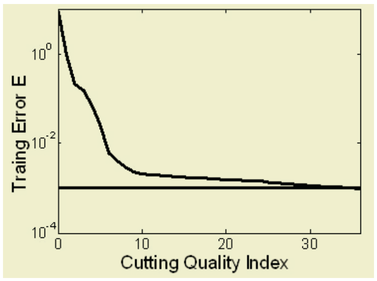


Fig. 2. Training of the Neural Network.

It is found that the prediction results are obviously different to the results predicted by the semi-empirical model published by Zeng, et al. in 1993. It is mainly caused by the large difference between the ranges of valid parameter values for two models. It is also found that the cutting surface of plastic material is generally smoother than that of brittle material on the same cutting quality index.

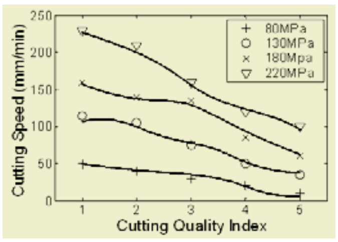


Fig. 3. ANN Prediction Results for Steel with thickness of 3.0mm.

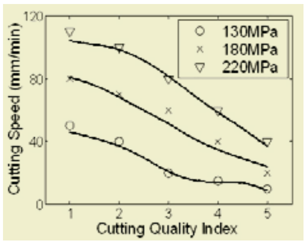


Fig. 4. ANN Prediction Results for Steel with thickness of 4.5mm.

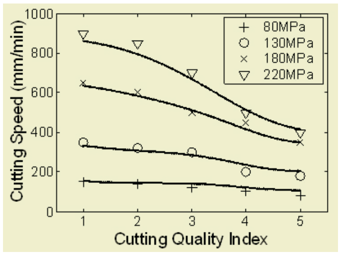


Fig. 5. ANN Prediction Results for Steel with Thickness of 11mm.

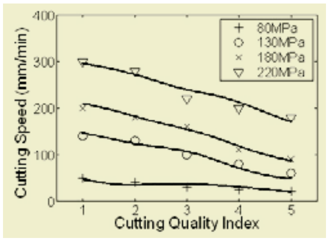


Fig. 6. ANN Prediction Results for Aluminum with Thickness of 4.0mm.

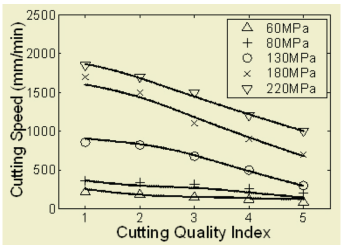


Fig. 7. ANN Prediction Results for Aluminum with Thickness of 10mm.

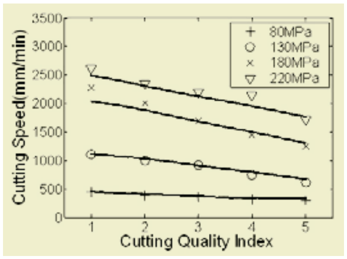


Fig. 8. ANN Prediction Results for ceramic tile with Thickness of 7.0mm.

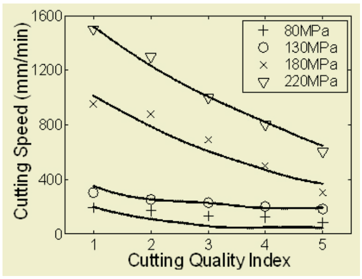


Fig. 9. ANN Prediction Results for glass Plate with Thickness of 4.2mm.

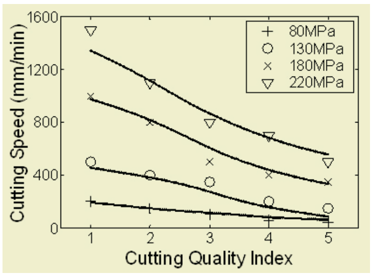


Fig. 10. ANN Prediction Results for PVC Plate with Thickness of 8.0mm.

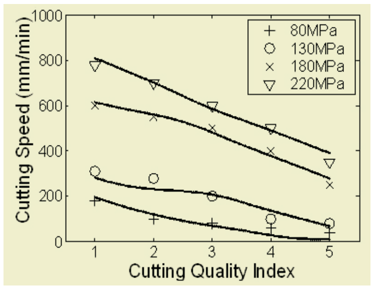


Fig. 11. ANN Prediction Results for Plexiglass Plate with Thickness of 8.0mm.

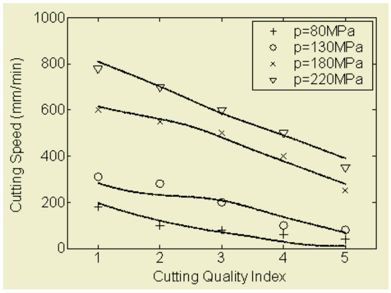


Fig. 12. ANN Prediction Results for Marble Plate with Thickness of 10mm.

4 Conclusions

The overall results indicate that the ANN is able to learn the complicated relationships between main AWJ input parameters and cutting speed with necessary cutting surface quality. The proposed prediction model for certain AWJ system can be used for parameter optimization and numerical simulation of AWJ cutting process.

References

1. Siores, E., Chen, L., et al.: Improving Surface Finish Generated by the Abrasive Waterjet Process. International Symposium on Advances in abrasive Technology. Sydney, Australia (1997) 187-191
2. Hashish, M.: A Modeling Study of Metal Cutting with Abrasive Waterjets. Journal of Manufacturing Science and Engineering, **106** (1984) 88-100
3. Hashish, M.: Characteristics of Surface Machined with Abrasive-waterjets. ASME Journal of Manufacturing Science and Engineering, **113** (1991) 354-362
4. Zeng, J. and Kim, T. J.: Parameter Prediction and Cost Analysis in Abrasive Waterjet Cutting Operations. Proceedings of the 7th American Watejet Conference, WJTA, St Louis, MO, USA (1993) 175-190
5. Zeng, J. and Munoz, J.: Surface Finish Evaluation for Abrasive Waterjet Cutting. Proceedings of the 9th American Waterjet Conference, WJTA, Dearborn, Michigan (1997) 1-14
6. Chen, L., Siores, E., Patel, K.: Improving the Cut Surface Qualities Using Different Controlled Nozzle Oscillation Techniques.International Journal of Machchine Tools & Manufacture, **42** (2002) 712-722
7. Babets, K., Geskin, E.S., Chaudhuri, B.: Neural Network Model of Waterjet Depainting Process. Proceedings of the10th American Waterjet Conference, WJTA, Houston, Texas, USA (1999)
8. Hecht-Nielsen, R.: Theory of the Backpropagation Neural Network. Proceedings of the International Joint Conference on Neural Networks, **1** (1989) 593-605
9. Hadi, M.N.S.: Neural Networks Applications in Concrete Structures. Computers & Structures, **81** (2003) 373-381

Intelligent Tool Condition Monitoring System for Turning Operations

Hongli Gao and Mingheng Xu

School Of Mechanical Engineering, Southwest Jiaotong University
Chengdu, Sichuan 610031, China

Abstract. In order to predict tool wear accurately and reliably under different cutting conditions, a new monitoring methodology for turning operations is proposed. Firstly, the correlation coefficients approaches were used to indicate the dependencies between the different sensed information features and tool wear amount, and the most appropriate features were selected. Secondly, B-spline neural networks were introduced to model the non-linear relationship between extracted features and tool wear amount, and multi-sensor information were fused by an integrated neural network. Lastly, the final result of tool wear was given through fuzzy modeling. Experimental results have proved that the monitoring system based on the methodology is reliable and practical.

1 Introduction

Tool condition monitoring is the key technique in automatic and unmanned machining process. The wear of cutting tool can cause the following disadvantages such as decreased tool life, inaccurate dimension, poor surface finish of workpiece, and high production cost etc, so construction of a reliable and cost-effective real-time monitoring system has become crucial for automatic machining system and Flexible manufacturing system.

With the development of neural network, many researchers have studied tool wear monitoring technique based on neural networks. Feed forward back propagation neural network, fuzzy neural network, SOM, ART, recurrent neural network, and genetic systems are all applied to predict tool condition, and make great progresses [1]-[5]. However, there are much uncertainties and random factors in the process of metal cutting, it make tool wear monitoring become difficult, especially for tool wear value prediction under various machining condition.

In the paper, AE sensor and dynamometer are used to monitor tool wear states. Lots of features associated with tool wear are obtained by the methods included time domain analysis, frequency domain analysis and wavelet packet transformation. The correlation coefficient approach is introduced to calculate dependences between features and tool wear, and the most sensitive features to tool wear are chosen according to the sort ascending of correlation coefficient for each signal. B-spline neural network are proposed to model the nonlinear relation between selected features and tool wear. When B-spline are utilized to model data sets, it is possible to modify the data locally, and the corresponding change in the network's response is also local. This is main advantage that makes it attractive for on-line adaptive modeling. Finally, integrated neural network fuse the estimated tool wear of each signal, and the final tool wear value is given through fuzzy modeling. Experimental results show that the method is effective for tool wear prediction in industry.

2 Design of Experiment

2.1 Sensor Selection

Up to now, a variety of tool wear sensing methods have been used to measure parameters which are correlated with tool wear, including cutting force signal detection, cutting temperature detection, electrical resistance measurement, cutting vibration detection, and measurement of acoustic emission [6].

As a nondestructive testing method, AE has been applied to monitor manufacturing process. Generally, AE sensor test the stress wave produced in primary and secondary zones when cutting, and in the presence of flank wear, the interface between tool and workpiece becomes an additional zone of acoustic emission generation due to intensive friction. The major advantage of using AE is that frequency of AE signals is much higher than that of machine vibrations and environmental noises [7]. Most research results has shown that cutting force signals are more stable and sensitive to tool wear than other measurements, tool wear directly cause the increase of cutting forces, so dynamometer and piezoelectric AE sensor are used in the experiment described in the paper.

2.2 Experimental Set-Up

The schematic diagram of the experimental set-up is shown in Fig 1. A series of machining experiments were conducted on a conventional lathe CA6150. In the turning operation, a Kistler 9257B dynamometer was mounted on the cross slide of the lathe, and a Kistler 9403 tool holder was fixed on the top surface of the dynamometer. Moreover, a commercial piezoelectric AE transducer (type RNT-200) made in Beijing university of Aeronautics and Astronautics was mounted on the surface of tool holder, and a YT15(type 41065A) inserts were used to machine a 45# middle-carbide steel workpiece (Φ95×600mm) . In the experiment, signals of three orthogonal cutting forces Fx, Fy, Fz were measured and amplified by a multichannel charge amplifier (type 5019B) , and through a A/D converter board to computer. Sampling rate is 3000Hz and sampling time is 10s. In the meantime, the AE signals were amplified and filtered with a band-pass filter, high-passed at 60 kHz and low-passed at 2MHz, and then sent to computer via an A/D converter board type Jv52014, sampling rate is 2MHz, sampling length is 256k.

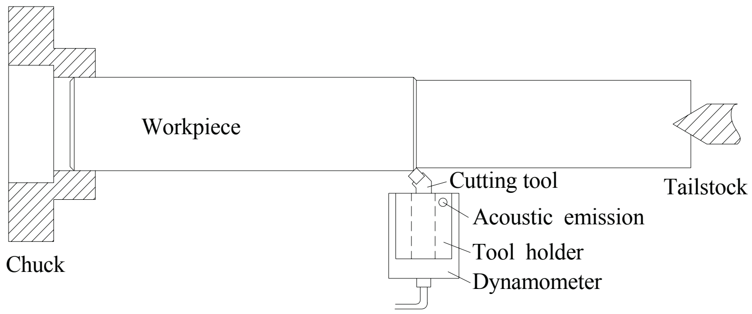


Fig. 1. The schematic diagram of the experimental set-up.

2.3 Experimental Conditions

The machining conditions used in the experiment is shown in Table.1, a tool microscope was used to observe and measure the flank wear of the cutter after machining the workpiece every five minutes. Tool inserts were artificially worn in a tool grinder beforehand according to the classification of tool flank wear shown in Table 2.

Table 1. Experimental conditions.

Cutting speed (m/min)	59.6	74.5	95.5	120
Feed rate (mm/r)	0.2	0.3	0.41	0.51
Depth of cut (mm)	0.5	1	1.5	2

Table 2. Classification of tool flank wear.

Classification	A	B	…	H	I	J
Tool wear value (mm)	0	0.05	…	0.35	0.4	0.5

3 Features Extraction of AE and Dynamometer

3.1 Features Extracted from AE Signals

It can be seen from AE signal in a time domain that there aren't obvious change with the progression of tool wear, so frequency analysis or time-frequency analysis must be used to extract features associated with tool wear. Compared with Fourier transform, an important advantage of wavelet transform is the possible localization in the time domain and in the frequency domain at the same time. Thus, wavelet packet transform is used to decompose AE signal and extract the most appropriate features. AE signal in the experiment are decomposed with db3 wavelet, the decomposing level is 5. The decomposed frequency bands are shown in tab 3. Fig 2 shows the partial decomposing results of AE signal in different frequency band through wavelet packet decomposition.

Table 3. The frequency range of AE signal through wavelet packet decomposition.

Serial number	1	…	31	32
Frequency band (kHz)	0～31.25	…	937.5～968.75	968.75～1000

In general, the root mean square (RMS) values of each frequency band extracted from the decomposed signal can be used as the monitoring features for tool wear, so we get 32 features from the 32 frequency band shown in Table 3.

3.2 Features Extracted from Dynamometer

The measured force is composed of static force and dynamic force. The analytical results indicate that the mean values of cutting forces Fx, Fy and Fz increase with progression of tool wear, frequency analysis and wavelet transform in frequency band 0-1500 Hz can also provide more information, so average value, resultant cutting force, root mean square, standard deviation, power value, skew value, kurtosis value,

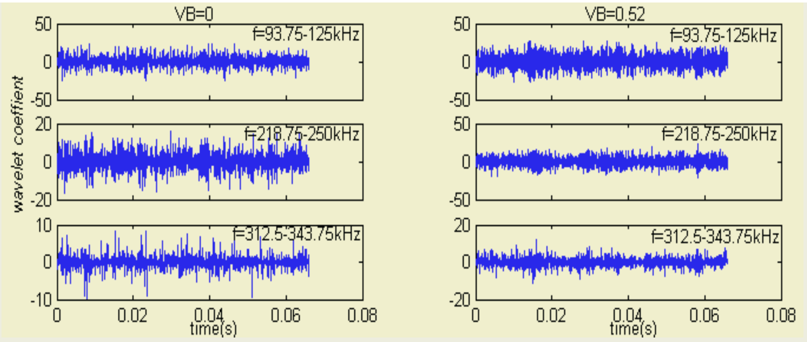


Fig. 2. Decomposing results of AE by wavelet packet transform. Cutting speed: 95.5 m/min; feed rate: 0.2 mm/r; depth of cut: 0.5 mm; workpiece material: 45# steel; tool material: YT15; without coolant.

frequency analysis, wavelet packet transform are all may be the input features of neural network. Then 40 features were acquired through above signal processing methods for each component of forces.

3.3 Feature Selection

The correlation coefficients approach is introduced to indicate the non-linear dependencies between features extracted from dynamometer and acoustic emission signals and tool wear. Generally, the correlation coefficient ρ can be calculated by equation(1):

$$\rho = \frac{\sum_i (\mathrm{x}_i - \overline{\mathrm{x}})(\mathrm{y}_i - \overline{\mathrm{y}})}{\sqrt{\sum_i (\mathrm{x}_i - \overline{\mathrm{x}})^2}\sqrt{\sum_i (\mathrm{y}_i - \overline{\mathrm{y}})^2}} \tag{1}$$

Where $\overline{x}$ and $\overline{y}$ are mean values of x and y, respectively; x_i are features extracted from signals; y_i are tool wear value, ρ is the correlation coefficient. When $|\rho|$ is approaching one, the relation between x and y is approaching linear, and when ρ is close to zero, y is almost independent of x.

It is known from equation (1) that the degree of independencies between signal features and tool wear are different. There are 120 features extracted from dynamometer and 32 features extracted from AE signal in all. In order to avoid curse of dimensionality, reduce learning time, and improve reliability and precision of monitoring system, we choose three most sensitive features for each signal according to sort ascending of the correlation coefficient ρ. Therefore, 12 features are chosen as the input vector to B-spline neural network.

4 Methodology

To data, the method of using a single network to model the mapping relation is still in prevail, and realizes tool classification successfully. However, simulation results indicated the method couldn't make precise prediction of tool wear value, a multi-sensor integrated method based on integrated neural network is then presented. Its structure is given in fig 3, a single B-spline neural network , which input nodes is the number of features, is used for each signal. Thus, four networks corresponding to three force signals and a AE sensor are implemented to calculate tool wear value, and final output of monitoring system is given by fuzzy modeling which fuse the outputs of the four sub networks.

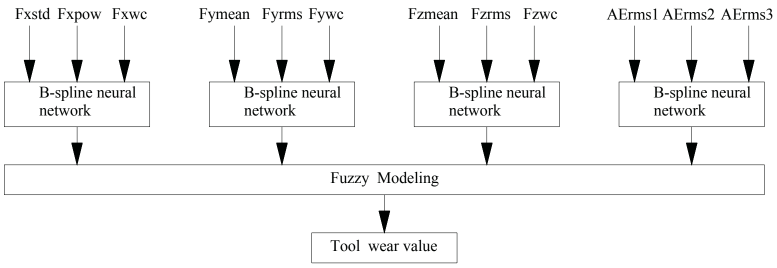


Fig. 3. The structure of integrated neural network.

5 B-Spline Neural Network

5.1 Structure of B-Spline Neural Network

The structure of B-spline neural network, as shown in fig 4. The network can be decomposed into two parts: a static, nonlinear topology conserving map and a adaptive linear mapping. Suppose x=(x_1, x_2, ⋯ x_n) be the n-dimensional input vector of network, the network's output is denoted by y. The jth multivariate B-spline function which defined on a lattice in the normalized input space is obtained by taking the tensor product of the univariate basis functions, the number of the multivariate B-spline function p depends on the order of the B-spline function and the number of knots defined on each axis of input vector x. Thus the output of the B-spline neural network is given by [8]:

$$y=\sum_{j=1}^{p} w_j N_k^j(x) \tag{2}$$

Where w_j is the weight corresponding to the jth basis function.

5.2 Learning Algorithm

The ability of functional approximation for B-spline network is determined by the order of B-spline basis function, knot placement of lattice defined on the input space and weight matrix from hidden layer to output layer. As the structure of lattice and the order of B-spline basis function is determined, the only mutative parameter is weight matrix *W*, and *W* can be calculated by using gradient descent rules.

The performance function of B-spline network is generally the Mean Square output error (MSE), it's given by:

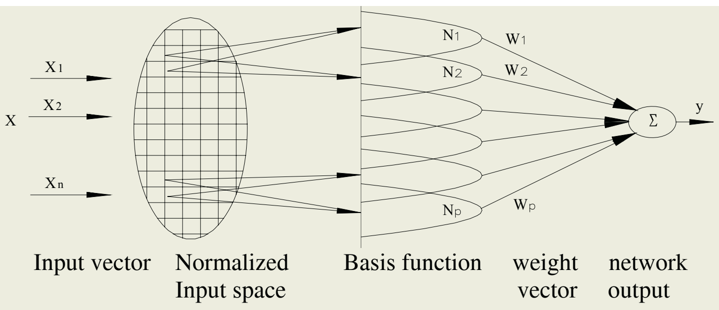


Fig. 4. The structure of B-spline neural network.

$$E = \frac{1}{L}\sum_{i=1}^{L}(\hat{y}_{\mathrm{i}} - y_i)^2 \tag{3}$$

Where $\hat{y}_{\mathrm{i}}$ is the desired output of the network, $\mathrm{y_i}$ is the practical output of the network, and L is the number of training sample. The weight vector are obtained by minimizing E, and are updated by,

$$\Delta \mathrm{w_j} = -\delta\partial \mathrm{E}/\partial \mathrm{w_j} = -2\delta\sum_{i=1}^{L}(\hat{y}_i - y_i)\partial \mathrm{y_i}/\partial \mathrm{w_j} = -\delta_0\sum_{i=1}^{L}(\hat{y} - y_i)N_k^j(x_i) \tag{4}$$

Where δ is the learning rate, and δ_0=2δ.

6 Experimental Results

The three input features of each B-spline neural network construct the three-dimensional lattice-based space, and input space was divided into 216 cubes as interior knots is five. When the order of B-spline univariate basis function is two, 343 multivariate basis functions are formed, and the number of adjustable weight vector is 343. For an arbitrary input vector, only 8 outputs of multivariate basis functions is non-zero, so computational cost is largely reduced. The recognized results of tool wear monitoring system founded on above method is shown in fig 5, the maximum error is 0.03, and the minimal error is zero. The results indicate that proposed methodology can accurately predict tool wear value.

7 Conclusions

Based on the proposed methodology and experimental results, the following conclusions are drawn:

1. For force signals, time domain analyses and frequency domain analyses is the most appropriate ways, but for AE signal, wavelet packet transformation is the best method.
2. The correlation coefficients approach introduced in the paper can effectively simplify and optimize the work of features selection in the design of tool condition monitoring system.

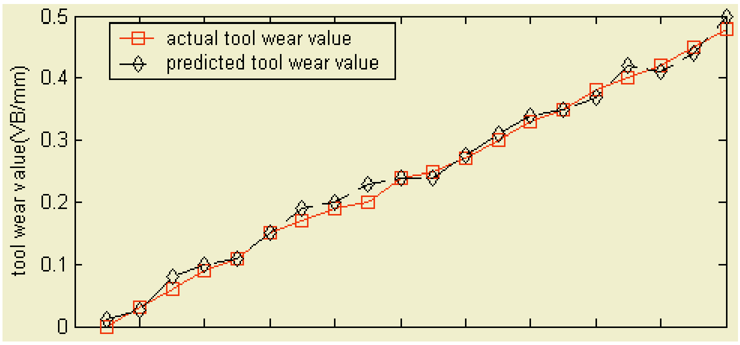


Fig. 5. Comparison between actual tool wear value and predicted tool wear value.

3. The method of using B-spline neural network to model the relation between features and tool wear amount is able to predict the amount of tool wear accurately
4. The data fusion of multi-sensor by integrated neural network is more reliable and precise than a single neural network.

References

1. Quan, Y., Zhou, M., Luo, Z.: On-line Robust Identification of Tool-wear via Multi-Sensor Neural Network Fusion. Engineering Application of Artificial Intelligence, 11 (1998) 717-722
2. Dimla Sr, D.E., Lister, P.M.: On-line Metal Cutting Tool Condition Monitoring.Ⅰ: Tool-state Classification Using Multi-layer Perception Neural Networks, International Journal of Machine Tools & Manufacture, **40** (2000) 769-781.
3. Wilkinson, P., Reuben, R.L.: Tool Wear Prediction from Acoustic Emission and Surface Characteristics via an Artificial Neural Network. Mechanical System and Signal Processing, **13** (1999) 955-966
4. Kuo, R.J.: Multi-sensor Integration for On-Line Tool Wear Estimation Through Artificial Neural Networks and Fuzzy Neural Network
5. Bernhard Sick: On-line and Indirect Tool Wear Monitoring in Turning with Artificial Neural Networks: A Review of More than a Decade of Research. Mechanical System and Signal Processing, **16** (2002) 487-546
6. Chung, K.T., Geddam, A.: A Multi-sensor Approach to the Monitoring of End Milling Operations. Journal of Materials Processing Technology, **139** (2003) 15-20
7. Jemielniak, K.: Some Aspects of AE Application in Tool Condition Monitoring. Ultrasonics, **38** (2000) 604-608
8. Brown, M., Harris, C.: Neurofuzzy Adaptive Modelling and Control[M], Prentice Hall International(UK) limited (1994) 296-308

A Recurrent Neural Network Modeling for Automotive Magnetorheological Fluid Shock Absorber

Changrong Liao[1], Honghui Zhang[1], Miao Yu[1], Weimin Chen[1], and Jiansheng Weng[2]

[1] Key Laboratory for Optoelectronic Technology & System Under Ministry of Education, Chongqing University, Chongqing 400044, China
crliao@cqu.edu.cn

[2] Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China

Abstract. Automotive Magnetorheological (MR) fluid shock absorbers have been previously characterized by a series of nonlinear differential equations, which have some difficulties in developing control systems. This paper presents a recurrent neural network with 3 input neurons, 1 output neuron and 5 recurrent neurons in the hidden layer to simulate behavior of MR fluid shock absorbers to develop control algorithms for suspension systems. A recursive prediction error algorithm has been applied to train the recurrent neural network using test data from lab where the MR fluid shock absorbers were tested by the MTS electro-hydraulic servo vibrator system. Training of neural network model has been done by means of the recursive prediction error algorithm presented in this paper and data generated from test in laboratory. In comparison with experimental results of MR fluid shock absorbers, the neural network models are reasonably accurate to depict performances of MR fluid shock absorber over a wide range of operating conditions.

1 Introduction

The main functions of automotive suspension systems are to provide support, to provide stability and direction control during handling maneuvers and to provide effective isolation from road irregularities. Automotive suspensions based on MR fluid shock absorbers have received significant attention in recent years, because they offer the adaptability of active suspension without requiring large power consumption and control system is stabilization. Due to its mechanical simplicity and high dynamic range, MR fluid shock absorbers can mesh well with application demands and constraints of improving suspension system behaviors. The MR shock absorbers have previously been characterized by a series of nonlinear differential equations, which result in some troubles to developing control systems. Researchers have investigated two types of dynamic models for controllable fluid damper: non-parametric models [1], [2] and parametric models [3],[4]. The parametric models can describe the nonlinear force-displacement and force-velocity behaviors, but they are readily able to obtain force in the low velocity and model parameters are not readily ascertained. In order to overcome the inconvenience, a phenomenological model based on the Bonc-wen hysteresis model has been proposed by Spenceer [3], in which some parameters will been obtained by optimal method and experimental data. Alternatively, this paper presents a recurrent neural network with 3 input neurons, 1 output neuron and 5 recurrent neurons in the hidden layer to simulate the dynamic behaviors of automotive MR fluid shock absorbers to develop control algorithms for automotive suspension systems. Training of neural network model has been done by recursive prediction error algorithm using data generated from test in laboratory.

2 Modeling of MR Fluid Shock Absorbers

The recurrent neural network designed to approximate damping performances of MR fluid shock absorbers is three-layer neural network with one local feedback loop in the hidden layer, whose architectures are shown in Figure 1. The primary property of recurrent neural network is the ability of the network to learn from its environment and to improve its performances by means of process of adjustments applied to its weights. The recurrent network with input signal $I_i(t)$ corresponding to input neuron *ith* has one output $O(t)$ by local feedback loop neurons in the hidden layer whose output sum is $S_j(t)$ corresponding to the neuron *jth*. The activation functions for both input neurons and output neurons are linear functions, while the activation for neurons in the hidden layer is sigmoid function.

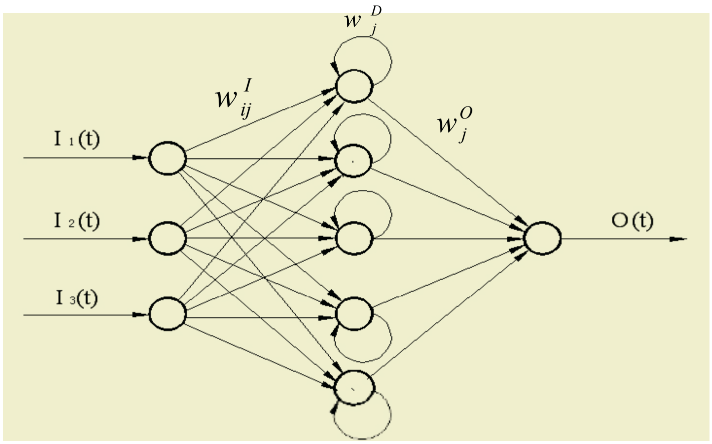


Fig. 1. Architectural graph of recurrent network.

$$X_j(t) = f\left[\sum_{i=1}^{p} w_{ij}^I \cdot I_i(t) + w_j^D \cdot X_j(t-1)\right]. \tag{1}$$

$$O(t) = \sum_{j=1}^{q} w_j^O \cdot X_j(t) \cdot \tag{2}$$

where w_{ij}^I , w_j^D , w_j^O are weight of the recurrent neural network, $X_j(t)$ is output of neuron with local feedback loop neuron in the hidden layer, p, q are input neuron number and feedback neuron number respectively. The modeling of MR shock absorbers based on recurrent neural network is shown figure 2. The weight of recurrent neural network can be adjusted by minimizing the error existing between MR shock absorber outputs and the recurrent neural network outputs.

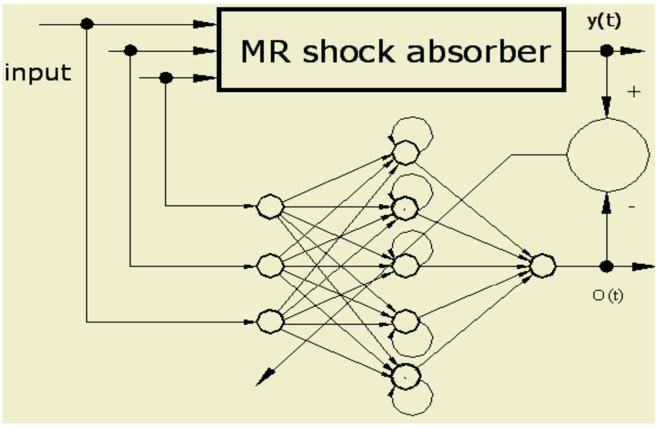


Fig. 2. The weight adjustment by minimum error.

3 Training Algorithm for Recurrent Neural Network

Under the condition of piston velocity, piston displacement and excitation electric current being given in accordance with the operating conditions for automotive suspension requirements, damping force $y(t)$ generated by MR shock absorbers can be dynamically obtained by tested through MTS electro-hydraulic servo vibrator system. The outputs of recurrent neural network $O(t)$ are used to close approximation to damping forces $y(t)$ by minimizing the error signal existing between MR shock absorber damping force and output of recurrent neural network. The objective function $E(t)$ can be defined in the terms of the error signal e(t) as[5]

$$E(t) = \frac{1}{2}[y(t) - O(t)]^2 = \frac{1}{2}e^2(t). \tag{3}$$

That is, $E(t)$ is the instantaneous value of the error energy. The step-by-step adjustments to synaptic weights of neuron are continued until the system reach steady state, i.e. the synaptic weights are essentially stabilized. Differentiating $E(t)$ with respect to weight vectors w yields

$$\frac{\partial E(t)}{\partial w} = -e(t)\frac{\partial O(t)}{\partial w}. \tag{4}$$

From expression (1), (2) and (3), differentiating $O(t)$ with respect to the weight vectors w_j^O, w_j^D, w_{ij}^I respectively yields

$$\frac{\partial O(t)}{\partial w_j^O} = X_j(t),\ \frac{\partial O(t)}{\partial w_j^D} = w_j^O \frac{\partial X_j(t)}{\partial w_j^D},\ \frac{\partial O(t)}{\partial w_{ij}^I} = w_j^O \frac{\partial X_j(t)}{\partial w_{ij}^I}. \tag{5}$$

From expression (1), (2) and (3), analyzing $\frac{\partial X_j(t)}{\partial w_j^D}$ and $\frac{\partial X_j(t)}{\partial w_{ij}^I}$ yields respectively recurrent formulas

$$\frac{\partial X_j(t)}{\partial w_j^D}=\frac{\partial f[S_j(t)]}{\partial w_j^D}\cdot\left[X_j(t-1)+\frac{\partial X_j(t-1)}{\partial w_j^D}w_j^D\right],\ \frac{\partial X_j(0)}{\partial w_j^D}=0\cdot \tag{6}$$

$$\frac{\partial X_j(t)}{\partial w_{ij}^I}=\frac{\partial f[S_j(t)]}{\partial w_{ij}^I}\cdot\left[I_i(t)+w_j^D\frac{\partial X_j(t-1)}{\partial w_{ij}^I}\right],\ \frac{\partial X_j(0)}{\partial w_{ij}^I}=0\cdot \tag{7}$$

Having computed the synaptic adjustments, the updated values of synaptic weights are determined by

$$w(t+1)=w(t)+\eta\cdot e(t)\cdot\frac{\partial O(t)}{\partial w}\cdot \tag{8}$$

where η the leaning-rate parameter, A detailed convergence analysis of the recurrent training algorithm is rather complicated to acquire the leaning-rate parameter value. According to expression (8), the weight vectors w for recurrent neural network can be adjusted. We establish a the *Lyapunov* function as follows $V(t) = 1/2\cdot e^2(t)$, whose change value $\Delta V(t)$ can be determined after some t iterations, in the sense that

$$\Delta V(t)=\frac{1}{2}\left[e^2(t+1)-e^2(t)\right]\cdot \tag{9}$$

We have noticed that the error signals $e(t)$ after some t iteration can be expressed as

$$e(t+1)=e(t)+\Delta e(t)=e(t)+\frac{\partial e(t)}{\partial w}\cdot\Delta w\cdot \tag{10}$$

from expression (10) and (11), $\frac{\partial e(t)}{\partial w}=-\frac{\partial O(t)}{\partial w}$, $\Delta w=-\eta e(t)\frac{\partial e(t)}{\partial w}=\eta e(t)\frac{\partial O(t)}{\partial w}$, the *Lyapunov* function increment can determined after some t iteration as follows

$$\Delta V(t)=-\eta\cdot e^2(t)\cdot\left\|\frac{\partial O(t)}{\partial w}\right\|^2+\frac{1}{2}\eta^2\cdot e^2(t)\cdot\left\|\frac{\partial O(t)}{\partial w}\right\|^4=-\lambda e^2(t)\cdot \tag{11}$$

where $\lambda=\frac{1}{2}\left\|\frac{\partial O(t)}{\partial w}\right\|^2\eta\left[2-\eta\left\|\frac{\partial O(t)}{\partial w}\right\|^2\right]\geq\frac{1}{2}\left\|\frac{\partial O(t)}{\partial w}\right\|^2\eta[2-\eta']>0$ · $\eta'=\eta\max_t\left\|\frac{\partial O(t)}{\partial w}\right\|^2$,

if $0<\eta'<2$, then $\lambda>0$, $\Delta V(t)<0$ and $e^2(t+1)<e^2(t)$, the recurrent training algorithm is convergent. To obtain optimal leaning-rate and rapid convergence, we have marked a choice $\eta' = 1$ in the light of the literature (Ku, C. C. and Lee, K. Y. 1995). The leaning-rate parameter of recursive prediction error algorithm can be expressed as

$$\eta^*=1/\max_t\left\|\frac{\partial O(t)}{\partial w}\right\|^2\cdot \tag{12}$$

4 Application to MR Shock Absorber

In order to obtain the training data and validate the recurrent neural networks model for MR shock absorbers, the automotive MR shock absorbers that were designed and fabricated in Chongqing University have been tested by MTS electro-hydraulic servo vibration system in the National Center for Coach Quality Supervision of China. The MR shock absorber and experimental site are shown in figure 3. In the electro-hydraulic servo vibration system, a hydraulic actuator is employed to drive the MR shock absorber from sinusoidal displacement cycles with amplitude of 25mm. A servo-valve was used to control actuating frequency according to the automotive shock absorber test standard JB3985-85. The applied current ranged from 0.0 to 1.4A in increments of 0.2A. The plots of damping force versus piston displacement and plots of damping force versus piston velocity have been recoded respectively to compare prediction those of recurrent neural networks. Those data about damping forces and piston velocity have been saved in hard disk of test computer to train recurrent neural networks.

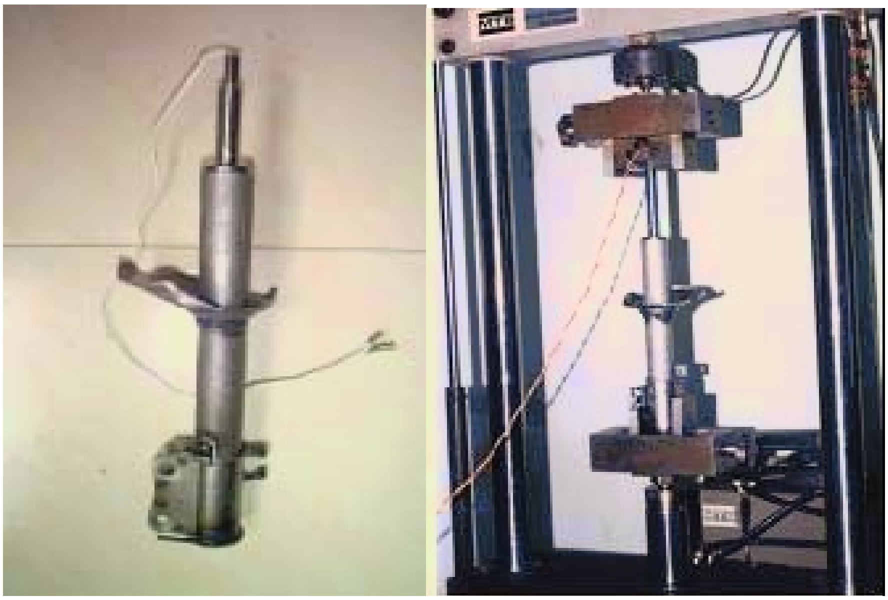


Fig. 3. MR shock absorber and test site.

Training of the recurrent neural network has been done by the recurrent algorithm presented in this paper using the data generated from test in laboratory, which is programmed in MATLAB from mathematical equations mentioned above. Both MR shock absorber damping force versus piston displacement and MR shock absorber damping force versus piston velocity obtained by experimental test have been compared with those predicted by the recurrent neural network respectively. The comparisons shown in figure 4 and figure 5 have demonstrated that the recurrent neural network model and its training algorithm presented by authors are reasonably precise to prognosticate behaviors of MR shock absorber. It can be seen that the predicted forces match well with the forces of the test and the trained neural network model can be a good approximation of the MR fluid shock absorbers performances, which can be integrated into a semi-active suppression systems.

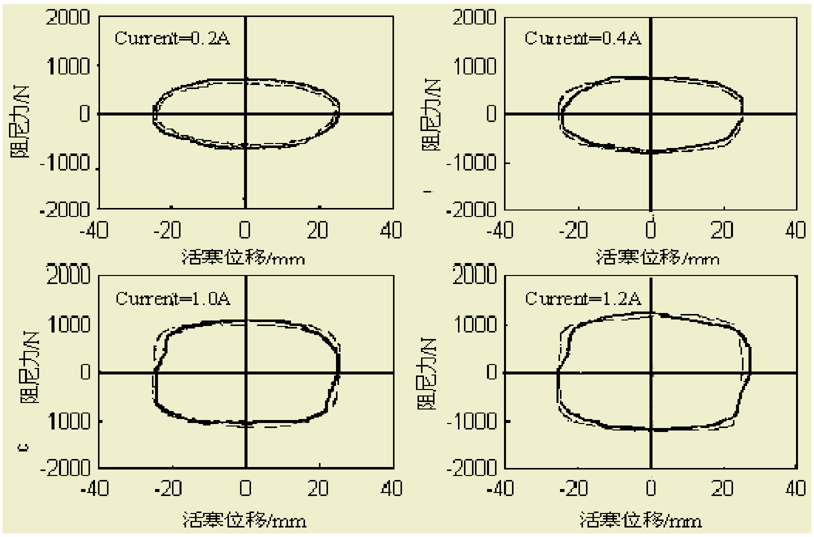


Fig. 4. Damping force versus piston displacement: neural network test.

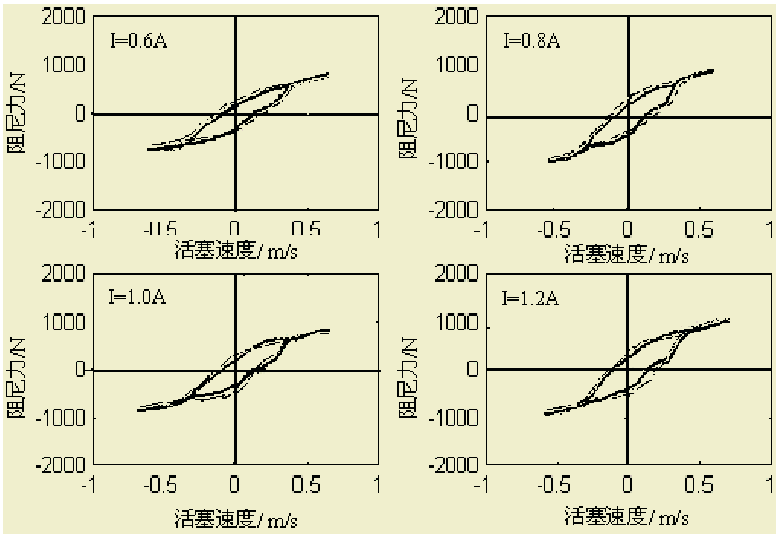


Fig. 5. Damping force versus piston velocity: Neural network: test.

5 Conclusions

This paper presents a recurrent neural network with 3 input neurons, 1 output neuron and 5 recurrent neurons in the hidden layer to simulate the dynamic behavior of MR fluid shock absorbers to develop control algorithms for suspension systems. The recurrent neural network model has been trained by the training algorithm presented by authors using the data generated from test in laboratory. Some conclusions reached from the investigations have been summarized as follows: The recurrent neural network model with 3 input neurons, 1 output neuron and 5 recurrent neurons in the hidden layer is efficient and reliable for simulation of behaviors for MR shock absorber. The training algorithm presented by authors is convergent for the recurrent neural network to be trained so long as the leaning-rate parameter can be satisfied.

References

1. Gavin, H.P., Hanson, R.D., Filisko, F.E.: Electrorheological Dampers. Part 2: Testing And Modeling, J. Applied Mech., ASME, **63** (1996) 676-682
2. Chang, C.C., Roschke, P.: Neural Network Modeling of Magnetorheological Damper. J. Intelligent Material System and structures, **9** (1998) 755-764
3. Spencer, B.F., Carson, J.D., et al.: Phenomenological Model of Magnetorheological Damper. J. Eng. Mech., ASME, **123** (1997) 230-238
4. Wereley, N.M., Pang, L., Kamath, G.M.: Idealized Hysteresis Modeling of Electrorheological and Magnetorheological Dampers. J. Intelligent Material System and structures, **9** (1998) 642-649
5. Ku, C.C., Lee, K.Y.: Diagonal Recurrent Neural Network for Dynamic Systems Control. J. IEEE Trans on NN, **6** (1995) 144-155

Geometrical Error Compensation of Gantry Stage Using Neural Networks

Kok Kiong Tan, Sunan Huang, V. Prahlad, and Tong Heng Lee

National University of Singapore, 4 Engineering Drive 3, Singapore 117576
{eletankk,elehsn,eleleeth}@nus.edu.sg

Abstract. This paper presents some results on geometrical error compensation using multilayer neural networks (NNs). It is the objective to attain higher compensation performance with less or comparable memory, using this approach. There are three main contributions. First, multilayer NNs are used to approximate the components of geometrical errors. This results in a significantly less number of neurons compared to the use of radial basis functions (RBFs). Secondly, the direction of motion is considered in the compensator. This is important as the geometrical errors can be quite distinct depending on the direction of motion due to backlash and other nonlinearities in the servo systems. Thirdly, the Abbe error is explicitly addressed in the compensator.

1 Introduction

Since the 1970s, software-based error compensation schemes have blossomed (see [2, 3]) in the area of precision machines. Common to all these works and more is a model of the machine errors ([4],[5],[7],[8],[9]), which is either implicitly or explicitly used in the compensator. In CMMs, the error model is normally used off-line to analyze and correct the measurement data in the final displayed look-up table form.

In this paper, a method for error compensation using the multilayer NNs will be presented. It is the objective to attain a good compensation performance with modest memory, using this approach. The individual geometrical error component will be decomposed into forward and reverse ones, when they are significantly different. Real machine measurements to be provided in this paper will strongly reinforced this necessity. The relative increase in memory due to the additional NNs is offset by the reduction in neuron numbers from the use of multilayer NNs. The Abbe error, which is not considered in [1], is also included in the proposed error compensation. Finally, the overall geometrical error is computed from these NNs based on a kinetic equation for the machine in point. The proposed method is applied to an XY table, and the evaluation tests show that the overall errors can be reduced to about $96\mu m$ from $324\mu m$.

2 Geometrical Error Modeling Using NNs

Once all the geometrical error components are measured and available, an overall error model can be used to yield the overall positional error. This model will consider the systematic geometric errors in the machine, and to simplify the model, the rigid-body assumption will be used. They are associated respectively with the table $(0, X, Y)$, the bridge $(0_1, X_1, Y_1)$, and the carriage $(0_2, X_2, Y_2)$. It will be assumed that, initially, all three origins coincide and the axes of all three systems are aligned.

Based on the 2D system described in [1], a volumetric error model can be derived which is given by

$$\varDelta x = \delta_x(x) + \delta_x(y) - y_p(\epsilon_x + \epsilon_y) + x_p, \tag{1}$$
$$\varDelta y = \delta_y(x) + \delta_y(y) + x\epsilon_y(x) - x\alpha + x_p(\epsilon_y + \epsilon_x) + y_p, \tag{2}$$

where x, y are the nominal positions; x_p, y_p represent the offsets of the tool tip (Abbe error); $\delta_u(v)$ is the translational error along the u-direction under motion in the v direction; ϵ_u is the rotation of the u axis; and α represents the out-of-squareness error.

It should be noted that the error sources are all calibrated using only appropriate combinations of linear displacement measurements. Since each error component varies with displacement in a nonlinear manner, it is more naturally inclined to represent the nonlinear profile using a nonlinear function compared to using a look-up table. The NNs are general tools for modeling nonlinear functions, since they can approximate any nonlinear function to any desired level of accuracy. Given $X \in R^N$, a multilayer NN has a net output given by $Y = f(X; \mathcal{W})$ with X is the NN input, Y is the NN output, and $\mathcal{W}$ is the NN weights. The activation function in the NN is selected differently in different applications. It is usually desirable to adapt the weights and thresholds of the NN off-line or on-line in real-time to achieve the required approximation performance of the net. That is, the NN should exhibit a "learning behavior". Various NN training algorithms can be found in Chapters 4 and 5 of [11].

For a XY table, six error sources comprising of the linear, angular, straightness and squareness errors, have to be approximated using NNs. The causes for these errors have been discussed in [1]. In the ensuing subsections, the results of error approximation using NNs will be shown.

Linear errors are the translational errors of carriage along their axes (X or Y) of motion. For each axis, the measurements must be carried out with the carriage moving in both the forward and reverse directions to determine the error associated with each direction. The difference in the errors for the forward and reverse runs are rather insignificant in this case. Thus, for this axis, the forward and reverse errors are simply averaged. A model of the linear error is thus available as $\delta_x(x) = f_{lin,x}(x; \mathcal{W}^*_{linx})$, where x_i is the input nominal distance along the X-axis, $f_{linx}(\cdot)$ represents the NN network, and $\mathcal{W}^*_{linx}$ is a set of the weighting values of the trained NN. The results are shown in Figure 1, where the output values measured using the laser system are plotted against the output values predicted by the NN.

For the Y-axis, the errors of both the forward and reverse runs are considered using separate NNs since the difference is large. Similar to the above procedure, the following NN models are obtained: $\delta_y^{[1]}(y) = f^{[1]}_{lin,y}(y; \mathcal{W}^*_{liny})$, $\delta_y^{[2]}(y) = f^{[2]}_{lin,y}(y; \mathcal{W}^*_{liny})$, where the symbols $^{[1]}$ and $^{[2]}$ are used to represent the separate error models corresponding to the forward and reverse runs, respectively. A comparison between the NN output and actual error measurements is given in Figure 1.

The three rotational motion errors are referred to as yaw, pitch, and roll errors. For a XY table, there is no Z direction to be considered. Thus, pitch and roll errors are not

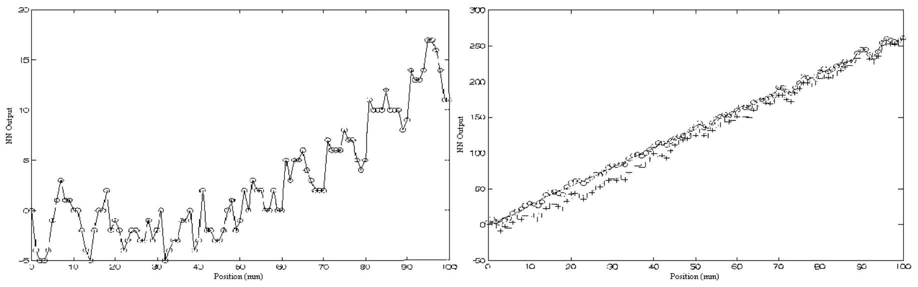


Fig. 1. NN approximation of the linear errors (X-axis, left): solid line is NN approximation and the circles represent the measured data; NN approximation of linear errors (Y-axis, right): solid line is NN approximation and circle represents the measured data (forward run); dotted line is NN approximation and plus line represents the measured data (reverse run)

necessary to be measured for the calibration. Only the yaw errors are measured along the XY table. The NN approximations of the yaw errors can be expressed as:

$$\epsilon_x^{[1]} = f_{yaw,x}^{[1]}(x; \mathcal{W}_{yawx}^*), \tag{3}$$
$$\epsilon_x^{[2]} = f_{yaw,x}^{[2]}(x; \mathcal{W}_{yawx}^*), \tag{4}$$
$$\epsilon_y^{[1]} = f_{yaw,y}^{[1]}(y; \mathcal{W}_{yawy}^*), \tag{5}$$
$$\epsilon_y^{[2]} = f_{yaw,y}^{[2]}(y; \mathcal{W}_{yawy}^*). \tag{6}$$

Straightness error measurements are derived from the perpendicular deviations from a reference straight line. For the XY table, there are two straightness error components to be determined: straightness of the X axis which is concerned with deviation along the Y-axis, and the straightness of the Y axis which is concerned with deviation along the X-axis. The corresponding NN-based models for straightness are respectively:

$$\delta_y^{[1]}(x) = f_{str,x}^{[1]}(x; \mathcal{W}_{strx}^*), \tag{7}$$
$$\delta_y^{[2]}(x) = f_{str,x}^{[2]}(x; \mathcal{W}_{strx}^*), \tag{8}$$
$$\delta_x^{[1]}(y) = f_{str,y}^{[1]}(y; \mathcal{W}_{stry}^*), \tag{9}$$
$$\delta_x^{[2]}(y) = f_{str,y}^{[2]}(y; \mathcal{W}_{stry}^*). \tag{10}$$

Squareness between 2 axes characterizes how far off from a 90 degree orientation, the 2 nominal axes are positioned relative to each other. The squareness measurement can be accomplished by performing two straightness measurements, one of which is made based on a 90 degree reference. In this experiment, Y-axis is chosen as the reference line for the squareness measurement, i.e. the straightness of X is measured with respect to Y. The squareness measurement yields a error which is single constant of 474 arcsec. Thus, the NN is not needed for use of approximating squareness error. For the calibration, $\alpha = 474 arcsec$ is substituted directly into the overall model.

3 Experimental Results

For XY table, the tool attached to the table may be moved in either the X or Y direction. The X and Y travel together to span a $100mm$ x $100mm$ 2D space. Notice that $x_p = 0, y_p = 52mm$ since there is a probe used in this experiment. The control algorithms are implemented on DS1102 DSP Controller Board from dSPACE GmbH, which utilizes TI's TMS320C31 32-bit floating point processor with $60M$Hz execution frequency. The dSPACE control development and rapid prototyping system integrate the entire development cycle seamlessly into a single environment, so that individual development stages between simulations and tests can be run and re-run without frequent re-adjustment. Define the error of X-axis: $e_x = X_d + \Delta x - x$. Notice that the PID control for X-axis is given by $u_x = K_{xp}e_x + K_{xi}\int_0^t e_x d\tau + K_{xd}\dot{e}_x$. Similarly, we define the error of Y-axis: $e_y = Y_d + \Delta y - y$ and the PID control for Y-axis is given by $u_y = K_{yp}e_y + K_{yi}\int_0^t e_y d\tau + K_{yd}\dot{e}_y$. The performance of the error compensation is tested along two diagonals of the working area $100mm \times 100mm$. This provides a fair basis to gauge the adequacy of the NN-based models. Also, this assessment follows the British Standard [10]. The linear errors are measured across the diagonals using the HP laser interferometer system. Figure 2 shows the experimental setup. For both measurements, five bi-directional runs are executed, with data points at interval $\sqrt{1^2 + 1^2} \approx 1.4142mm$. 100 points are measured of the linear displacement along the diagonal with and without the error compensation. At each point, there are 10 measurements taken from the forward and reverse runs.

For an X-Y diagonal (A-D), the total error before compensation is $166\mu m$ (see Figure 3). After compensation, it is reduced to $91\mu m$ (see Figure 3). For the other $X-Y$ diagonal (B-C), the maximum error is $324\mu m$ before compensation (see Figure 4) and $96\mu m$ after compensation (see Figure 4). Thus, on the whole, a significant improvement

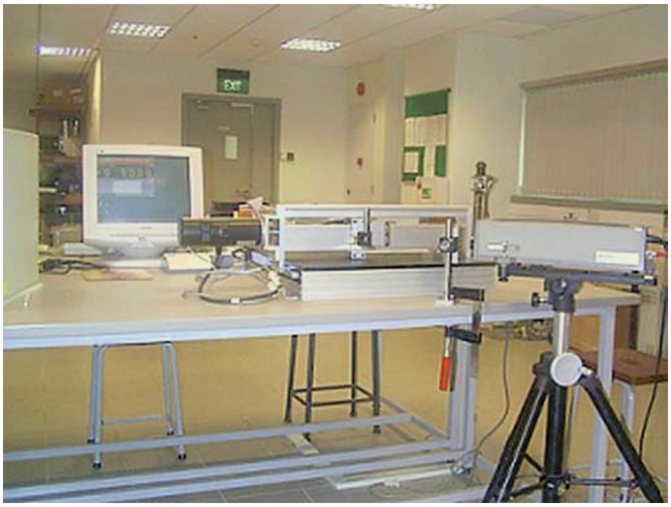


Fig. 2. Experimental set-up

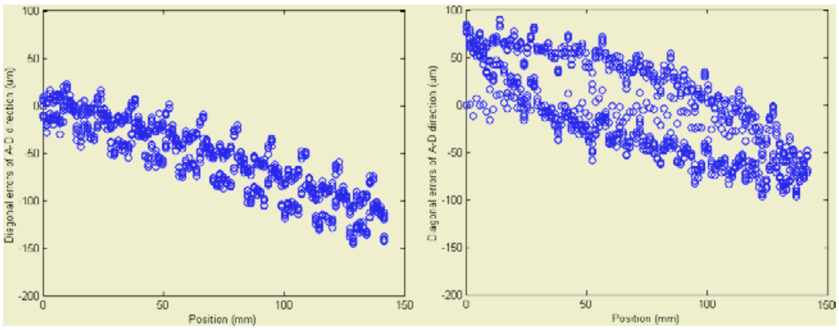


Fig. 3. Diagonal errors along A-D before compensation(left);Diagonal errors along A-D after compensation (right)

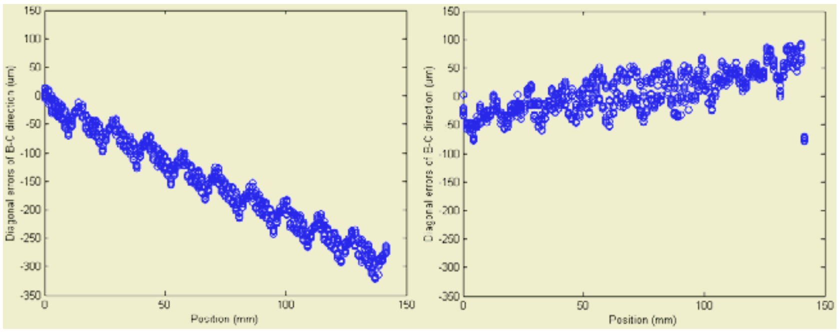


Fig. 4. Diagonal errors along B-C before compensation(left);Diagonal errors along B-C after compensation(right)

in machine accuracy is achieved with the maximum error over the working region being reduced from more $320\mu m$ to less than $100\mu m$.

For comparison, a look-up table, which is a 20×20 matrix, is used. All the error measurements are averaged from the five forward and reverse runs. For a quantitative comparison of the overall performance over the travel area, the following index can be used: $J = \frac{1}{2}\sum_{i=1}^{1000} e(i)^2$, where 1000 is obtained from 100×10 (100 points and 5 bi-directional runs). Based on this index, Table 1 compares the performance of compensators based on the look-up table and NN approximations. It is clear that the NN approximation method can provide an improved compensation.

Table 1. Quantitative performance evidence

	A-D diagonal	B-C diagonal
Look-up table	$J = 3.3113 \times 10^6$	$J = 6.6480 \times 10^5$
NN approximation	$J = 1.4772 \times 10^6$	$J = 6.1859 \times 10^5$

4 Conclusion

A geometrical compensation method using multilayers NNs has been developed and tested in the paper. It is capable of addressing the asymmetric errors from forward and reverse motion of servo systems. The Abbe error is also included in the overall compensation model. Compared to [1], the neuron number necessary is reduced, resulting in a simpler implementation. Experimental tests on an XY table verified the effectiveness of the proposed method.

References

1. Tan, K.K., Huang, S.N., Seet,H.L.: Geometrical Error Compensation of Precision Motion Systems Using Radial Basis Functions. IEEE Trans.on Instrument and Measurement. **49** (2000) 984-990
2. Love,W.J., Scarr,A.J.: The Determination of the Volumetric Accuracy of Multi Axis Machines. The 14th MTDR. (1973) 307-315
3. Bush,K., Kunzmann,H., Waldele,F.: Numerical Error Correction of Co-Ordinate Measuring Machines. Proc.Int.Symp.on Metrology for Quality Control in Production, Tokyo (1984) 270-282
4. Hayati,S.A.: Robot Arm Geometric Link Parameter Estimation. Proceedings of the 22nd IEEE Conference on Decision and Control.(1983) 1477-1483
5. Veitschnegger,W.K., Wu,C.H.: Robot Accuracy Analysis Based On Kinematics. IEEE Journal of Robotics and Automation. **RA-2/3**(1986) 171-179
6. Barron,A.R.: Approximation and Estimation Bounds for Artificial Neural Networks. Proc.4th Ann. Workshop on Computational Learning Theory, San Mateo (1991) 243-249
7. Zhang,G., Veale,R., Charlton,T.,Hocken,R.,Borchardt,B.: Error compensation of coordinate measuring machines. Annals of the CIRP, **34** (1985) 445-448
8. Duffie, N.A., Maimberg, S.J.: Error Diagnosis and Compensation Using Kinematic Models and Position Error Data. Annals of the CIRP. **36** (1987) 355-358
9. Weekers,W.G., Schellekens, P.H.J.: Assessment of Dynamic Errors Of CMMs for Fast Probing. Annals of the CIRP. **44** (1995) 469-474
10. British Standard:Coordinate Measuring Machines, Part 3. Code of practice, BS 6808,U.K. (1989)
11. Haykin, S.: Neural networks: A Comprehensive Foundation. 2nd edn, Prentice Hall (1999)

Neural Particle Swarm Optimization for Casing Damage Prediction

Quansheng Dou[1], Chunguang Zhou[1], Guanyu Pan[1], Hongwen Luo[2], and Quan Liu[1]

[1] College of Computer Science and Technology, Jilin University, Changchun, Jilin 130012, China
{li_dou,cgzhou,gypan,quanliu}@163.com
[2] College of Mathematics, Jilin University, Changchun, Jilin 130012, China
luohongwen@163.com

Abstract. Casing damage of oil wells is a serious problem, which has caused enormous economic losses to oil field of china. The reasons of casing damage are very complex, the experts of oil production can only predicted the damage trend by their personal experiences. This paper put forward a new method called Neural Particle Swarm Optimization (NPSO), which has been applied to the prediction of casing damage. In this method, combined with feedforward neural network and PSO (Particle Swarm Optimization PSO), the feedforward neural network is regarded as particles in space, called neural particles. The learning processes of neural network are take cover in the movement of neural particles following individual best and swarm best. NPSO method was used to predict casing damage of oil wells, the experimental result show that the learning efficiency and converge rate of NPSO improved much more than tradition learning method (such as BP) and has great application value for prediction of casing damage.

1 Introduction

In recent years, the casing damage of oil wells is a serious problem of chinese oil fields, which has caused enormous economic losses. A normal oil well is shows as following Figure 1.

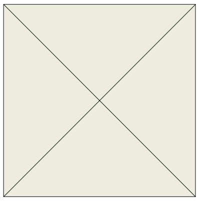


Fig. 1. Normal Oil Well.

In the course of oil production, especially in some special geological condition, original distribution of stratum stress may change, which will course casing damage of oil wells. following Figure 2 shows the structure of a casing damage well.

The reasons that may course casing damage are very complex, so oil production engineers can only predict the status of casing by their experiences. herefore, it is very difficult to predict casing damage. Moreover, there is still many problems to descript the complex physical model such as casing damage by mathematics method, furthermore, some geological parameters in actual application are unable to obtained. This paper put forward a new method to predict casing damage by NPSO (Neural Particle Swarm Optimization), it overcame the deficiency of mathematics method, which has received a good prediction result.

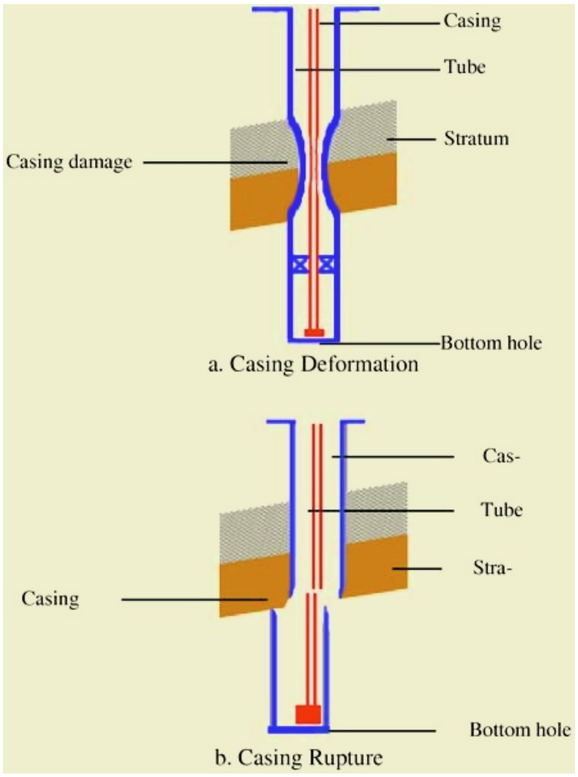


Fig. 2. Oil Well Which Casing Damaged.

2 Neural Particle Swarm Optimization

The Particle Swarm Optimization (PSO) method, which was a kind of evolutionary computing technology based on Swarm Intelligence, was originally designed by Kennedy and Eberhart in 1995[1],[2].Its original idea was from the research such as fish schooling and bird flocking. The basic idea of bird flocking can be summarized as follows: Among the birds, every bird has to search food, and everyone shares the information from other birds as well, therefore, the area that a bird will search next time is influenced by self-experience and other birds'experiences,And individuals determine how to find hopeful position at next step by the two factors above. PSO method are widely applied in function optimization, neutral network training, pattern recognition, fuzzy system control and other fields[3]-[10].

In PSO, every potential solution is defined as a "particle" in the searching space. Each particle can be denoted as (x_i, v_i, p_i), Here x_i is the particle's current position, v_i is its current velocity, p_i is the best position of the particle itself has encountered. the particle's current velocity v_i determines its "flying" direction and distance. The standard PSO method produces a particles group on S randomly, every particle adjusts its position as follows:

$$\begin{gathered} v_i = \chi(wv_i + c_1\varphi_1(p_i - x_i) + c_2\varphi_2(p_g - x_i)) \\ x_i = x_i + v_i \end{gathered} \tag{1}$$

Here x_i and v_i are position vector and velocity vector of the i^{th} particle respectively, p_i is the best position of the i^{th} particle encountered, p_g is the best position of all the particles encountered, c_1 and c_2 are positive constants, r_1 and r_2 are random numbers in the interval (0,1), ω is the inertia parameter χ is the constriction factor.

The theory research of neural network system has got a great development, the development has important influence on a lot of fields, such as computer science, artificial intelligence, etc. Multi-layer feedforward neural network is the most widely used neural network system in fields of associative memory, predicting and optimizing, etc. because it has a simple structure and is easy to realize. In fact, feedforward neural network realize a nonlinear mapping between network input to output. Feedforward neural network is regarded as particles in the Neural Particle swarm, called neural particles, expressed by NP. Each neural particle's input-output-number of hidden layers and neuron's number of each layer are fixed. On the assumption that the total number of training sample is M, T_i is teacher signal of training sample i i=1, …, M. Each neural particle will be evaluate by formula (2) as following:

$$F(NP) = \sum_{i=1}^{M} \left| O(NP) - T_i \right| \tag{2}$$

Different from tradition learning method, the learning processes of neural network are take cover in the movement of neural particles following individual best and swarm best in NPSO. The detailed description of NPSO is as follows:

Algorithm 1. Neural Particle Swarm Optimization

Step1. Initialize neural particle swarm *POP*, assume that the number of neural particles in *POP* is *N*, determine the neural particles' input-number of neural cells-activation function and so on, produce a group of right value randomly for each neural particle in the fixed range;

Step2. Base on a group of training samples and teacher signals, to evaluate each neural particle NP_j j = 1, 2, …, N by equation (2), record the best position that a particle j has encountered as P_NP_j, and G_NP of the whole swarm;

Step3. Update each particle's next position by equation (1);

Step4. Select the neural particles of best fitness value if terminal condition is satisfied, otherwise turn to Step2;

There are two main problems that restrict feedforward neural network as follows: i. The learning algorithm of feedforward neural network converges slowly with a low study efficiency. ii.The feedforward neural network is very sensitive to initial value , an improper initial value can make the network diverge. It can be seen from algorithm 1, the particles in NPSO method determine their positions by self-experience and group experience and easy to realize, so the efficiency improves a lot. Moreover, particles distribute in the whole space and avoid the influence of initial value on the network astringency.

To predict oil well casing damage by NPSO method which received a good result.

3 Predict Casing Damage by NPSO

There have been found that more than two thousand oil wells are casing damage in an oilfield of china since 1980s, the number of casing damage accounts for 1/4 of the total amount. The experts has carried on an initial analysis, the casing damage of this oilfield is caused by the fluctuation of production wells' pressure in a certain geological condition.

We collected the ralated data of casing damage and nomal production wells in this oilfiled, and set the casing's working life, casing pressure,and tube pressure as the input of neural particles, set the current casing damage status as teacher signal., select sigmoid function as neuron's activation function Set the particle swarm's parameter, set the population size N=40, w=0.5, c_1=1.4 and c_2=1.4. Run NPSO method and BP method 40 times seperately, the converge rate of both is as following table1:

Table 1. Comparison of Converge rate Between NPSO and BP.

Structure of	Converge rate (%)	
Neural network	NPSO	BP
6-6-6-1	90	87.5
6-8-8-1	82.5	80
6-10-10-1	80	72.5
6-12-12-1	70	60
6-14-14-1	72.5	60

From table1, the converging proportion of neural network became lower and lower when the structure of neural network became complex. (Here 6-6-6-1 means that there are four layers in network and the neuron's number of each layer are 6, 6, 6 and 1 respectively) the converging proportion of NPSO are better than it of BP. at the Same time , the learning efficiency of NPSO is also better than BP method because that there are no complex mathematical computing in NPSO.

The networks were be trained by NPSO and BP respectively, the two hundred wells of above oilfield were be predicted, the experimental result as following table2.

Table 2. Comparison of casing damage identify rate between NPSO and BP.

Structure of	Casing Damage Identify Rate (%)	
Neural network	NPSO	BP
6-6-6-1	61.5	59.5
6-8-8-1	66	62
6-10-10-1	71.5	65
6-12-12-1	78	71
6-14-14-1	78.5	72

From table2, we can see that the identify ability of neural network which trained by NPSO was better than it which trained by BP, in other words, the error of NPSO was less than BP's. So the conclusion can be drawn that NPSO can break away from the attraction of local optimum and approach the global optimum accurately.

4 Conclusions

This paper propose neural particle swarm optimization method (NPSO) in which the feedforward neural network was regard as a particle in space. The learning processes of neural network are take cover in the movement of neural particles following individual best and swarm best. NPSO is easy to realize compare with BP. The experimental result show that not only converge rate but also identify rate of NPSO are better than BP's. NPSO was used to predict casing damager of oil wells, establish a nonliner mapping between the production parameters of oil wells and casing status, this mapping is very difficult to realize by mathematical method. At the same time casing damage identify rate of NPSO is satisfied, so NPSO has great application value for casing damage prediction.

References

1. Eberhart, R.C., Kennedy, J.: A New Optimizer Using Particle Swarm Theory. Proceedings of the Sixth International Symposium on Micro Machine and Human Science, Nagoya Japan (1995) 39-43
2. Kennedy, J., Eberhart, R.C.: Particle Swarm Optimization. Proc IEEE International Conference on Neural Networks. IEEE Service Center, Piscataway, NJ, IV (1995)1942-1948
3. Hu, X., and Eberhart, R.C.: Adaptive Particle Swarm Optimization: Detection and Response to Dynamic Systems. Computational Intelligence, Hawaii, IEEE Press (2002)
4. Salman, A.: Discrete Particle Swarm Optimization for Heterogeneous Task Assignment Problem. Proceedings of World Multiconference on Systemics, Cybernetics and Informatics Cybernetics and Informatics (2001)
5. Krink, T., Vesterstrom, J.S., and Riget, J.: Particle Swarm Optimization with Spatial Particle Extension. Proceedings of the IEEE Congress on Evolutionary Computation Honolulu, Hawaii USA (2002)
6. Hirotaka, Yoshida and Kenichi: A particle Swarm Optimization for Reactive Power and Voltage Control Considering Voltage Stability. IEEE International Conference on Intelligent System Applications to Power Systems Rio de Janeiro (1999)
7. Voss, M.S., and Xin Feng, X.: Arma Model Selection Using Particle Swarm Optimization and Aic Criteria. 15th Triennial World Congress, Barcelona , Spain (2002)
8. Clerc, M., and Kennedy, J.: The Particle Swarm: Explosion, Stability, and Convergence in a Multi-Dimensional Complex Space. IEEE Transactions on Evolutionary Computation (2002) 58-73
9. Parsopoulos, K.E., Vrahatis, M.N.: Particle Swarm Optimization Method in Multiobjective Problems. Proceedings of the 2002 ACM Symposium on Applied Computing (2002) 603-607
10. Van den Bergh, F., Engelbrecht, A.P: Cooperative Learning in Neural Networks Using Particle Swarm Optimizers. South Africam Computer (2000) 84-90

A Novel Chamber Scheduling Method in Etching Tools Using Adaptive Neural Networks*

Hua Xu[1], Peifa Jia[1], and Xuegong Zhang[2]

[1] State Key Laboratory of Intelligent Technology and Systems, Tsinghua University, Beijing 100084, China
{Xu-h,dcstjpf}@tsinghua.edu.cn
[2] Department of Automation, Tsinghua University, Beijing 100084 , China
zhangxg@tsinghua.edu.cn

Abstract. Chamber scheduling in etching tools is an important but difficult task in integrated circuit manufacturing. In order to effectively solve such combinatorial optimization problems in etching tools, this paper presents a novel chamber scheduling approach on the base of Adaptive Artificial Neural Networks (ANNs). Feed forward, multi-layered neural network meta-models were trained through the back-error-propagation (BEP) learning algorithm to provide a versatile job-shop scheduling analysis framework. At the same time, an adaptive selection mechanism has been extended into ANN. By testing the practical data set, the method is able to provide near-optimal solutions for practical chamber scheduling problems, and the results are superior to those generated by what have been reported in the neural network scheduling literature.

1 Introduction

Chamber scheduling is a major issue faced by all the control of etching tools [1]. Corresponding to the general scheduling definition [2], [3], the chamber scheduling is to assign chambers to different wafers in order to insure the termination of these etching tasks in a reasonable amount of time and to derive the benefits from effective scheduling. The chamber scheduling problem essentially is a job shop problem [4] with small size. The general problem is to find a sequence in which the jobs (e.g. etching tasks) pass through the resources (e.g. chambers), which constitutes a feasible and optimal schedule with respect to some specific performance criterion [5].

Traditionally, there are three kinds of approaches to solve job-shop scheduling problems: priority rules, combinatorial optimization and constraints analysis [6], [7], [8]. However, it has been demonstrated [9] that job-shop scheduling is usually an NP-complete (nondeterministic polynomial time complete) problem. So, researchers turned to search its near-optimal solutions with all kinds of heuristic algorithms [10], especially Hopfield networks [10]-[15]. In order to suit the change adaptively, Zhang [16] has presented an approach based on neural networks, which has also been proved to converge to local minimum. In this paper, a chamber scheduling system is proposed on the base of Zhang's neural network method [16]. It is used to schedule different chambers to etch wafers according to the processing requirements.

* This work was jointly supported by the National Nature Science Foundation for Youth Fund (Grant No: 60405011), the China Postdoctoral Foundation for China Postdoctoral Science Fund (Grant No: 20040350078) and the National 863 Project of China.

This paper is organized as the following. In section 2, the structure of chamber scheduling system is explained in details. Section 3 presents the intelligent chamber scheduling tool. Section 4 analyzes the experimental scheduling results. Finally, the conclusion is summarized in section 5.

2 Architectures of Chamber Scheduling Module

Chamber scheduling module (CSM), designed to generate the chamber scheduling sequences in etching tools, has several tasks to resolve as the following:

(1) Extracting the processing features of every wafer in etching tools;
(2) Representing the wafer processing requirements and chamber features;
(3) Scheduling process;
(4) Learning/adjusting process.

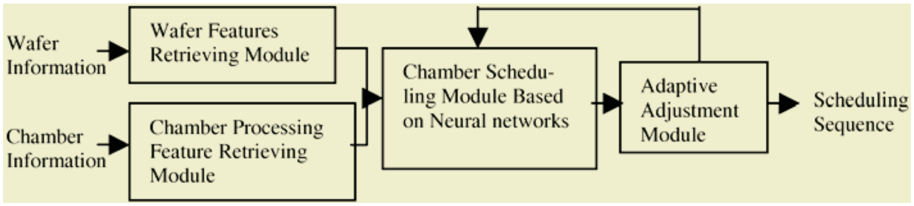

Wafer Information
Wafer Features Retrieving Module
Chamber Information
Chamber Processing Feature Retrieving Module
Chamber Scheduling Module Based on Neural networks
Adaptive Adjustment Module
Scheduling Sequence

Fig. 1. The architecture of CSM is composed of *four* modules: (1) a wafer features retrieving module; (2) a chamber processing feature retrieving module; (3) a chamber scheduling module based on neural networks; (4) an adaptive neural network adjustment module.

3 Chamber Scheduling

In CSM, the wafer information should be input to the wafer processing features retrieving module. Also is the chamber information input to the chamber feature retrieving module. The results of the two modules are as the input of the chamber scheduling module. The scheduling results are compared and selected by the adaptive adjustment module. Finally, the scheduling sequence and execution time can be got from the final results.

3.1 Wafer Features Retrieving Module

In etching tools, the wafers to be processed are stored in the cassette. The maximum number of wafers in the cassette is 22. In each cassette, the wafer number may be different. Also are the processing requirements various respectively. So the wafer processing feature retrieving module is designed to get them.

The wafer features retrieving module completes the following tasks. It counts the exact wafer number in cassette. In our approach, the wafer number is registered as N. According to the processing requirements of every wafer, the number of processing steps of every wafer can be got according to the technical processing requirements. In our approach, the number of processing steps of the i^{th} wafer is k_i.

3.2 Chamber Processing Feature Retrieving Module

In the integrated circuit processing procedures, every wafer in one cassette may be required to be processed differently at some time. So they have various technical processing requirements. And the processing conditions are various, which include the processing beginning time of the kth operation of the ith wafer - s_{ik} and the processing duration time of the kth operation - t_{ik}.

In our approach, a chamber processing feature retrieving module is designed to get the processing features of every wafer in the cassette according to the processing requirements and the characters of processing chamber. The beginning time "s_{ik}" and the operation duration time "t_{ik}" satisfy the following conditions.

$$s_{ik}-s_{ik+1}+t_{ik} \leqslant 0,\ \ i=1\ldots,N\ ;\ k=1,\ldots,k_i-1 \tag{1}$$

$$s_{i1} \geqslant 0\ ,\qquad i=1,\ldots,N \tag{2}$$

$$s_{ik}-s_{jp}+t_{ik} \leqslant 0 \text{ or } s_{jp}-s_{ik}+t_{ik} \leqslant 0 \tag{3}$$

where s_{ik} and t_{ik} are the processing beginning time and the duration time of the kth operation of the ith wafer in the cassette. N is the number of wafers in the cassette. k_i is the last operation of the wafer.

The formula (1) restricts that any processing step of any wafer can not start until the former step completes. The formula (2) requires that the beginning time of every processing procedure be greater than 0. The formula (3) describes the relation between the two operations: O_{ik} and O_{jp} on the same chamber. It needs that one chamber should conduct only one operation at any time. That is to say, one operation can be done in any chamber before or after another one.

3.3 Chamber Scheduling Module Based on Neural Network

A chamber scheduling module based on neural network is designed. Its function is to schedule the chambers in the etching machine according to the wafer processing features and chamber characters.

Suppose that $\Sigma_{i=1}^{N} s_{i,ki}$ is the sum of the last operation beginning time of all the wafers. The chamber scheduling problem is to find the minimum value of $\Sigma_{i=1}^{N} s_{i,ki}$. Then neural networks are used to get the minimum value.

In our neural network model, s_{ik} represents the neuron. Then the energy function is defined as the following.

$$\begin{aligned} E = \sum_i s_{ik_i} \ &+\ \sum_i\sum_k H_1 \bullet F_1(s_{ik}-s_{ik+1}+t_{ik}) \ +\ \sum_i H_2 \bullet F(-s_{i1}) \\ &+\ \sum\sum H_3 \bullet \min(F(s_{ik}-s_{jp}+t_{ik}),F(s_{jp}-s_{ik}+t_{ik})) \end{aligned} \tag{4}$$

where H_1, H_2 and H_3 are the weight coefficients and

$$F(x) = \begin{cases} e^x & x > 0 \\ 0 & else \end{cases} \tag{5}$$

In our neural network, the neurons use the following learning algorithm.

$$s_{ik}(t+1) = \begin{cases} s_{ik}(t) - \Delta, & \dfrac{\partial E}{\partial s_{ik}} > 0 \\ s_{ik}(t) + \Delta, & \dfrac{\partial E}{\partial s_{ik}} < 0 \\ s_{ik}(t), & \dfrac{\partial E}{\partial s_{ik}} = 0 \end{cases} \tag{6}$$

where E is the energy function and Δ is a positive constant. The above algorithm has been proved that it converges to the near-optimal value [16]. To get much better optimal or near-optimal results, an adaptive adjustment module is designed.

3.4 Adaptive Adjustment Module

An adaptive adjustment module is designed to find much better near-optimal value. It compares and selects the results from the chamber scheduling module as the following. After it gets the results calculated from the scheduling module, it assigns the beginning time of the next operation with the ending time of current completed operation. And the neural network will evolve in a new result space. When the neural network is stable, a new result can be got, and the better one among these results and the former are preserved. This procedure will not stop until no better results can be got. The detailed steps can be defined formally as the following.

Table 1. The Adaptive Algorithm in the Adaptive Adjustment Module.

```
Step 1: Using the scheduling module, a scheduling result and a total operation time "total-time" can be got.
Step 2: Select the neuron $s_{jp}$, whose state will change currently.
Step 3: Set $s_{jp}=s_{jp-1}+t_{jp-1}$.
Step 4: Using the scheduling module, a new result can be got.
Step 5: Calculate the new total operation time "new-total-time".
Step 6: If new-total-time < total-time, then
        (1) replace the current result with the new one;
        (2) Select the next neuron as the state changing one $s_{jp}$.
        (3) Go to step 3.
Step 7: If it doesn't end, select the next neuron as the state changing one $s_{jp}$ and go to step 3.
Step 8: end.
```

In step 7, the ending condition is that no new better results can be got for N times. N is the number of the neurons in the neural network. Because the minimum value exists, the time t exists, after which no new better results can be got.

4 Experiments

In real etching tools of integrated circuit factory, when the wafers in one cassette arrive to the platform, the controller should firstly plan the chamber schedule to use the chambers in the etching tool effectively according to the wafer processing requirements. In our etching tool, the scheduling method discussed above is used in scheduling the chambers. It will schedule the chambers to process the wafers in the cassette according to some scheduling sequence. In the general etching tool of 100 nm, the etching process steps include the following ones: Aligning, Processing and Cooling. Fig.2 has depicted the system structure of our etching tool. Chamber 1 and 7 are load lots respectively, through which the wafers are transferred to or from the etching tool. Chamber 2 is the aligner. And chamber 3, 4 and 5 are processing modules. Chamber 6 is cooler in the etching tool. In the etching process, the aligning procedure may last for about 50s. The number of the wafers in the cassette is 10. For the processing procedure, one of them (No.1) needs 200s. Two of them (NO.2 and 3) need 100s. The others (No.4-10) need 50s. Finally, the cooling procedure may last for about 50s. Our scheduling algorithm is used. One of the scheduling sequences is depicted in Fig.3. The minimum processing/execution time got by the method is 700s.

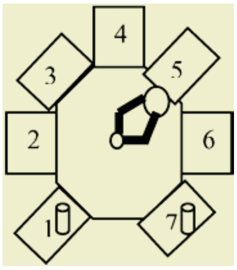


Fig. 2. The Structure of Etching Tool 1, 7: Load Lot; 2: Aligner; 3, 4, 5: Processing Chambers; 6: Cooler.

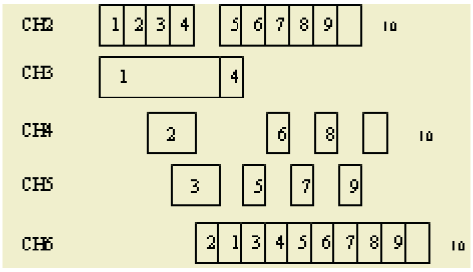


Fig. 3. Gant Figure of the Scheduling.

5 Conclusions

Chamber scheduling is an important problem in the control of etching tools. This paper presents a neural network based method to complete chamber scheduling. In our approach, wafer features and chamber processing features are firstly retrieved respectively by Wafer Features Retrieving Module and Chamber Processing Feature Retrieving Module. Then, as the input, these feature values are processed in a BP neural network in CSM so as to get the chamber scheduling results, which has been proved to converge to near optimization value. An adaptive adjustment module is also used to revise the results. The validity of this method has been demonstrated by using it in the realistic chamber scheduling.

References

1. LAM Research Company: LAM 9400 Handbook, California (2001)
2. Fonseca, D.J., Navaresse, D.: Artificial Neural Networks for Job Shop Simulation. Advanced Engineering Informatics, **16** (2002) 241-246
3. Dempster, M., Lenstra, J., Kan, R.: Deterministic and Stochastic Scheduling: Introduction. Proceedings of the NATO Advanced Study and Research Institute on Theoretical Approaches to Scheduling Problems, **1** (1981) 3-14
4. Chenga, R., Genb, M., Tsujimura, Y.: A Tutorial Survey of Job-shop Scheduling Problems Using Genetic Algorithms, Part II: Hybrid Genetic Search Strategies. Computers & Industrial Engineering, **36** (1999) 343-364
5. Jones, A., Rabelo, L.: Survey of Job Shop Scheduling Techniques. Gaithersburg, MD: NISTR, National Institute of Standards and Technology, Gaithersburg (1998)
6. Shengxiang, Y., Dingwei W.: A New Adaptive Neural Network and Heuristics Hybrid Approach for Job-shop Scheduling. Computers & Operations Research, **28** (2001) 955-971

7. Dubois, D., Fargier, H., Prade, H.: Fuzzy Constraints in Job Shop Scheduling. Journal of Intelligent Manufacturing, **6** (1995) 215-234
8. Montazeri H.: Analysis of Scheduling Rules for an FMS. International Journal of Production Research, **28** (1990) 785-802
9. Garey, M.R., Johnson, D.S.; Sethi, R.: The Complexity of Flow Shop and Job-shop Scheduling. Math. Opl. Res., **1** (1976) 117-129
10. French, S.: Sequencing and Scheduling: An Introduction to the Mathematics of the Job-shop. Wiley, New York (1982)
11. Foo, S.Y., Takefuji, Y.: Integer Linear Programming Neural Networks for Job-shop Scheduling. Proceedings of IEEE IJCNN, San Diego, **2** (1988) 341-348
12. Foo, S.Y., Takefuji, Y., Szu, H.: Job-shop Scheduling Based on Modeled Tank-Hopfield Linear Programming Networks. Engineering Application and Artificial Intelligent, **7** (1994) 321-327
13. Zhou, D.N., Charkassky, V., Baldwin, T.R., Hong, D.W.: Scaling Neural Network for Job-Shop Scheduling. Proceedings of IEEE International Joint Conference on Neural Networks, New York, **3** (1989) 889-894
14. Willems, T.M., Brandts, L.: Implementing Heuristics as an Optimization Criterion in Neural Networks for Job-shop Scheduling. Journal of Intelligent Manufacturing, **6** (1995) 377-387
15. Yang, S., Wang, D.: Constraint Satisfaction Adaptive Neural Network and Heuristics Combined Approaches for Generalized Job-shop Scheduling. IEEE Transactions on Neural Networks, **11** (2000) 474-486
16. Zhang, C.S., Yan, P.F.: Neural Network Method of Solving Job-shop Scheduling Problem, Acta Automatic Sinica, **21** (1995) 760-712

CFNN Without Normalization-Based Acetone Product Quality Prediction

Jiao Wang and Xiong Wang

Department of Automation, Tsinghua University, Beijing 100084, China
wangjiao02@mails.tsinghua.edu.cn,
wx@mail.au.tsinghua.edu.cn

Abstract. This paper presents a kind of model based on compensatory fuzzy neural network (CFNN) without normalization to predict product quality in the acetone refining process. Important technological influence factors are selected according to the analysis results of several variables selection methods. Using the selected factors as the input variables of the network, a product quality prediction model is constructed. Experiment results show that the trained model achieves good effects, and has more advantages in convergence speed and error precision compared with CFNN with normalization.

1 Introduction

Acetone refining is a complicated chemical process, which refines rough acetone to obtain acetone product. The factors that influence acetone product quality not only are relevant to material ingredients but also have close relations with technological conditions. It has great significance to predict product quality according to the data of technological influence factors in the acetone refining process. Purity is the primary index of acetone product quality. Since the relation between technological factors and acetone purity is highly nonlinear, uncertain and fuzzy, the prediction results using conventional methods are discrepant with real data.

Zhang and Mendel[1] proposed a new kind of fuzzy neural network using compensatory fuzzy operators. CFNN is a hybrid system integrating compensatory fuzzy logic and neural network, so it fully utilizes the knowledge expression ability of fuzzy system and strong self-adaptive ability of neural network. Because CFNN adopts compensatory fuzzy operators to perform fuzzy reasoning, the network can not only adaptively adjust input and output fuzzy membership functions, but also dynamically optimize fuzzy reasoning. This strategy makes the network more error-tolerant and the system more stable, and CFNN has applied in many fields [2],[3]. The researches of Seker et al [3],[4] have shown that CFNN without normalization converges faster than CFNN with normalization. In this paper, variables selection methods are used to select important ones from the technological influence factors in the acetone refining process. Product quality prediction model base on CFNN without normalization is constructed with input variables as the selected factors. Experiment results show that good effects are achieved, and training epochs and prediction precision have distinct improvement compared with CFNN with normalization.

2 Architecture of CFNN

Figure 1 shows the architecture of an *n*-input-one-output CFNN, which has five layers: input layer, fuzzification layer, pessimistic-optimistic operation layer, compensatory operation layer and defuzzification layer [1]. There are *n* nodes in the input layer directly connected to *n* input variables, *m* pairs of nodes in the pessimistic-optimistic operation layer and *m* nodes in the compensatory operation layer corresponding to *m* rules, and $n\times m$ nodes in the fuzzification layer. The structure of CFNN is determined by the number of input variables *n* and rules *m*.

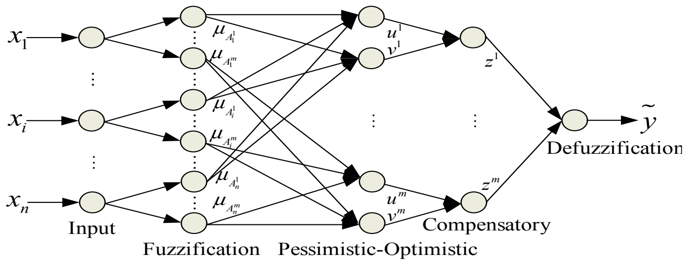


Fig. 1. The Architecture of CFNN.

The *m* fuzzy IF-THEN rules of CFNN are of the following form:

$$\mathrm{FR}^{(k)}: \text{ IF } x_1 \textit{ is } A_1^k \textit{ and } \cdots \textit{ and } x_n \textit{ is } A_n^k, \quad \text{THEN } \ y \textit{ is } B^k . \tag{1}$$

where A_i^k and B^k are input and output fuzzy sets, respectively, i=1,2,⋯,n, k=1,2,⋯,m. The fuzzy membership functions of A_i^k and B^k are defined as

$$\mu_{A_i^k}(x_i) = \exp[-(\frac{x_i - a_i^k}{\sigma_i^k})^2], \quad \mu_{B^k}(y) = \exp[-(\frac{y-b^k}{\delta^k})^2] \cdot \tag{2}$$

where a_i^k and σ_i^k are the centers and widths of input membership functions; $b^k$and δ^k are the center and width of the output membership function.

Given an *n*-dimensional input vectors $x^p = (x_1^p, x_2^p, \cdots, x_n^p)$ and the corresponding one-dimensional target output data y^p, p=1,2,⋯,N, each node in the fuzzification layer calculates the input membership degree $\mu_{A_i^k}(x_i^p)$ for the *i*th feature of the *p*th input datum in rule *k* according to (2). Each pair of nodes in the pessimistic-optimistic operation layer performs pessimistic operation (3) and optimistic operation (4), respectively.

$$u^k = \prod_{i=1}^{n} \mu_{A_i^k}(x_i^p) . \tag{3}$$

$$v^k = \left[\prod_{i=1}^{n} \mu_{A_i^k}(x_i^p) \right]^{\frac{1}{n}} . \tag{4}$$

And each node in the compensatory operation layer performs fuzzy reasoning by using (5)

$$z^k = (u^k)^{1-\gamma^k}(v^k)^{\gamma^k} = \left[\prod_{i=1}^{n}\mu_{A_i^k}(x_i^p)\right]^{1-\gamma^k+\frac{\gamma^k}{n}}. \tag{5}$$

where $\gamma^k \in [0,1]$ is the compensatory degree of rule k. Improved centroidal defuzzification is used to obtain the accurate output of CFNN with normalization [2]

$$f(x) = \frac{\sum_{k=1}^{m} b^k \delta^k z^k}{\sum_{k=1}^{m} \delta^k z^k}. \tag{6}$$

FNN without normalization converges faster than that with normalization [5], the output of CFNN without normalization is defined as [3],[4]

$$f(x) = \sum_{k=1}^{m} b^k \delta^k z^k. \tag{7}$$

The parameters of CFNN can be optimally adjusted using BP learning algorithm. Define the index function as

$$E^p = \frac{1}{2}\left[f(x^p) - y^p\right]^2. \tag{8}$$

where $f(x)^p$ and y^p are the output of CFNN and the target output of the pth datum. According to the gradient descent method, the centers and widths of the input and output membership functions are trained as

$$a_i^k(t+1) = a_i^k(t) - \eta\frac{\partial E^p}{\partial a_i^k}|_t, \quad o_i^k(t+1) = \sigma_i^k(t) - \eta\frac{\partial E^p}{\partial \sigma_i^k}|_t. \tag{9}$$

$$b^k(t+1) = b^k(t) - \eta\frac{\partial E^p}{\partial b^k}|_t, \quad \delta^k(t+1) = \delta^k(t) - \eta\frac{\partial E^p}{\partial \delta^k}|_t. \tag{10}$$

since $\gamma \in [0,1]$, redefine $r = \frac{c^2}{c^2+d^2}$, then

$$c(t+1) = c(t) - \eta\frac{2c(t)d^2(t)}{[c^2(t)+d^2(t)]^2}\frac{\partial E^p}{\partial \gamma}|_t. \tag{11}$$

$$d(t+1) = d(t) + \eta\frac{2c^2(t)d(t)}{[c^2(t)+d^2(t)]^2}\frac{\partial E^p}{\partial \gamma}|_t. \tag{12}$$

$$\gamma(t+1) = \frac{c^2(t+1)}{c^2(t+1)+d^2(t+1)}. \tag{13}$$

where η is the learning rate and t is training epochs, t=0,1,2,···

The learning process of CFNN can be divided into the following three steps: first, initialize the network parameters; second, for an input-output data pair, make forward compensatory fuzzy logical reasoning and calculate the output of the network; finally, back propagate the error, and adjust the network parameters.

3 Variables Selection

There are many technological influence factors in the acetone refining process. If treating all these factors as the input variables of CFNN, the network will be complicated and extra error is prone to be introduced. Therefore, variables selection is a key issue in the acetone product quality prediction. It is necessary to use variables selection methods to select important factors, which can not only ensure the validity and importance of the input variables of network, but also reduce the extra error brought by the introduction of redundant variables. Variables selection simplifies the network model under the condition of ensuring that valid variables are selected and invalid variables are removed [6].

In the product quality prediction, 400 pairs of input-output data are selected from the acetone refining process, which are described by 12 technological influence factors and acetone purity. The influence factors and their symbols are represented in table 1.

Table 1. Influence factors and the symbols.

X_1	flux of heating stream	X_2	reflux ratio
X_3	flux of circulating acetone	X_4	liquid position of tower
X_5	liquid position of reflux pot	X_6	pressure of top tower
X_7	pressure of bottom tower	X_8	temperature of top tower
X_9	temperature of 26# tower board	X_{10}	temperature of 55# tower board
X_{11}	temperature of 67# tower board	X_{12}	temperature of sensitive board

Variables selection is prepared for the consequent prediction. To eliminate the influence caused by the dimension and change range of the technological factors, raw data are normalized before variables selection. Since important factors determined by each selection method are not identical, various methods should be adopted to make comparisons [7]. In this paper, influence factors are analyzed by several methods, including correlation analysis (CA), Fisher index analysis (FIA), principal component regression (PCR) and partial least square regression (PLSR). According to the analysis results of the above methods, the importance orders of these factors are shown in table 2.

Table 2. Importance orders of influence factors.

Methods	1	2	3	4	5	6	7	8	9	10	11	12
CA	X_2	X_8	X_{10}	X_3	X_6	X_{11}	X_9	X_{12}	X_7	X_1	X_5	X_4
FIA	X_2	X_3	X_{11}	X_{12}	X_8	X_1	X_6	X_{10}	X_5	X_4	X_9	X_7
PCR	X_2	X_{10}	X_8	X_6	X_1	X_7	X_3	X_5	X_{12}	X_4	X_{11}	X_9
PLSR	X_2	X_8	X_{10}	X_3	X_6	X_{11}	X_9	X_{12}	X_7	X_1	X_5	X_4

Referring to table 2, the importance order of influence factors determined by every selection method is different. The analysis results are synthesized to objectively evaluate the importance of each influence factor. In this paper, the factors in the first five score 1, in the last five score -1, and in the middle score 0. Table 3 shows the score result.

Table 3. Score result of influence factors.

Influence factor	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}	X_{11}	X_{12}
Score	-1	4	3	-4	-4	3	-3	4	-2	2	0	-2

According to table 3, X_2, X_3, X_6, X_8 and X_{10} have more influence on acetone purity. Moreover, X_8 is highly correlative with X_{10} and X_6 (the result of correlation analysis shows that the correlation coefficient between X_8 and X_{10} is 0.7979, and between X_8 and X_6 is 0.9823). Therefore, the selected important influence factors are X_2, X_3 and X_8, which is coincident with the technological mechanism analysis. The three selected factors are treated as the input variables of CFNN.

4 Experiments and Results

Define a three-input-one-output CFNN without normalization with input data as reflux ratio (X_2), flux of circulating acetone (X_3) and temperature of top tower (X_8) and output data as acetone purity. Of the 400 pairs of input-output data, 2/3 are treated as training data and the rest 1/3 are used as testing data. After the network is trained, testing data are input to validate the prediction precision and generality of the prediction model, and the testing outputs are transformed to original data form. In addition, CFNN with normalization is applied to train and test the same data. The curve concerned with error between real data and prediction values of acetone purity is shown in figure 2. Table 4 gives the correct prediction rate of both CFNN-based models within certain error scope.

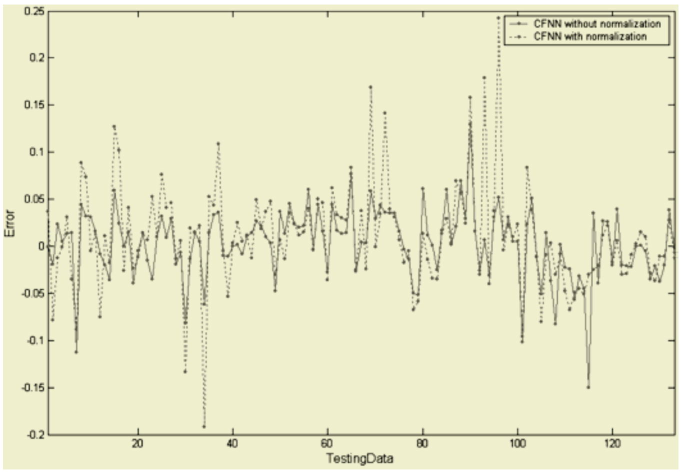


Fig. 2. Error Curve.

Table 4. Correct prediction rate.

Error scope	CFNN without normalization	CFNN with normalization
≤0.02	45.11%	36.84%
≤0.05	84.96%	77.44%

We also compare the results of CFNN without normalization and CFNN with normalization in terms of number of epochs and prediction error. The comparison results are shown in table 5 and 6.

Table 5. Comparison with respect to training epochs.

Algorithms	Initial Global Error	Number of Epochs
CFNN without normalization	1.9088	52
CFNN with normalization	1.7104	69

Table 6. Comparison with respect to prediction error.

Algorithms	Mean	Standard Deviation	Maximum Error	Relative Error
CFNN without normalization	0.0031	0.0377	0.1500	≤15.5%
CFNN with normalization	0.0096	0.0563	0.2422	≤25%

CFNN without normalization with BP learning algorithm is an efficient technique for the prediction of acetone product quality According to the comparison results in table 4, 5 and 6, CFNN without normalization is superior to CFNN with normalization in training epochs and prediction results.

5 Conclusions

Considering that the relation between technological influence factors and acetone product quality has the characteristics such as nonlinearity, uncertainty and fuzziness, CFNN without normalization is applied in the acetone refining process to predict product quality. Variables selection is an effective method for improving quality and accuracy of CFNN via selecting important technological factors. The experiment indicates that the method has theoretical meaning and application value. However, there still exist some questions unsolved in the application of CFNN without normalization to predict acetone product quality, such as increasing prediction precision and reducing computation time to make the model more practical; these issues are valuable for further exploration.

Acknowledgements

This research was supported by National High-tech. R&D Program of P. R. China (Grant No. 2002AA414610).

References

1. Zhang, Y.Q. and Kandel, A.: Compensatory Neurofuzzy Systems with Fast Learning Algorithms. IEEE Transactions on Neural Networks, **9** (1998) 83-105
2. Zeng, X.H., Geng, X.Y. and Huang, X.Y.: Predicting Reservoir Water Saturation Based on Fuzzy Neural Networks. Journal of System Simulation, **15** (2003) 735-736
3. Seker, H., Evans, D.H.: Compensatory Fuzzy Neural Networks-Based Intelligent Detection of Abnormal Neonatal Cerebral Doppler Ultrasound Waveforms. IEEE Transactions on Information Technology in Biomedicine, **5** (2001) 187~194
4. Seker, H., and Evans, D.H.: Non-normalised Compensatory Hybrid Fuzzy Neural Networks. In: Proc. Int. Neural Networks Joint Conf., **6** (1999) 4300-4303
5. Lin, Y. and Cunningham, G.A.: A New Approach to Fuzzy-Neural System Modeling. IEEE Transactions on Fuzzy Systems, **3** (1995) 190-197
6. Zhou, X.Z., Fan, Z.X., Zhan, J.J.: Application of Fuzzy Mathematics in Chemistry. National University of Defense Technology Press, Changsha (2002)
7. Chen, N.Y., Qin, P., Chen, R.L., Lu, W.C.: Application of Pattern Recognition Methods in Chemistry and Chemical Engineering. Science Press, Beijing (2000)

Combining Classifiers in Software Quality Prediction: A Neural Network Approach

Qi Wang[1], Jie Zhu[1], and Bo Yu[2]

[1] Electronic Engineering Department, Jiaotong University, Shanghai 200030 China
{wangqi_sjtu,jiezhu}@sjtu.edu.cn
[2] System Verification Departments, Lucent Technology Optical Network Ltd, Shanghai 200233 China
boyu@lucent.com

Abstract. Software quality prediction models seek to predict quality factors such as whether a component is fault prone or not. This can be treated as a kind of pattern recognition problem. In pattern recognition, there is a growing use of multiple classifier combinations with the goal to increase recognition performance. In this paper, we propose a neural network approach to combine multiple classifiers. The combination network consists of two neural networks: a Kohonen self-organization network and a multilayer perceptron network. The multilayer perceptron network is used as Dynamic Selection Network (DSN) and Kohonen self-organization network is served as the final combiner. A case study illustrates our approach and provides the evidence that the combination network with DSN performs better than some other popular combining schemes and the DSN can efficiently improve the performance of the combination network..

1 Introduction

Software quality models seek to predict quality factors of software components based on product and process attributes. A quality attribute, for example, "reliability", may be measured with various factors such as the time between failures or the number of faults. Quite often, predicting the number of faults is not necessary. Rather, identifying components that are probably fault-prone may be sufficient for practical purposes [1]. Many of software components' metrics can be collected earlier in the development life cycle, such as lines of code, total operators, number of control structures, etc. These software complexity metrics can be considered as predictors to predict the quality of software components [2]. Thus, the software engineers can predict whether a component is fault-prone early enough to apply reliability enhancing processes to those high-risk components.

There are many methods and techniques that can be used for software fault prediction including optimal set reduction [3], fuzzy classification [4], classification trees [5],[6], neural networks [7], etc. Since universal models do not exist, for a company, selecting an appropriate quality prediction model is a difficult decision. It is widely believed that combining several classifiers will improve the performance of individual classifier. Previous methods for classifier combination include intersection of decision regions [8], voting methods [9], prediction by top choice combinations [10], and use of Dempster-Shafer theory [11].

In this paper, a combining scheme based on neural network is proposed. The network is composed of two kinds of neural network: a Kohonen self-organization network and a multilayer perceptron network. The multilayer perceptron network is used as Dynamic Selection Network (DSN) and Kohonen self-organization network is served as the final combiner. The approach is demonstrated by using software data collected over a very large telecommunication system maintained by professional programmers in Lucent Technology Ltd. The experimental results show that this combination network performs better than some other combining schemes, and that the DSN can improve the performance of such combination network.

2 Combination Network

The overall system is composed of M individual classifiers, a DSN based on multilayer perceptron neural network and a combiner based on Kohonen self-organization neural network. The structure of this system is shown in Figure 1.

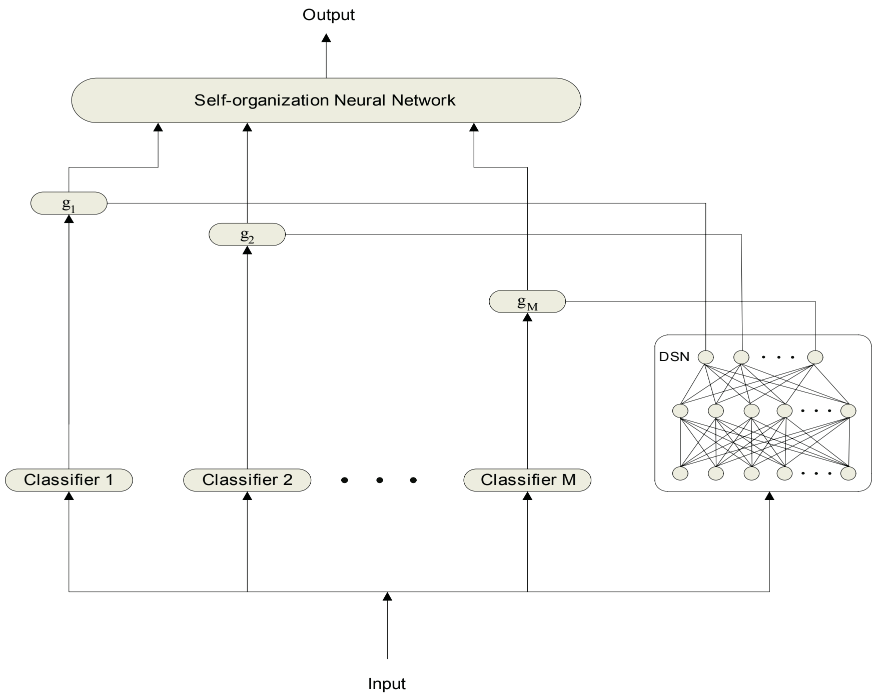


Fig. 1. The overall system: M individual classifiers, a DSN and a Self-organization Neural Network.

2.1 Bagging

To test the performance of the combination network, the technique of bagging is used. The M classifiers used in the bagging procedure are CARTs which are popularly employed in software quality prediction [5],[6]. Bagging is a simple method to increase the performance of a classification technique that depends on training the classifier. Bagging consists of the following steps:

1) New training sets are created by randomly sampling with replacement from the original training set. A number of training sets between 10 and 20 is sufficient. The number of samples in each training set is usually equal to the number of samples in the original training set. Note that the number of different samples is probably smaller, as doubles are possible (and even very likely).
2) For each new training set, train a classifier using a consistent technique. The bagged classifier is now complete.
3) For classification, each sample is classified by all classifiers.

2.2 Kohonen Self-organization Neural Network

As we know, information processing in the human brain is concerned with extracting significance from the external patterns in the world and then combining them. For example, recognizing an object often relies on the integration of vision, tactile sensation, sense of hearing and sense of taste. The Kohonen self-organization neural network is similar to the biochemistry mapping model of the brain. Evidence combination in the Kohonen self-organization neural net is in accordance with the human manner of recognition. Hence the Kohonen self-organization neural network is employed as the classifier combiner. The Kohonen self-organization neural network was proposed by Kohonen in 1982. The self-organization neural network realizes the self-organization feature map, and the neurons are placed at the nodes of a lattice that is usually one or two-dimensional. The neurons become selectively tuned to various input patterns or classes of input patterns. A self-organization feature map is characterized by the formation of a topographic map of the input patterns, in which the spatial locations (i.e. coordinates) of the neurons in the lattice correspond to intrinsic features of the input patterns. The basic principles of Kohonen's neural network model are described in [12],[13].

2.3 Dynamic Classifier Selection

The DSN is a three-layer perceptron network. It takes a sample as input, and outputs a real number between 0 and 1. For each classifier being combined it indicates how much confidence should be placed in the classifier's decision on every given sample. The way that DSN affects the combining network is very similar to the gain control network used in the multiple expert learning network architecture. Backpropagation is used to train DSN. The target for an output unit in DSN is 1 if the corresponding classifier has the correct answer on a given sample, and 0 if the classifier misclassifies. This target function should lead DSN to approximate P (classifier *i* is correct | given sample S).

3 Case Study

The training and testing data used in this paper are collected over a very large telecommunication system, written in high-level language (C and C++), and maintained by professional programmers in Lucent Technology Ltd. The entire system has more than 10^6 lines of code. We study two products, namely: A and B, and four consecutive releases of every product, labeled as RA1~RA4, RB1~RB4 are considered. The number of observations in these eight different products' releases is about 500 respectively. A component consists of a set of related source-code files. Fault data are collected at the component-level by the fault reporting system. A component is considered:

- Fault-prone (fp) if any faults were discovered by customers or system test engineers and resulted in changes to source code in the component,
- Not fault-prone (nfp), otherwise.

Some Software metrics are extracted from these two products (Table 1).

Table 1. Software metrics on routine level.

Symbol	Description
RtnArgXplSum	Sum of the explicit argument numbers
RtnCalXplNbr	Number of explicit function/method calls in the routine
RtnCastXplNbr	Number of explicit type casts in the routine
RtnComNbr	Number of comment sections in the routine scope
RtnComVol	Size in characters of all the comments in the routine
RtnCplCtlSum	Total complexity of the control predicates
RtnCplCycNbr	Cyclomatic number of the routine
RtnCplExeSum	Total complexity of the executable statements
RtnStmDecNbr	Number of declarative statements in the routine
RtnStmDecObjNbr	Number of variable/object declaration statements

4 Experimental Results

In the experiment, dataset is generated from the four consecutive releases of the software products. Every sample in the dataset is a feature-value pair. The feature includes the software metrics extracted from the corresponding software component, and the value indicates whether the component is fault-prone or not. The dataset is divided into training and test set. The training set is composed of the earlier two releases and the test set includes the later two releases. There are 1023 samples in training set, 987 samples in test set. The experiment procedure is as follows:

1) 19 CART classifiers are created by bagging. This number is well chosen over 10 to make certain that any late bagging effects are captured.
2) For each possible number of classifiers n ($1 \leq n \leq 19$), randomly draw a combination of classifiers of size n from the available 19 classifiers. Repeat this 100 times, so there will be 100 of such combinations for each size n.
3) For each combination of classifiers drawn, classify all test samples.
4) For all combination size n, record the mean classification accuracy over all combinations for that size.

To test the performance of our combination network, we implement several different combination schemes for comparison. Such schemes include majority vote rule, plurality vote rule and sum rule. To test the effect of the DSN, we compare the combination network with DSN and the one without DSN.

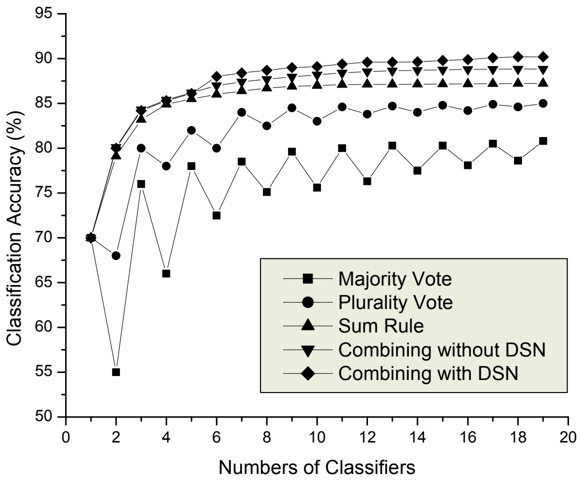


Fig. 2. Experiment results.

The results of the experiment are displayed in Figure 2. The performance is given in percentages of correctly classified samples. The x-axis denotes the number of classifiers in the combination. The results shown for each combination size are the average over all 100 random combinations of classifiers. The serrate edge of some performance curves (e.g. the majority vote and the plurality vote) is due to the difference in an odd or an even number of classifiers. An even number of classifiers is more likely to produce a tied result when two classes vie for the top spot.

The experiment results show that the combination network with DSN performs best in the five combining schemes. Note that when the number of classifiers is smaller than 10, the effect of DSN is very slight. But when the number of classifiers is between 10 and 20, the combination network with DSN performs better than the combination network without DSN. Therefore, we can conclude that the DSN can improve the performance of the combination network.

5 Conclusions

In this paper, we presented a combining scheme based on neural network. A Kohonen self-organization network and a multilayer perceptron network constitute the system. The multilayer perceptron network is used as DSN and Kohonen self-organization network is served as the final combiner. The approach is demonstrated by using software data collected over a very large telecommunication system maintained by professional programmers in Lucent Technology Ltd. The experimental results show that the combination network performs better than some other combining schemes and the DSN can efficiently improve the performance of the combination network.

References

1. Munson, J.C., Khoshgoftaar, T.M.: The Detection of Fault-prone Programs. IEEE Transactions on Software Engineering, **18** (1992) 423–433

2. J.C. Munson, T.M. Khoshgoftarr: Software Metrics for Reliability Assessment. In M.Lyu, editor. Handbook of Software Reliability Engineering, chapter 12, McGraw-Hill, New York (1996) 493–529
3. Briand, L.C., Basili, V.R., Thomas, W.M: A Pattern Recognition Approach for Software Engineering Data Analysis. IEEE Transaction on Software Engineering, **12** (1992) 931–942
4. C. Ebert: Classification Techniques for Metric-based Software Development. Software Quality Journal, Chapman & Hall, **18** (1996) 255–272
5. Khoshgoftaar, T.M., Allen, E.B., Jones, W.D., Hudepohl, J.I.: Classification Tree Models of Software Quality over Multiple Releases. In Proceedings of the 10th International Symposium on Software Reliability Engineering (1999) 116–125
6. Khoshgoftaar, T.M., Seliya, N.: Tree-based Software Quality Estimation Models for Fault Prediction. Proceedings of Eighth IEEE Symposium on Software Metrics, **4** (2002) 203–214
7. Khoshgoftaar, T.M., Allen, E.B., Hudepohl, J.P., Aud, S.J.: Application of Neural Networks to Software Quality Modeling of a Very Large Telecommunications System. IEEE Transactions on Neural Networks, **5** (1997) 902–909
8. Haralick, R.,M.: The Table Look-up Rule. Communication Statistics-Theory and Methods, A**5** (1976) 1163–1191
9. Mazurov, V.D., Krivonogov, A.I., Kazantev, V.L.: Solving of Optimization and Identification Problems by the Committee Methods. Pattern Recognition, **20** (1987) 371–378
10. Wernecke, K.D.: A Coupling Procedure for the Discrimination of Mixed Data. Biometrics, **48** (1992) 497–506
11. Mandler, E., Schuermann, J.: Combining the Classification Results of Independent Classifiers Based on the Dempster/Shafer Theory of Evidence. Pattern Recognition and Artificial Intelligence, North-Holland, Amsterdam, **5** (1988) 381–393
12. Kohonen, T.: An Introduction to Neural Computing. Neural Networks, **1** (1988) 3–16
13. Kohonen, T.: Self-organization and Associative Memory. Springer-Verlag, Berlin (1989)

Neural-Network-Driven Fuzzy Reasoning for Product Development Processes

Yingkui Gu[1], Hongzhong Huang[1,2], and Yonghua Li[1]

[1] Key Lab. for Precision and Non-traditional Machining Technology of Ministry of Education, Dalian University of Technology, Dalian, Liaoning 116023, China
{guyingkui,yonghuali}@163.com

[2] School of Mechatronics Engn, University of Electronic Science and Technology of China, Chengdu, Sichuan 610054, China
hzhhuang@dlut.edu.cn

Abstract. The quantitative and qualitative dependency measures of serial and parallel product development processes are analyzed. And the neural-network-driven fuzzy reasoning mechanism of dependency relationships is developed in the case that there is no sufficient quantitative information or the information is fuzzy and imprecise. In the reasoning mechanism, a three-layer feedforward neural network is used to replace fuzzy evaluation in the fuzzy system. A hybrid learning algorithm that combined unsupervised learning and supervised gradient-descent learning procedures is used to build the fuzzy rules and train membership functions.

1 Introduction

Product development process can be viewed as a set of sub-processes that their physical meanings are varied continuously along with time [1]. The dependency relationship is one of the most important relationships among product development processes. The purposes to analyze the dependency relationship are as follows:

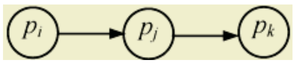
1. Offering a method for design candidate identification under the dynamic and uncertain development environment.
2. Reducing the cost and complexity of process improvement.
3. Increasing the amount of information provided to the designers for making decisions in the case that design information is incomplete and imprecise.

However, it is very difficult to analyze the quantitative and qualitative dependency relationships because of the complexity, fuzziness and dynamic uncertainty of product development process. The disadvantage will prolong the development time and increase the cost. Fuzzy logic and neural network provide stronger tools for process modeling under the fuzzy and uncertain development environment [2-4]. In this paper, we present an approach to analyze the dependency relationships among product development processes using fuzzy logic and neural network.

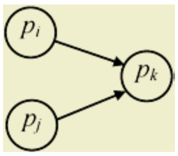
2 The Dependency Reasoning Mechanism

The serial processes and parallel processes are shown in Fig. 1.

As shown in Fig. 1(a), the qualitative dependency reasoning rules for serial processes are developed and listed as follows:



(a) Serial structure



(b) Parallel structure

Fig. 1. Structure of process.

(1) If ψ_{ij} ="+" and ψ_{jk} ="+", then $\psi_{ij\to k}$ ="+";

(2) If ψ_{ij} ="+" and ψ_{jk} ="–", then $\psi_{ij\to k}$ ="–";

(3) If ψ_{ij} ="–" and ψ_{jk} ="–", then $\psi_{ij\to k}$ ="+";

(4) If ψ_{ij} ="–" and ψ_{jk} ="+", then $\psi_{ij\to k}$ ="–";

(5) If ψ_{ij} ="+"("–") and ψ_{jk} ="0", then $\psi_{ij\to k}$ ="0";

(6) If ψ_{ij} ="0" and ψ_{jk} ="+"("–"), then $\psi_{ij\to k}$ ="+"("–").

As shown in Fig. 1(b), the qualitative dependency reasoning rules for parallel processes are developed and listed as follows:

(1) If ψ_{ik} ="+" and ψ_{jk} ="+", then $\psi_{ij\to k}$ ="+";

(2) If ψ_{ik} ="–" and ψ_{jk} ="–", ψ_{jk} ="–" then $\psi_{ij\to k}$ ="–";

(3) If $\psi_{ik}(\psi_{jk})$="0" and $\psi_{jk}(\psi_{ik})$="+"("–"), then $\psi_{ij\to k}$ ="+"("–");

(4) If $\psi_{ik}(\psi_{jk})$="+"("–") and $\psi_{jk}(\psi_{ik})$="–"("+"), then $\psi_{ij\to k}$ = ?.

Where "+" represents that the change of process p_i has good influence on the improvement of process p_j. "–" represents that the change of process p_i has bad influence on the improvement of process p_j. "0" represents that the change of process p_i has not influence on the improvement of process p_j. ? represents the other cases.

3 Neural-Network Driven Fuzzy Reasoning Mechanism

To parallel structure, if the two processes change along the same direction, the common influence on the third process is to make it be strengthened. However, if the directions of the change of the two processes differ, the common influence on the third process is sometimes vague. The neural-network-driven fuzzy reasoning approach provides a more flexible and natural way to represent the quantitative dependency among parallel processes.

Let V is the rate of change of process p_k that caused by the change of processes p_i and p_j. $T(V)$ is the set of names of linguistic terms of V.

$T(V)$ = {PL, PM, PS, NL, NM, NS}
= {Positive Large, Positive Same, Positive Small, Negative Large,
Negative Same, Negative Small}

Thirty-six fuzzy rules can be developed to represent the dependencies among process p_i, p_j and p_k, as shown in Table 1.

Table 1. Fuzzy rules.

Then $\psi_{ij\to k}$		IF ψ_{ik}					
		PS	PM	PL	NS	NM	NL
IF ψ_{jk}	PS	PS	PM	PL	PS or NS	NS	NL
	PM	PM	PL	PL	PM	NS or PS	NL or NM
	PL	PL	PL	PL	PL	PL	?
	NS	PS or NS	PM	PL	NS	NM	NL
	NM	NM	PS or NS	PL	NM	NL	NL
	NL	NL	NL or NM	?	NL	NL	very NL

In fuzzy reasoning, the minimum and maximum membership function values can be obtained using logical "and" and "or" operations. We can achieve the output value by calculating the center of gravity of the output membership function. For each rule, we can obtain the membeship function measures for the two input variables ψ_{i0} and ψ_{j0} firstly, and the smaller value can be selected as the measure to evaluate the matching of the rule. The result membership function of fuzzy reasoning considering only one rule is the minimum of the membership function at the THEN part of the rule and rule matching measure. The result membership function of fuzzy reasoning $\mu_{i,j\to k}(\psi)$ considering all the relevent rules can be achived by obtaining the maximum value of these result membership functions for these rules.

The output value of variable is the gravity center of the output membership function $\mu_{i,j\to k}(\psi)$, calculated by [4]

$$\psi_w = \frac{\int_{\psi_{\min}}^{\psi_{\max}} \mu_{i.j\to k}(\psi)\psi \mathrm{d}\psi}{\int_{\psi_{\min}}^{\psi_{\max}} \mu_{i,j\to k}(\psi)\mathrm{d}\psi} . \tag{1}$$

However, with the increase of number of fuzzy rules and complexity of membership functions, the calculation of output variable values becomes difficult. Because neural network have learning capacity and model-free characteristics, it can be used to replace the fuzzy reasoning process in fuzzy logic systems.

In our model to reason the dependency relationships among product development processes, a three-layer feedforward neural network with an input layer, a hidden layer and an output layer was used to replace the fuzzy reasoning process, as shown in Fig. 2. The input nodes are used to describe the fuzzy membership functions for different fuzzy sets. The input nodes can be grouped based on the number of processes. In each group, there exist six nodes used to represent the fuzzy set membership function measures. x_0 and y_0 are the initial values of process p_i and p_j. The output nodes are used to describe the output membership functions for output variables. There exist six output nodes used to represent membership functions of six output variables, where z_w is the gravity center of the output membership function. The hidden layer functions are operated as membership functions and fuzzy rules.

In the training process, a hybrid learning algorithm that combined unsupervised learning and supervised gradient-descent learning procedures is used to build the fuzzy rules and train membership functions. Through assigning values to the input

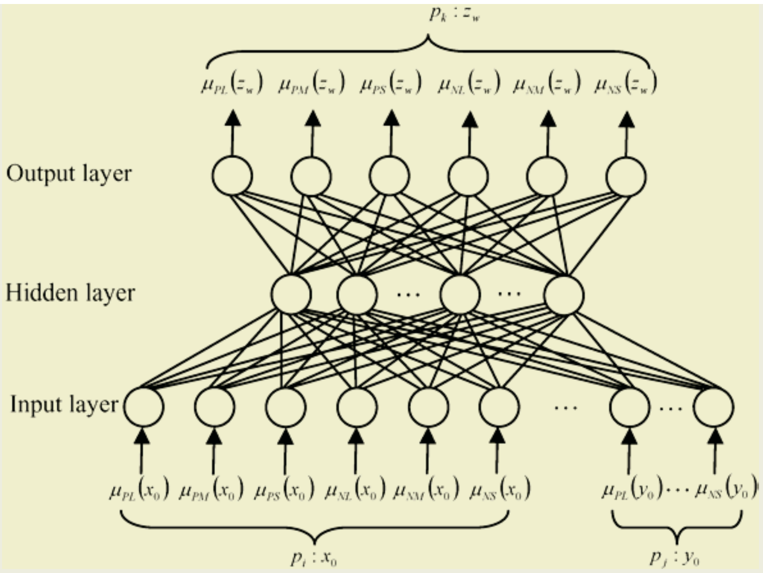

$p_k : z_w$
$\mu_{PL}(z_w)$ $\mu_{PM}(z_w)$ $\mu_{PS}(z_w)$ $\mu_{NL}(z_w)$ $\mu_{NM}(z_w)$ $\mu_{NS}(z_w)$
Output layer
Hidden layer
Input layer
$\mu_{PL}(x_0)$ $\mu_{PM}(x_0)$ $\mu_{PS}(x_0)$ $\mu_{NL}(x_0)$ $\mu_{NM}(x_0)$ $\mu_{NS}(x_0)$
$\mu_{PL}(y_0)\cdots\mu_{NS}(y_0)$
$p_i : x_0$
$p_j : y_0$

Fig. 2. A three-layer feedforward neural network for fuzzy reasoning.

variables, calculating the membership function measures of the input nodes, and achieving the output membership function using fuzzy reasoning, the correct data set can be achieved. All the fuzzy rules are involved and can be encoded in the neural network.

4 Case Study

Take the design process of worm drive as an example to analyze the dependency relationships using the proposed neural-network-driven fuzzy reasoning mechanism. Requirement analysis (p_1), cost analysis (p_2) and scenario design (p_3) are the three sub-processes of the whole development process. The process structure of the three processes is parallel. First, we pick up the key process variables of each process respectively, $v_1 = \text{transmission efficiency}$, $v_2 = \text{cost}$, $v_3 = \text{satisfaction degree of scenario}$. The change of key process variables is the dominant factors to result in the change of the process and its relative processes. The key of improving the process is to improve its key variables. Take the change rate of key process variables as fuzzy variable (i.e., ψ_1, ψ_2 and ψ_3). Thirty-six fuzzy rules can be developed as shown in Table 2.

Now, the process variable v_1 (transmission efficiency) will increase 20% because of the change of working conditions of worm, which will make v_2 (cost) increase 30% at the same time. The neural-network-driven fuzzy reasoning mechanism proposed in Section 3 can be used to analyze the change of v_3 (satisfaction degree of scenario) which results from the change of transmission efficiency and cost. A three-layer feedforward neural network is developed to reason the dependency relationships among the three processes, as shown in Fig. 3. The neural network includes twelve input nodes and six output nodes. The input nodes can be classed into two groups.

From the results we can see, the satisfaction degree of scenario decrease a little when the transmission efficiency increases 20%. That is to say, the process p_3 changes along the negative direction in some sort when p_1 and p_2 change according to the design requirements.

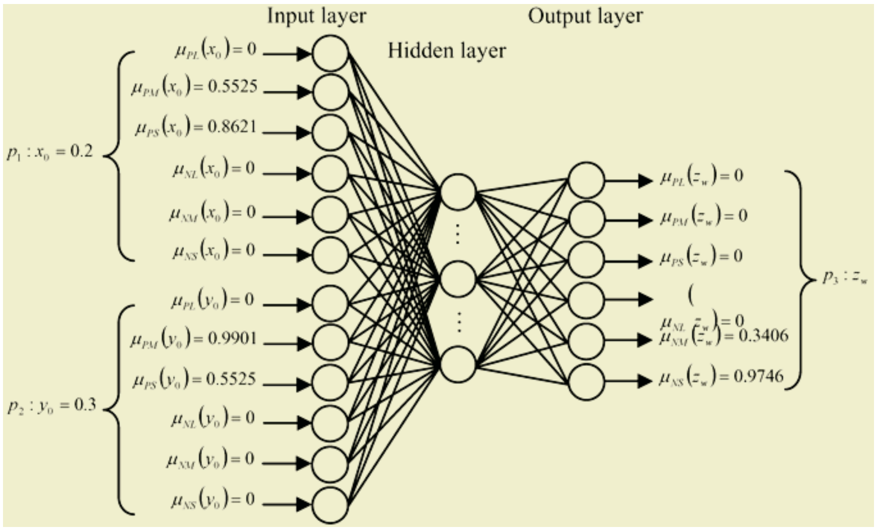


Fig. 3. The NN-driven fuzzy reasoning for dependency relationships among process p_1, p_2 and p_3.

Table 2. Fuzzy rules.

Then ψ_3		IF ψ_1					
		PS	PM	PL	NS	NM	NL
IF ψ_2	PS	PS	NS	NM	PS	PM	PL
	PM	PM	PS	NM	PM	PL	PL
	PL	PL	PS	NM	PL	PL	very PL
	NS	NS	NM	NL	PS	PM	PL
	NM	NM	NL	NL	NS	PS	PM
	NL	NL	NL	very NL	NL	NL	NM or NL

5 Conclusions

In the development process, the design information is fuzzy and incomplete sometimes. In order to offer a stronger decision-making tool for the designers, fuzzy logic and neural network are used to develop an intelligent reasoning mechanism. The neural-net-driven fuzzy reasoning mechanism can be used to analyze the dependency relations of development processes. The proposed method is effective in process programming and improving when the information is incomplete or the design knowledge is limited. It provides a method and theory foundation for realizing automatization of product development.

Acknowledgements

This research was partially supported by the National Excellent Doctoral Dissertation Special Foundation of China under the contract number 200232.

References

1. Gu, Y.K, Huang, H.Z., Wu, W.D.: Product Development Microcosmic Process Model Based on Constraint Nets. Journal of Tsinghua University, **43** (2003) 1448-1451
2. Kusiak, A.: Dependency Analysis in Constraint Negotiation. IEEE Transaction on System, Man, and Cybernetics, **25** (1995) 1301-1313
3. Du, T.C., Wolfe, P.M.: Implementation of Fuzzy Logic Systems and Neural Network in Industry. Computer in Industry, **32** (1997) 261-272
4. Sun, J., Kalenchuk, D.K., Xue, D., Gu, P.: Design Candidate Identification Using Neural Network-Based Fuzzy Reasoning. Robotics and Computer Integrated Manufacturing, **16** (2000) 383-396

The Integration of the Neural Network and Computational Fluid Dynamics for the Heatsink Design

Yeander Kuan and Hsinchung Lien

Department of Mechanical Engineering, Northern Taiwan Institute of Science and Technology, 2, XUE Yuan Rd., Peitou, Taipei 112, Taiwan, China
ydkuan@ntist.edu.tw1, hclien@ntist.edu.tw2

Abstract. The objective of the present paper is to develop an artificial intelligent system for heatsink parameter design and performance evaluation. The system is combined both the artificial neural network and the computational fluid dynamics (CFD) techniques. The algorithm of the artificial neural network used in this paper is the Back-propagation Neural Network (BNN), which makes the intelligent design and performance evaluation for the extruded heatsink. The related data of the thermal resistance, pressure drop, velocity drop, and average heatsink temperature rise on the different heatsink designs and operating conditions are obtained by the results of the CFD simulation and the data will be sent to the BNN. After well trained, the BNN model is able to accurate estimate the corresponding output values under different heatsink design and operating conditions.

1 Introduction

As continuous miniaturization and intensive application of the electronic devices results in rapid increase in the power density on the electronic dies, thermal management issue has been becoming one of the most important challenges in electronic packaging industry. The assemblage in small volumes yields a tremendous heat generation even if the heat production is low in the majority of electronic devices. The rate of heat removal for those components with high heat flux has become very important in order to keep the electronics devices at an acceptable operating temperature [1]. Kraus and Bar-Cohen [2] introduced the concept to apply the conventional heat transfer to the field of the cooling in the electronics cooling. In general, the thermal management in the electronics includes the levels of the system, heatsink, PCB board, and package. The detailed introduction and review of the basic principle, design, and analysis for the heatsinks are shown in the book of Kraus and Bar-Cohen [3]. The computational fluid dynamics (CFD) has been tremendously applied to the heatsink design and the heat transfer in the electronic systems [4],[5],[6].

Even the CFD simulation could reduce the cost and time of the design cycle, but the thermal designers still need to make several trial and errors to reach an acceptable result in each case. Therefore, a more efficient methodology is desired to conform the diversified requirements.

BNN (back-propagation neural network) has been successfully applied to many fields such as efficiently resolving problems with classification and prediction studies [7-11]. Up to now, very limit literature could be found applying neural network to the field electronics cooling; Kos [12] tried to use BNN to make the better placement of the cooling components of an electronic power system, none applied to the heatsink design in the literature. In this paper, a methodology of integrating BNN and CFD is introduced to make the prediction of the multi-inputs and multi-outputs on an extrude heatsink under some boundary conditions.

2 Problem Description

In this paper, the design of an extrude heatsink is presented using BNN and CFD. Figure 1 Illustrates the computational fluid dynamics (CFD) model construction, an extruded heatsink with heating element (a dummy CPU) put in a computational wind tunnel. Figure 2 shows the definition of heatsink parameters. The parameters include heatsink length, heatsink width, fin thickness, fin gap, fin height, and heatsink base height.

Assume the heatsink material is aluminum (thermal conductivity is 210W/m-K), the heating element (dummy die) material is copper (thermal conductivity is 35W/m-K), and the dimension of dummy die is 20mm x 20mm, the heatsink is square, i.e., Ls = Ws. In the numerical wind tunnel, the heasink is cooling under the different wind airflow velocity. Therefore, six inputs of the BNN are heatsink length or width (s_l or s_w), fin thickness (f_t), fin gap (f_g), fin height (f_h), heatsink base height (b_h), and inlet airflow velocity (V_{in}).

In general, the temperature on center point at the top surface of a chip (here is dummy die) T_c (°C) is observed to be a reference of evaluating the cooling performance. If the power input of the dummy die is P (W) and the ambient temperature is T_a (°C), then the thermal resistance from case to ambient can be defined as follows [4].

$$\theta_{ca}(^{o}C/W) = \frac{T_c - T_a}{P} \quad . \tag{1}$$

The other important properties are the pressure drop (ΔP, N/m^2), airflow velocity drop (ΔV, m/s), and average heatsink temperature increase (ΔT, °C). The pressure drop is defined the average pressure drop caused by the airflow passing through the fins of the heatsink; the airflow velocity drop is defined as the average airflow velocity drop caused by the airflow passing through the fins of the heatsink. Therefore there are four outputs through the BNN calculation: θ_{ca}, ΔP, ΔV, and ΔT.

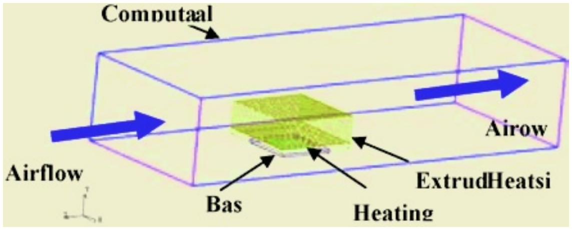


Fig. 1. The Illustration of the computational fluid dynamics (CFD) model construction, a extruded heatsink with heating element (a dummy CPU) put in a computational wind tunnel.

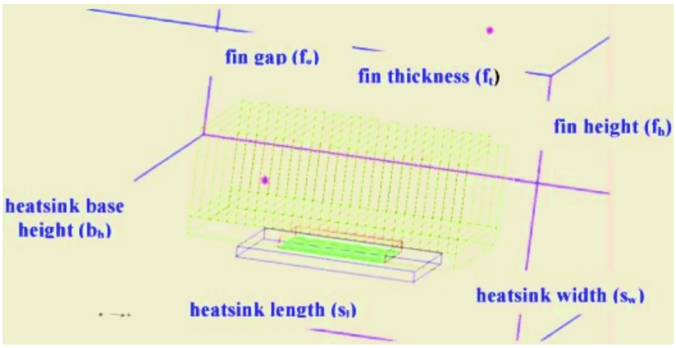


Fig. 2. The definition of heatsink parameters: including heatsink length (Ls), heatsink width (Ws), fin thickness (Tf), fin gap (Lg), fin thickness (Tf), and heatsink base height (Tb).

3 Basic Theories

3.1 Back-Propagation Neural Network (BNN)

The neural network has three layers, the input, the hidden, and the output layer. The BNN forward learning process optimizes the connection weighting u_{ij} from the input layer node x_i to the hidden layer nodes h_j and w_{jk} from the hidden layer node h_j to the output layer node y_k based on the input properties x_i. This is shown in equation 2 and 3:

$$h_j = f(h_{input(j)}) = \frac{1}{1+e^{-h_{input(j)}}} \quad . \tag{2}$$

$$y_k = f(y_{input(k)}) = \frac{1}{1+e^{-y_{input(k)}}} \quad . \tag{3}$$

where $h_{input(j)} = \sum_i (u_{ij}x_i - \theta_j)$ and $y_{input(k)} = \sum_j (w_{jk}h_j - \theta_k)$ represent all inputs to the hidden layer node j and all inputs to the output layer node k, respectively; and θ is the bias. The difference between the theoretical output y_k and the actual output t_k is the error of the output node k. That is, the neural network error function is $e_k=(t_k-y_k)$ and the cost function is $E = \frac{1}{2}\sum_k (t_k - y_k)^2$. The BNN backward learning process calculates backpropagation error functions, $\delta_k^w = (t_k - y_k)y_k(1-y_k)$ and $\delta_j^u = (\sum_{k=1}^{m} \delta_k^w w_{jk})h_j(1-h_j)$.

3.2 Computation Fluid Dynamics (CFD)

The CFD is to use numerical process and make the iterative calculation to solve the heat and fluid related governing equations. Through the numerical simulation, the fluid flow, heat transfer, mass transfer, chemical reactions, and related phenomena could be predicted. Fluid flow and heat transfer could be solved simultaneously in the CFD process. One of the CFD scheme, finite volume method (FVM) is widely used in the computational fluid dynamics field. In the FVM, the domain is discredited into a finite set of control volumes or cells. The general conservation (transport) equation for mass, momentum, energy, etc., are discredited into algebraic equations. The general conversation equation is shown in the Equation 4. The CFD simulation is done by Icepak, a finite-volume based CFD software [13].

$$\frac{\partial}{\partial t}\underbrace{\int_V \rho\phi dV}_{\text{Unsteady}} + \underbrace{\oint_A \rho\phi V\cdot dA}_{\text{Convection}} = \underbrace{\oint_A \Gamma\nabla\phi\cdot dA}_{\text{Diffusion}} + \underbrace{\int_V S_\phi dV}_{\text{Generation}} \qquad (4)$$

Eqn.	ϕ
Continuity	1
x-mom.	u
y-mom.	v
energy	h

3.3 The Integration of BNN and CFD

The present methodology is adopting CFD to run 96 trials under the limitation of parameter inputs are described as follows to fit the specific application: s_i 38-50 mm, f_t is 0.1-3 mm, the f_g is 0.1-3.5mm, f_h is 5-45 mm, b_h is 1-8mm, and V is 0-5 m/s. The input values are random generated. In the 96 sets of data, 72 are taken as the training data, and the rest 24 are taken as the validation data. The accuracy of the BNN model could be estimated after comparison with the CFD results.

4 Results and Discussion

Figure 3 is the convergent plot of the BNN training process. After 31846 epochs, the BNN model tends to convergent, and MAX error is about 16.43 %, and the RMS is about 7.63%. On the other hand, the BNN training module shows a pretty high estimated accuracy of 92.73%. Table 1 is the data comparison between the CFD modeling and BNN calculation. The results show that the BNN model is able to give a pretty good estimation compared with the CFD simulation. Figure 4 is the CFD model and the contour of temperature distribution of the best case for θ_{ca} under the 24 validation data. Each CFD simulation in this study takes about 20-30 minutes, but the BNN model could make thousands of predictions in minutes because of fixed node and weights after finishing training. Therefore, the BNN could make two major contributions, one is to predict the heatsink performance under some specific conditions in a short time, another is to make a better heatsink design based on the BNN model.

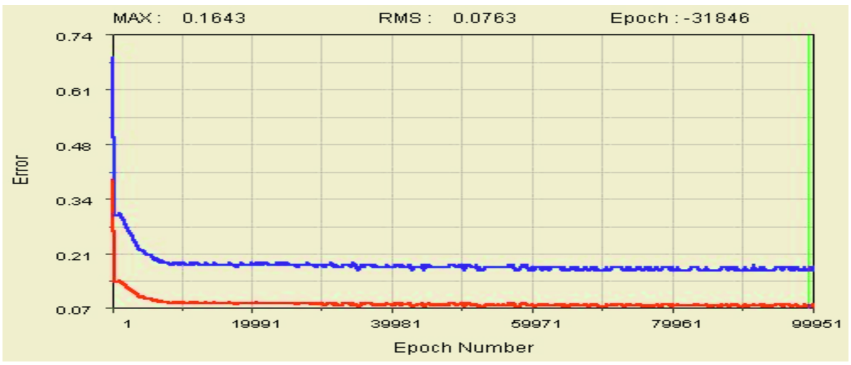
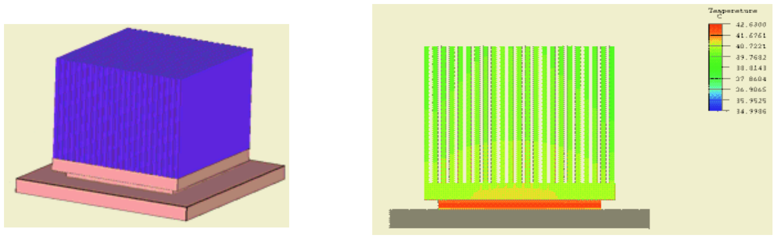


Fig. 3. The convergent plot of the model under BNN training based on 72 training data, the RMS falls into 7.63% and the maximum error is about 16%.



(a) CFD Model (b) Temperature Distribution

Fig. 4. The CFD model and the contour of temperature distribution at the best condition of θ_{ca} among the 24 validation data.

Table 1. The comparison between the CFD simulation and BNN results.

Inputs							Outputs											
							BNN	CFD	Err	BNN	CFD	Err	BNN	CFD	Err	BNN	CFD	Err
ID.	V_{in} (m/s)	s_l (mm)	f_t (mm)	f_g (mm)	f_h (mm)	b_h (mm)	ΔT	ΔT	ΔT	Θw	Θw	Θw	ΔP	ΔP	ΔP	ΔV	ΔV	ΔV
1	1.25	44	2	3	10	1.63	35.8	45.0	20.4%	3.3	4.2	20.0%	0.9	0.7	41.7%	0.3	0.3	17.3%
2	2.5	41	2	0.5	25	6	21.4	24.9	14.1%	2.1	2.4	12.9%	4.3	3.9	10.8%	0.7	0.7	3.1%
3	3.75	38	2	1.13	40	3.75	9.4	7.5	25.2%	0.9	0.8	16.6%	8.7	8.0	7.7%	1.1	1.1	0.7%
4	3.75	50	1.5	0.5	32.5	1.63	9.2	11.6	20.5%	1.1	1.2	8.9%	13.0	12.0	8.3%	0.9	0.9	2.6%
5	5	47	1.5	1.13	10	6	11.1	12.7	12.9%	1.1	1.2	14.0%	11.5	14.7	21.8%	1.1	1.2	1.1%
6	0.63	44	1.5	1.75	25	3.5	37.9	35.0	8.2%	3.4	3.3	3.1%	0.3	0.2	74.5%	0.2	0.2	3.6%
7	1.56	41.75	1.5	2.22	36.25	1.56	11.6	11.7	0.9%	1.2	1.3	2.4%	1.1	1.0	8.5%	0.4	0.4	3.3%
8	2.5	39.5	1.5	2.69	19.38	4.47	16.4	14.8	10.7%	1.6	1.5	8.5%	2.4	2.4	1.2%	0.6	0.6	0.5%
9	2.5	48.5	1.13	2.22	13.75	3.5	14.1	13.8	2.8%	1.4	1.4	1.8%	2.9	3.2	6.7%	0.5	0.6	9.7%
10	3.44	46.25	1.23	2.69	25	1.56	9.0	9.6	6.5%	0.9	1.0	5.5%	4.1	3.8	8.7%	0.6	0.6	2.9%
11	4.37	44	1.13	0.81	36.25	4.47	6.4	5.7	12.5%	0.7	0.6	7.3%	13.5	12.9	4.3%	1.2	1.3	9.4%
12	1.09	42.31	1.22	1.4	20.78	2.77	26.3	27.4	4.1%	2.5	2.6	5.0%	0.7	0.7	5.6%	0.3	0.3	10.0%
13	1.8	40.63	1.22	1.75	29.22	4.95	11.3	11.6	2.3%	1.2	1.2	0.7%	1.6	1.7	6.9%	0.5	0.5	1.4%
14	1.8	47.38	0.94	1.4	25	4.22	10.8	11.4	4.8%	1.1	1.2	0.6%	2.1	2.1	2.5%	0.5	0.5	0.5%
15	2.5	45.69	0.94	1.75	33.45	2.77	6.1	6.3	3.6%	0.7	0.7	8.8%	3.2	3.1	1.5%	0.7	0.7	0.0%
16	3.2	44	0.94	2.1	20.78	4.95	8.6	8.1	5.6%	0.9	0.9	2.9%	4.5	4.5	1.4%	0.8	0.7	2.7%
17	3.9	42.31	0.94	2.45	29.22	3.5	8.0	7.8	2.6%	0.8	0.9	1.3%	5.1	5.2	1.4%	0.7	0.7	0.4%
18	1.45	41.47	1.08	1.22	18.67	2.41	23.5	23.8	1.0%	2.2	2.3	3.7%	1.0	1.1	7.7%	0.4	0.4	3.0%
19	1.45	46.53	1.92	2.28	31.32	4.59	13.0	13.5	4.4%	1.3	1.4	2.3%	1.3	1.2	4.1%	0.4	0.4	4.0%
20	1.97	45.26	1.92	1.22	21.83	3.5	17.9	18.1	1.1%	1.8	1.7	3.5%	2.7	2.5	8.7%	0.5	0.5	3.3%
21	2.5	44	1.92	1.49	28.16	2.41	10.8	10.9	0.7%	1.1	1.1	0.3%	3.6	3.8	3.8%	0.7	0.7	0.9%
22	3.03	42.73	1.92	1.75	18.67	4.04	13.8	12.9	7.1%	1.4	1.3	6.3%	4.6	4.7	2.5%	0.8	0.8	0.1%
23	3.55	41.47	1.92	2.01	25	2.95	10.1	8.8	14.6%	1.0	0.9	12.2%	5.5	6.0	8.4%	0.9	0.9	4.9%
24	3.55	46.53	1.71	1.75	21.83	2.41	8.2	9.0	9.8%	0.8	1.0	14.4%	6.6	6.6	0.6%	0.9	0.9	4.9%

5 Conclusions

In this paper, the BNN has been successfully applied to the extrude heatsink design. The BNN is integrated with the computational fluid dynamics (CFD). The CFD simulations makes 96 sets of data, 72 of them are taken as the training samples, and other are taken to be the validation ones. According to the comparison between the BNN and CFD results, the maximum error is about 16.43% and the RMS is about 7.63%, and the BNN model could make a very fast estimation under acceptable accuracy. So after well trained under the training and testing data taken from the CFD, the BNN model could give quick heatsink performance estimation under some specific conditions; moreover, the BNN model could even help to make a better design of the heatsink. Following the BNN designs, the CFD could help to make the final adjustments and this will save a lot of design cycle and cost.

Acknowledgements

The authors would like to appreciate the financial support by the Taiwan Semiconductor Manufacturing Company, Ltd., and the acknowledgement would be extended to the full support throughout the research from Northern Taiwan Institute of Science and Technology.

References

1. Soule, C.A.: Future Trends in Heat Sink Design, Electronics Cooling, **7** (2001) 18-27
2. Kraus, A., Bar-Cohen, A.: Thermal Analysis and Control of Electronic Equipment. Hemisphere Publishing Corp (1983)
3. Kraus, A.D., Bar-Cohen, A.: Design and Analysis of Heat Sinks. John Wiley (1995)
4. Wong, H. Lee, T.Y.: Thermal Evaluation of a PowerPC 620 Multi-Processor in a Multi-Processor Computer. IEEE Transaction on Components, Packaging, and Manufacturing Technology – Part A, **19** (Dec. 1996) 469-477
5. Chang. J.Y., Yu, C.W., Webb, R.L.: Identification of Minimum Air Flow Design for a Desktop Computer Using CFD Modeling. Journal of Electronic Packaging. Transactions of the ASME, **123** (2001) 225-231
6. Yu, C.W., Webb, R. L.: Thermal Design of a Desktop System Using CFD Analysis. Seventeenth IEEE SEMI-THERM Symposium (2001) 18-26
7. Lien, H.C., Lee, S.: A Method of Feature Selection for Textile Yarn Grading Using the Effective Distance Between Clusters. Textile Res. J., **72** (2002) 870-878
8. Lien, H.C., Lee, S.: Applying Pattern Recognition Principles to Grading Textile Yarns, Textile Res. J., **72** (2002) 320-326
9. Lien, H.C., Lee, S.: Applications of Neural Networks for Grading Textile Yarns. Neural Computing and Applications, **13** (2004) 185-193. Lien, H.C., and Lee, S., Applications of Neural Networks for Grading Textile Yarns, Neural Computing and Applications, **13** (2004) 185-193
10. Ludwig, L., Sapozhnikova, E., Lunin, V., Rosenstiel, W.: Error Classification and Yield Prediction of Chips in Semiconductor Industry Applications. Neural Computing & Applications, **9** (2000) 202-210
11. Verikas, A., Malmqvist K, Bergman L, Signahl M.: Color Classification by Neural Networks in Graphic Arts. Neural Computing & Applications, **7** (1998) 52-64
12. Kos, A.: Approach to Thermal Placement in Power Electronics using Neural Networks. Proceedings - IEEE International Symposium on Circuits and Systems, **4** (1993) 2427-2430
13. Icepak 4.1 User's Guide, Fluent Inc. (2003)

The Modeling and Application of Cost Predication Based on Neural Network

Xiaoling Huang[1], Jiansheng Xue[1], and Liju Dong[2]

[1] School of Information Sci & Tech. , Liaoning Univ., Shenyang, Liaoning 110036, China
hxiao0@yahoo.com.cn
[2] School of Information Engineering, Shenyang Univ., Shenyang, Liaoning 110044, China

Abstract. Cost prediction is very important for cost control, but the factors of influencing cost are many and complex. The factors affect each other, and the coupling phenomenon exists, so enterprise cost is difficult to be predicted correctly. On the basis of production cost composition model, the product cost prediction model based on neural network is established. A hybrid algorithm that trains neural network weight by real-coded adaptive mutation genetic algorithm is designed, and it overcomes the disadvantage that traditional neural network is easy to fall into local minima. Furthermore, the model is successfully applied to cost prediction in an ore dressing plant, and it improves the prediction accuracy.

1 Introduction

Production cost control is very important to enterprise management. Production cost is a comprehensive economic and technical index, which reflects the difference of management level and production technology. With the development of information integration technology, the method of cost control is studied and cost control software is developed in many countries [1],[2],[3]. As the important part of cost control, cost prediction provides the basis for cost planning and information for dynamic cost control. Accurately predicting cost has become the key link of cost control. Now there are many methods to predict cost, such as, linear regression, non-linear regression and tendency prediction and so on [4]. Among these methods there is a common shortcoming that they must give the mathematics models of cost prediction or artificially confirm some parameters. Meanwhile, traditional mathematic models omit the couplings among influence factors of production cost. Therefore, their precisions are not high.

Because BP neural network has good nonlinear approach ability and higher prediction accuracy, it has been used for cost prediction [5]. Combined genetic algorithm with neural network, this paper presents hybrid training algorithm and establishes cost prediction model which the interactions among cost factors are taken into account. The method has been applied to an ore dressing plant to predict product cost and improved accuracy of prediction.

2 The Neural Network Weight Study Based on Real-Coded Genetic Algorithm

2.1 Coding Method

In order to guarantee network learning precision and avoid that weight step-change, the real coding is adopted. The structure of feed-forward neural network is shown in Fig.1. w_{ij} is the connection weight from input layer to hidden layer; w_{jh} is the connection weight from hidden layer to output layer; θ_j is the threshold of hidden layer node; θ_h is the threshold of output layer node; the transform function of hidden layer node is Sigmoid function.

During process of coding, all weights and thresholds of neural network are regarded as the genes of chromosome. All genes form chromosome vectors $V = [v_1, \cdots, v_k, \cdots, v_L]$, v_k is the k^{th} gene of a chromosome.

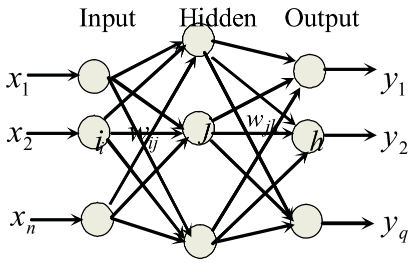


Fig. 1. A three-layer BP network.

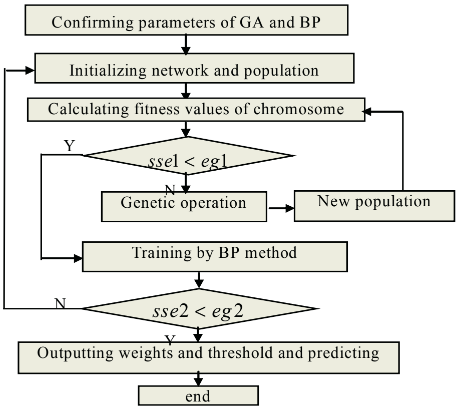

Confirming parameters of GA and BP
Initializing network and population
Calculating fitness values of chromosome
Y
sse1 < eg1
N
Genetic operation
New population
Training by BP method
N
sse2 < eg2
Y
Outputting weights and threshold and predicting
end

Fig. 2. The flow chart combined BP with GA.

2.2 Fitness Function

Assuming the population size is μ, that is, the number of chromosome is μ, compute the actual output of every chromosome y_k, where k=1,2,⋯,p, p is the number of input samples, then the sum of square error for the ith chromosome is

$$E_i = \sum_{k=1}^{p} (t_k - y_k)^2 \ . \tag{1}$$

Where i=1,2,⋯,μ, t_k is the expected output, then individual fitness of chromosome can be defined as:

$$f_i = 1/\exp(E_i - \min_i E_i) \ . \tag{2}$$

In the formula, exponential fitness is adopted to favor the individual fitness with smaller the sum of square error.

2.3 Crossover Operation

Crossover operation is a mechanism for combining parent chromosomes to produce offspring. In the paper, the arithmetic crossover and the directional crossover are selected randomly. Because arithmetic crossover can make sure that the next generation lie between two chromosomes of parents and the directional crossover can extend searching space effectively

a) Arithmetic crossover:

$$\begin{aligned} V_1' &= \alpha V_1 + (1-\alpha)V_2 \\ V_2' &= \alpha V_2 + (1-\alpha)V_1 \end{aligned} . \tag{3}$$

b) Directional crossover:

$$\begin{aligned} V_1' &= \alpha(V_1 - V_2) + V_1 \\ V_2' &= \alpha(V_2 - V_1) + V_2 \end{aligned} . \tag{4}$$

Where V_1,V_2 are vectors of chromosome, α is a random number that ranges from 0 to 1.

2.4 Mutation Operation

In the paper, adaptive mutation probability is adopted; searching ability and convergence speed of genetic algorithm is improved. The adaptive mutation operator as follows [6]:

$$v_k' = v_k + \Delta(t, b_k - v_k) \qquad or \;\; v_k' = v_k - \Delta(t, v_k - a_k) \tag{5}$$

The value of mutation site v_k ranges from a_k to b_k.

$$\Delta(t, y) = y * (1 - r^{T^\lambda}) . \tag{6}$$

Where $T = 1 - \frac{f(x)}{f_{\max}}$. $f(x)$ is the fitness value of the current individual, $f_{\max}$ is the maximum fitness value. $f_{\max}$ is hardly to be determined and the maximum fitness value of current generation is used instead. γ is the random number between 0 and 1, $\lambda \in [2,5]$, in this paper, λ =2.

2.5 Neural Network Algorithm Based on the Genetic Algorithm

The flow chart of hybrid algorithm is shown in Fig.2, eg1 and eg2 are the target error which are set, and $sse1$ is the sum of square error of GA, $sse2$ is the sum of square error of neural network.

3 Neural Network Cost Prediction Model

3.1 Product Cost Composition Model

The production cost of each product comprises many cost indexes, and each index comprises many cost factors in ore dressing enterprise. So the composition of production cost is shown as matrix:

$$Y_{m\times l} = \begin{bmatrix} \overline{y_1} \\ \overline{y_2} \\ \vdots \\ \overline{y_m} \end{bmatrix} = \begin{bmatrix} y_{11} & y_{12} & \cdots & y_{1l} \\ y_{21} & y_{22} & \cdots & y_{2l} \\ \vdots & & & \\ y_{m1} & y_{m2} & \cdots & y_{ml} \end{bmatrix}.$$

Where $l = \max\{\dim(\overline{y_1}), \dim(\overline{y_2}), \ldots, \dim(\overline{y_m})\}$; $\overline{y_i}$ is the vector of the ith cost index that forms production cost; y_i is the i th cost index that forms production cost; y_{ij} is the jth cost factor of ith cost index vector.

If $\dim(\overline{y_i}) = q < l$, then $y_{i_{q+1}}, y_{i_{q+2}}, \cdots, y_{il}$ equals 0, it means that the i th cost index is composed of q cost factors Production cost can be expressed by the equation:

$$y = \sum_{i=1}^{m} y_i = \sum_{i=1}^{m}\sum_{j=1}^{l} y_{ij}\,. \tag{7}$$

Every index cost can be expressed by the equation:

$$y_i = \sum_{j=1}^{l} y_{ij}\,. \tag{8}$$

3.2 Product Cost Prediction Model

Model 1: historical value prediction mode is defined that product cost is predicted according to history values. The model as follows:

$$y'(t) = f(y(t-n), y(t-n+1), \cdots, y(t-1))\,.$$

Where $y'(t)$ is production cost prediction value, the output of neural network; $y(t-k)$, $(k=1,2,\cdots n)$ is y the historical value of product cost, the input of neural network. The model has many inputs and one output.

Model 2: influencing factors prediction model is defined that production cost is predicted by using its influence factors. The factors are selected according to its dynamic characteristic and generality in order to reflect objectively the major factors. Influencing factors include quantitative factors and qualitative factors (such as, market situation, equipment operation condition, etc.). While using neural network to predict cost, the qualitative factors should be quantified fuzzily at first. Assuming the m factors affect production cost, so the prediction model as follows:

$$y'(t) = \sum_{i=1}^{m} y_i'(t)\,.$$

$$y_i'(t) = f_i(y_1(t-1), y_1(t-2), \cdots, y_1(t-k), \cdots, y_m(t-1), y_m(t-2), \cdots, y_m(t-k))\,.$$

Where $y'(t)$ is prediction value of product cost; $y_i'(t)$ is prediction value of index cost, the output of neural network; $y_i(t-k)$, $(i=1,2,\cdots,m)$ is historical cost of m indexes, the inputs of neural network. The model has many inputs and many outputs

Please always cancel any superfluous definitions that are not actually used in your text. If you do not, these may conflict with the definitions of the macro package, causing changes in the structure of the text and leading to numerous mistakes in the proofs.

4 Application Example

Because proportion of material consumption and energy consumption is about 80% in the product cost of ore-dressing, the model built in this paper is mainly to predict the cost of material consumption and energy consumption considering the key of controlling cost. To avoid that the prediction results of cost is influenced by prices of material and energy, the model in this paper is to predict the unit consumption of material and energy. The prediction unit consumption of material and energy multiplies price adding other expenses, then the prediction results of product cost can be obtained by the product cost composition model. The prediction of other expenses does not discuss in this paper.

Giving an example, in ore dressing industry, the neural network model is established and applied to dynamic cost control software in an ore dressing plant. According to user's need, the system can train samples, predict cost and confirm network structure automatically. In this paper, taking the prediction of product cost of material & energy in ore dressing plant as an example, we have predicted the three costs indexes of the Jingtie ore unit consumption, electric unit consumption and blast furnace gas unit consumption that affect each other, and the results showed that the model is better than the above proposed models. Integrated history value prediction model with influencing factors prediction model, MIMO prediction model with multi-indexes is built, the model as follows:

$$\begin{bmatrix} \hat{x}_1(t) \\ \hat{x}_2(t) \\ \vdots \\ \hat{x}_m(t) \end{bmatrix} = \begin{bmatrix} f_1(x_1(t-k), x_1(t-k+1), \cdots, x_1(t-1), \cdots, x_m(t-k), x_m(t-k+1), \cdots, x_m(t-1)) \\ f_2(x_1(t-k), x_1(t-k+1), \cdots, x_1(t-1), \cdots, x_m(t-k), x_m(t-k+1), \cdots, x_m(t-1)) \\ \vdots \\ f_m(x_1(t-k), x_1(t-k+1), \cdots, x_1(t-1), \cdots, x_m(t-k), x_m(t-k+1), \cdots, x_m(t-1)) \end{bmatrix}$$

According to production cost composition model, $y'(t) = \sum_{i=1}^{3} y_i'(t)$; Where x_i is the input of neural network, $\hat{x}_i$ is the output, the specific meaning as follows:

x_1 :historical cost of unit consumption of Jingtie ore	$\hat{x}_1$:prediction cost of unit consumption of Jingtie ore
x_2 : historical cost of unit consumption of electricity	$\hat{x}_2$:prediction cost of unit consumption of electricity
x_3 : historical cost of unit consumption of blast furnace gas	$\hat{x}_3$:prediction cost of unit consumption of blast furnace gas

Set k=2, that means using last two ten-days' historical data to predict the cost of the nest ten-days' data of three indexes.

4.1 Pretreatment of Input Value and Target Value

It takes 72 groups data of two years as sample in this paper, among which the first 60 groups data (Table1) is to build up prediction model as learning sample and the last 12groups data (Table2) is to test as test sample. (Only part datum are listed in Table1).

According to the scope of output value of sigmoid function, it is necessary to transform input value and target value as follows:

$$x_i' = [(x_i + x_{i\max} - 2x_{i\min})/(x_{i\max} - x_{i\min}) - 1] \times d_1 + d_2$$

Where parameters d_1 is set 0.9, parameters $d_2 = (1 - d_1)/2$, the input value ranging from 0.05 to 0.95 after normalized. As a result, the performance of network becomes better. $x_{i\max}$ and $x_{i\min}$ are the i^{th} index's maximum and minimum.

Table 1. Training Sample Data.

Serial number	Jingtie ore t/t	Electricity kwh/t	furnace gas m3/t	Serial number	Jingtie ore t/t	Electricity kwh/t	furnace gas m3/t
1	2.268	54.340	0.551	5	2.205	57.990	0.535
2	2.060	59.430	0.594	6	2.252	57.340	0.511
3	1.988	51.000	0.405	7	2.292	57.610	0.540
4	2.073	53.200	0.463	8	2.052	54.810	0.527

4.2 Pretreatment of Learning Sample

Learning samples are gotten from historical cost data. While using historical cost data to predict cost, we should classify the cost data at first. For instance, current cost is predicted according to history records of cost data in heavy repairing month when heavy repairing takes place in a period of ten days.

The historical data which are used to predict cost should be selected when enterprise operates steadily. There are many prediction forms that mean we can not only predict the total cost but also the cost of a certain index. Considering the coupling among cost factors, in this paper, taking two ten-days' cost data $x_1(t-2), x_1(t-1), x_2(t-2), x_2(t-1), x_3(t-2), x_3(t-1)$, as input, next ten-day's cost index data $\hat{x}_1(t), \hat{x}_2(t), \hat{x}_3(t)$, as output, then a learning sample is set up. Analogically, a group of learning samples are set up. Its advantage is that the couplings among cost indexes can be solved by model itself so that the prediction error for couplings is avoided, such as, predicting raw material cost just according to the historical cost of raw material.

4.3 Prediction Result

The parameters in this paper are: the number of Input layer neuron is 6; the number of hidden layer Neuron is 9; the number of output layer Neuron is 3; population size

N=80; the number of generation G=200; crossover probability Pc=0.6; the error of genetic algorithm eg1=0.01; the error of BP network eg2=0.0001. The system uses the cost data of last two ten-days to predict the cost of next ten-day in the plant. The prediction results are presented in Table 2 and Fig 3. The prediction results which are gotten by the methods of linear regression model and time series model are also presented. Regression analysis model adopts single regression to predict cost data, the formula is $y(t) = a + bx(t)$ (y is the prediction unit consumption of a index of a ten-day, x is the planning output of a ten-day); time series model adopts a weighted moving average to predict cost data, the formula is $y(t) = \sum_{i=1}^{4} a_i x(t-i)$ (y is the prediction unit consumption of a index of a ten-day, $x(t-i)$ is the historical unit consumption of a index, a_i (i=1,2,3,4) is the weighted coefficient, got 0.4,0.3,0.2,0.1 respectively, the weight becomes larger when close to prediction ten-day). According to Table 2(Only part (a) datum are listed in Table2), we know the method which is presented in this paper is more accurate than linear regression and time series. The greatest error does not exceed 6% in three kinds of unit consumption prediction. Especially linear regression method, when the liner relations between variable x and y are not very obvious, the error is relatively great, such as the blast furnace gas predicts that the error is close to 50%, predict that the result has been already insincere.

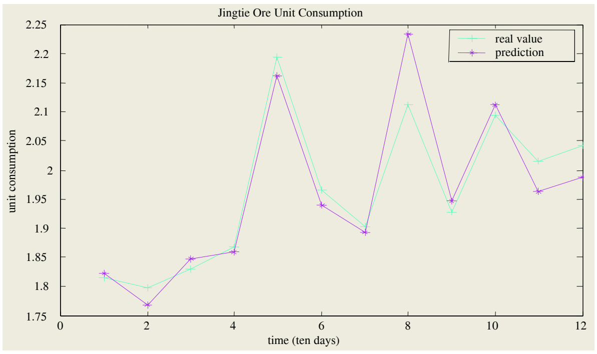


Fig. 3. (a) Predicting curve of Jingtie Ore unit consumption.

Table 2. (a) Test data of Jingtie Ore unit consumption.

Serial number	Actual value	Neural network		Linear Regression		Time sequence	
		Predicting value	Error %	Predicting value	Error %	Predicting value	Error %
1	1.815	1.823	-0.44	1.945	-7.16	2.083	-14.74
2	1.797	1.768	1.61	1.930	-7.40	2.063	-14.82
3	1.830	1.847	-0.93	1.932	-5.57	2.085	-13.92
4	1.867	1.859	0.43	1.988	-6.48	1.948	-4.32
5	2.193	2.161	1.46	2.094	4.51	1.818	17.11
6	1.965	1.940	1.27	2.099	-6.82	1.861	5.32
7	1.903	1.893	0.53	2.132	-12.03	1.927	-1.27
8	2.112	2.233	-5.73	2.150	-1.80	1.988	5.87
9	1.927	1.947	-1.04	2.102	-9.08	2.059	-6.82
10	2.094	2.112	-0.86	2.109	-0.72	1.972	5.83
11	2.016	1.962	2.68	2.104	-4.37	1.990	1.31
12	2.041	1.988	2.60	2.127	-4.21	2.043	-0.11

5 Conclusions

In the paper, the prediction model by using neural network of production cost is established; the model pays attention to the interactions among cost factors. The model adopts hybrid algorithm which is conjoined by real coded genetic algorithm and neural network to optimize neural network.

It overcomes the disadvantage that traditional neural network is easy to fall into local minima, and improves prediction accuracy. The model provides a method to predict accurately production cost.

References

1. Abudayyeh, O., Temel, B.: An Intranet-based Cost Control System, Advances in Engineering Software, **32** (2001) 87-94
2. Banin, A.A., Letavin, M.I.: Evaluation of Cost of Products of Integrated Iron and Steel Works by Mathematical Modeling Methods. Metallurgy, **66** (1999) 47-48
3. Li, X.N., Chai, T.Y: Dynamic Cost Control Method in Production Process and Its Application. 15th IFAC World Congress, Barcelona, Spain (2002) 238-243
4. Wang, Y.Z.: Industrial Enterprise Cost Prediction. Stand Letter Accounting Press, Shanghai (1995)
5. Wang, Q, Stockton, D J, Baguley, P: Process Cost Modelling Using Neural Networks. International Journal of Production Research, **38** (2000) 3811-3821
6. Zhou, M, Sun, S D: Genetic Algorithms: Theory and Applications. National Defense Industrial Press , Beijing, China (2000)

Combining SOM and Fuzzy Rule Base for Sale Forecasting in Printed Circuit Board Industry

Pei-Chann Chang[1] and K. Robert Lai[2]

[1] Department of Industrial Engineering and Management,
Yuan-Ze University, Nei-Li, Tao Yuan, Taiwan 32026, China
iepchang@saturn.yzu.edu.tw
[2] Department of Computer Science & Engineering,
Yuan-Ze University, Nei-Li, Tao Yuan, Taiwan 32026, China

Abstract. A key to success for manufacturing company in the worldwide competition is to build a reliable and accurate forecasting model that can predict in time suitable items at sufficient quantity and adapt to an uncertain environment. This paper presents a novel approach by combining SOM and fuzzy rule base for sales forecasting. Independent variables related to sales' variation are collected and fed into the SOM for classification. Then, corresponding fuzzy rule base is selected and applied for sales forecasting. Genetic process is further applied to fine-tune the composition of the rule base. Finally, using the simulated data, the effectiveness of the proposed method is shown by comparing with other approaches.

1 Introduction

Printed Circuit Board (PCB) industry occupies a significant portion of Taiwan's manufacturing, which produces about 15% of the global production value. With a ranking of top 3 in the world, its production value is only outperformed by the IC industry in Taiwan. PCB plays an extremely important position in our nation's economy, and has significant influences. Under the reality situation of the short lifespan and high circulating rate of related electronic products of PCB, general production models cannot fulfill customers' demands effectively. Thus, how to predict the order demand quantity and prepare the material flows in advance to reduce the cycle time has become an emergent issue to be dealt with. Furthermore, an efficient sales forecasting tool will be the key to strengthen the company's survival ability in the competitive environment. Therefore, it becomes indispensable to build a forecasting model to predict the production demands for next month or next year through an efficient and effective manner.

As for the development of forecasting approaches, the most commonly used in early times are mainly statistic methods such as Trend Analysis and Extrapolation. However, except for the consideration of time, this method cannot offer effective explanations for factors such as the nature of tendency, the seasonal factor, and the changes of industrial and social structures, so it's eventually weeded out. After that the alternative approaches include the so-called "Econometric Model" and "Time Series Analysis". The main purpose of Econometric Model is to investigate the relationship between the external economic variables, but research only considering Econometric Model as the analysis or forecasting tool has been rarely done so far.

Srinivasan [10] proposed a forecasting model based on Econometric Model integrating with Neural Network and proved that it could obtain smaller prediction errors compared to merely using Time Series Analysis or Regression Model. The advantage of this method is to avoid spending too much time and cost in collecting the data. Generally speaking, Time Series Analysis performs better than other models as for the forecasting models with trendy and seasonal features. Abdel-Aal [1] and Hsu et al. [5] adopted some Time Series Models as forecasting models and found positive results.

Recently with the development of the Artificial Intelligence Models, several methods are found to have better effectiveness than traditional models when being applied to forecasting models, and Artificial Neural Network (ANN) is the most commonly used tool applied in forecasting. After being trained by historical data, ANN can be used to predict the possible results produced in the future. Because it's fast and accurate, many researchers use ANN to solve sales forecasting related problems as in Chang and Hsieh [2], Chow and Leung [3], Kimoto and Asakawa [6], Kuo [7], Law and Au [8], Luxh et al. [9], Tawfiq and Ibrahim [12] and Tawfiq [11].

This paper presents a novel approach by combining SOM and fuzzy rule base for sales forecasting. Independent variables related to sales' variation are collected and fed into the SOM for classification. Then, corresponding fuzzy rule base is selected and applied for sales forecasting. Genetic process is further applied to fine-tune the composition of the rule base.

2 Features of the Prediction Model

Forecasting plays an important role in today's business planning. The fundamental basis of the master production schedule is according to the sales forecasted and the orders received from the sales department to explode the future production quantity of the next three months for production planning department to follow. However, traditional forecasting methods suffer from several deficiencies and limitations which make them severely inadequate for strategic business planning in today's business environment. First, the relationship between the forecasted output and the decision factors that influence it cannot always be expressed by a mathematical model. This limits the usefulness of such conventional forecasting methods like econometric forecasting. Second, today's business environment is constantly changing, thus causing the decision boundaries to shift. Conventional forecasting methods lack changes. Third, conventional forecasting methods rely heavily on large historical databases which are often unavailable in the real world. It is therefore desirable to develop a new business forecasting method that can overcome these deficiencies and limitations.

An accurate and reliable forecasting system must satisfy the following criteria:

1. To quickly react to a significant variation of trend and seasonality in the market;
2. To identify and to smooth purely random noises;
3. To consider the influences of endogenous variables related to the PCB product itself;
4. To take into account the influences of exogenous variables (demographic variables, macroeconomic indicators, competitors, etc.).

Four different domains as described in Yang [13] are studied and they are Time Serial domain, Macroeconomic domain, Downstream demand domain, and Industrial production domain For each domain a factor with the superior value by Gray relation analysis will be selected to represent the input value of NN in these three domains. In order to increase the efficiency of learning in the network, X_1, X_2, and X_3 are chosen to be the inputs of the network; they are Consumer Price Index, Liquid Crystal Element Demand and PCB Production value respectively.

To enable the generation of explicit knowledge, this paper constructs fuzzy rule bases with the aid of Self-organizing Map (SOM) and Genetic Algorithm (GA). The SOM is first used to classify the data and after the classification, Wang and Mendel (WM) model is used to extract fuzzy rules for each cluster. Later on, the fuzzy rule base can be applied as a tool for prediction.

3 Methodology

The major concern of this research is to develop an accurate and reliable forecasting model to meet customer's demand (in terms of the quantity and due date). A forecasting model by combining SOM and fuzzy rule base is illustrated in Figure 1. To predict the future sale in next month, firstly different influencing variables including X_1, X_2, and X_3 are chosen to be the inputs of the network; they are Consumer Price Index, Liquid Crystal Element Demand respectively and PCB Production value in previous month and these information will be fed into the SOM system for classification. Through the step of clustering, the aim of least diversity within a group and most difference among groups could be reached. Secondly, data including different influencing variables and sales from previous periods will be applied by using WM method for fuzzy rule extraction. The number of linguistics terms T used in this fuzzy rule extraction has to be predetermined and the distribution of the membership functions in each dimension of the domain is assumed to be evenly distributed. For ease of interpolation and computational simplicity, the shape of the membership functions used in this rule extraction technique is triangular.

After the fuzzy rule bases corresponding to each class have been constructed, the consistence and effectiveness of the rule base will be checked. Conflict and redundant rules will be eliminated before performing flow time prediction. With this set of fuzzy rules, a human analyst can now examine the behavior of the prediction and changes and modification can be performed if necessary.

3.1 Data Preprocessing by Self-organizing Maps

For sales forecasting in printed circuit board industry, the first step is to classify the available data into different classes so that the data are split into homogeneous sub-populations. Self-organizing Map (SOM) is used to divide the data into sub populations and hopefully reduce the complexity of the whole data space to something more homogeneous.

The training algorithm can be summarized in four basic steps. Step 1 initializes the SOM before training. Best matching unit (BMU) is the neuron, which resembles the input pattern most. On Step 2, best matching unit is determined. Step 3 involves adjusting best matching neuron (or unit) and its neighbors so that the region surrounding the best matching unit will represent the input pattern better. This training process continues until all input vectors are processed. Convergence criterion utilized here is in terms of *epochs*, which defines how many times all input vectors should be fed to the SOM for training.

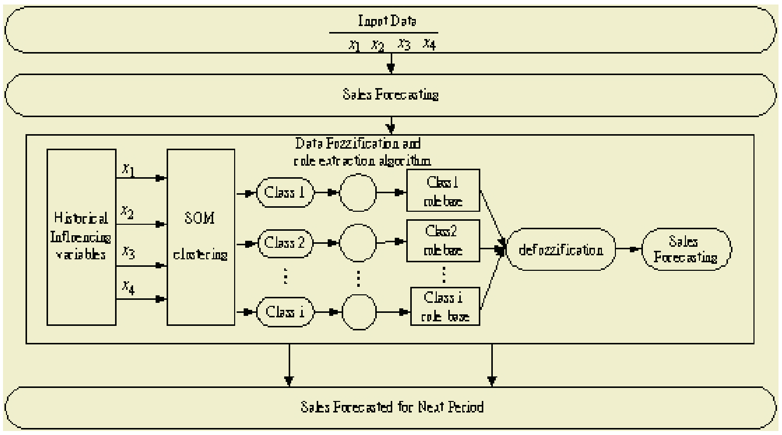


Fig. 1. The diagram of Sale Forecasting by combining SOM and Fuzzy Rule Base.

3.2 The Fuzzy Rule Generation Method

The fuzzy modeling method proposed by Wang and Mendel [15], which is called the WM method, is one of many better-known ones. However, it has two major weaknesses: uniform partition of the domain space and arbitrary selection of the number of partitions. To rectify the second weakness, we propose to hybridize the WM method with genetic algorithms (GAs). Essentially, a simple GA is used to determine the near-optimal number of fuzzy terms for each variable.

3.3 The WM Method

The WM method consists of five steps, as summarized below.

Step 1. Divide the Input and Output Spaces into Fuzzy Regions
Given a set of examples with multiple inputs (m) and single output, denoted as (x_j^k; y^k) where j = 1… m and k = 1… n. Define the universe of discourse of each input variable as [x_j^-; x_j^+] and the output variable as [y^-; y^+] and then divide each universe of discourse into N regions.

The minimal and maximal values of each variable are often used to define its universe of discourse. That is, [x_j^-; x_j^+] = [min (x_j), max (x_j)]. They are also considered to be the center of the left end term and the right end term, respectively. That is, c_{1j} = min (x_j) and c_{Nj} = max (x_j). Accordingly, the other term center, c_{ij}, can be computed as follows:

$$c_{ij} = \min(x_j) + i\,(\max(x_j) - \min(x_j))/(N-1), \text{ where } i = 2, \ldots, N-1. \qquad (1)$$

Step 2. Generate Fuzzy Rules from Given Examples
First, determine the membership degrees of each example belonging to each fuzzy term defined for each region, variable by variable (including the output variable). Secondly, associate each example with the term having the highest membership degree variable by variable, denoted as md_j. Finally, obtain one rule for each example using the term selected in the previous step. The rules generated are "and" rules and the antecedents of the IF part of each rule must be met simultaneously in order for the consequent of the rule to occur. Letting Tx_j be a term selected for variable x_j of an example, a rule could look like:

$$\begin{aligned}&\text{IF } x_1 \text{ is } Tx_1 \text{ (with } md_1\text{) and } x_2 \text{ is } Tx_2 \text{ (with } md_2\text{) and } \ldots \\ &\text{and } x_{ni} \text{ is } Tx_{ni} \text{ (with } md_m\text{) THEN } y \text{ is } Ty \text{ (with } md_y\text{).}\end{aligned} \tag{2}$$

Step 3. Assign a Degree to Each Rule
The rule degree is computed as the product of the membership degree of all variables. Let D^k be the degree of the rule generated by example k. Mathematically,

$$D^k = \prod\nolimits_{\mathrm{j=1,\ldots m\ and\ y}} md_j^k . \tag{3}$$

The degree of a rule generated by an example indicates our belief of its usefulness.

Step 4. Create a Combined Fuzzy Rule Base
When the number of examples is high, it is quite possible that the same rule could be generated for more than one example. These rules are redundant rules. In addition, rules with the same if part but a different then part could also be generated. These rules are conflicting rules. The redundant and conflicting rules must be removed to maintain the integrity of the rule base. This is achieved by keeping only the rule with the highest degree for each fuzzy region: this rule is deemed most useful.

Up to this step, the fuzzy rule base is complete; however, the usefulness of the rule base must be shown using some fuzzy inference method, as introduced in the next step.

Step 5. Determine a Mapping Based on the Combined Fuzzy Rule Base
To predict the output of an unseen example denoted as x_j, the centroid defuzzification formula is used. Accordingly, the predicted output, ŷ, is computed as

$$\hat{y} = \sum_{r=1}^{R} amd^r c^r \Big/ \sum_{r=1}^{R} amd^r \tag{4}$$

where $amd^r = \prod_{j=1,m} md_j^r$; c^r is the center value of the consequent term of rule r; and R denotes the total number of combined rules.

3.4 Simple Genetic Algorithm

To allow for varying the number of partitions for each universe of discourse, a genetic algorithm by Goldberg [4] is introduced to modify the first step of the WM method. The fuzzy terms of each variable will be evolved as described in Yao [14] using genetic algorithm. The binary strings are used to represent chromosomes and the length of each binary string is dictated by the number of variables and the maximally allowed number of fuzzy terms for each variable. For the sake of simplicity, the maximal allowed number of fuzzy terms is assumed to be the same for all variables. The decoding scheme is a simple binary-to-decimal conversion. Each decoded chromosome corresponds to a possible combination of fuzzy terms for all variables involved.

The fitness value, f, is derived from two values produced by the testing process: root mean-squared error (RMSE) and model size (or number of rules). First the model size, R, is normalized by the maximal possible size that is equal to the total number of possible fuzzy terms to the power of the number of input variables, m. Let MaxN and MinN denote the maximum and minimum fuzzy term value, respectively. The total number of possible fuzzy terms is thus (MaxN-MinN+1). A weighted averaging operation is then applied to calculate the fitness value. The inverse of RMSE (multiplied by 106 in order to get a large value) is considered far more important than the model size in this study by assigning the weight ratio of 19 to 1. Mathematically,

$$F = 0.95 \times 10^6 / RSME - 0.05R/(MaxN - MinN + 1)^m. \tag{5}$$

4 Experiment Results and Analysis

This study mainly forecasts the monthly production demand for PCB industry, and the data are from a PCB company in Taiwan from 1999/1 to 2003/12 totally 60 real monthly demand data for testing and training. In addition, there are 15 items of related production index from "Yearbook of Industrial Production Statistics", department of statistics, Ministry of Economic Affairs; "Statistical yearbook of the Republic of China" and "Indices of Consumer Price", from Directorate General of Budget Accounting and Statistics Executive Yuan, R.O.C..

Commercial NN and language software, such as Neural Work Professional II Plus by NeuralWare, MATLAB by Math Works and C++ Builder 5.0 by Borland, are used in order to determine which model would give the best estimates for the given problems.

4.1 Comparison of SOM&WM and Linear Regression

In this research, the input factors of linear regression are liquid crystal element demand (X_1), PCB Production Value (X_2), Consumer Price Index (X_3) and the forecasting of Winter's exponential smoothing (X_4). The regression formula is, $Y = a_1X_1 + a_2X_2 + a_3X_3 + a_4X_4 + b$. The following table shows the errors, estimated by two different measures, i.e., MAPE and MAD, of SOM&WM and linear regression.

Table 1. Comparison of SOM&WM and linear regression.

	MAPE		MAD	
Testing sample	SOM&WM	Linear Regression	SOM&WM	Linear Regression
Avg.	3.13%	7.14%	23991	101508
Std.	2.66%	4.78%	22230	172766
Max	8.58%	16.85%	70334	153667
Min	0.40%	1.11%	96	7188

From the table above, SOM&WM is superior to the linear regression.

4.2 Comparison of SOM&WM and BPN

The three-layered BPN (Back Propagation Network) is applied as a forecasting tool, and the number of neurons in the hidden layer is 2. The sigmoid function is used as the activation function between the input layer and the hidden layer. The Delta-Rule is used as the learning rule, where the learning ratio is 0.4, momentum is 0.5 and epoch size is 100,000. The results are shown in the following table. The performance of SOM&WM is still better than BPN.

Table 2. Comparison of SOM&WM and BPN.

	MAPE		MAD	
Testing sample	SOM&WM	BPN	SOM&WM	BPN
Avg.	3.13%	6.64%	23991	93868
Std.	2.66%	5.63%	22230	162101
Max	8.58%	16.98%	70334	151998
Min	0.40%	0.18%	96	1330

5 Conclusions

In this paper, we presented a new forecasting model to help PCB companies in short- to mid-term sale forecasting for their monthly product demands. The experimental result shows that the performance of SOM&WM is superior to traditional statistical models and Back Propagation Network. Also, the model can obviously provide a very effective and accurate forecast and reduce extra costs produced by other inferior forecasting methods. The SOM&WM provides a promising solution to the forecasting problem for relevant industries.

References

1. Abdel-Aal, R.E., Al-Garni, A.Z.: Forecasting Monthly Electric Energy Consumption in Eastern Saudi Arabia Using Univariate Time Series Analysis. Energy. **22** (1997) 1059-1069
2. Chang, P.C., J.C. Hsieh: A Neural Networks Approach for Due-Date Assignment in a Wafer Fabrication Factory. Int. J. of Industrial Engineering, July (2002)
3. Chow, T.W.S., C.T. Leung: Nonlinear Autoregressive Integrated Neural Network Model for Short-Term Load Forecasting. IEE Proceeding Online no. **19960600** (1996) 500-506
4. Goldberg, D. E: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison Wesley, (1989)
5. Hsu, P.H., Wang, C.H., Shyu, J.Z., Yu, H.C.: A Litterman BVAR Approach for Production Forecasting of Technology Industries. Technological Forecasting & Social Change, **70** (2002) 67-82
6. Kimoto, T. , Asakawa, K.: Stock Market Prediction System with Modular Neural Network. IEEE International Joint Conference on Neural Network, (1990) 1-6
7. Kuo, R.J.: A Sales Forecasting System Based on Fuzzy Neural Network with Initial Weights Generated by Genetic Algorithm. European Journal of Operational Research, **129** (2001) 496-517

8. Law, R., Au, N.: A Neural Network Model to Forecast Japanese Demand for Travel to Hong Kong. Tourism Management, **20** (1999) 89-97
9. Luxh, J. T., Jens, O. Riis, Stensballe, B.: A Hybrid Econometric-Neural Network Modeling Approach fFor Sales Forecasting. The International Journal of Production Economics, **43** (1996) 175-192
10. Srinivasan, D.: Evolving Artificial Neural Networks for Short Term Load Forecasting. Neural computing, **23** (1998) 265-276
11. Tawfik, M.: Sales Forecasting Practices of Egyptian Public Enterprises: Survey Evidence. International Journal of Forecasting, **16** (2000) 359-368
12. Tawfiq, A.S., E.A. Ibrahim: Artificial Neural Networks as Applied to Long-Term Demand Forecasting. Artificial Intelligence in Engineering, **13** (1999) 189-197
13. Yang, T.H.: A Study of Integrating Genetic Algorithm & Neural Network in Forecasting Problem of Printed Circuit Board Industry. Unpublished Master Thesis, Yuan Ze University, Taiwan (2004)
14. Yao, X.: Evolving Artificial Neural Networks. Proceedings of the IEEE, **87** (1999) 1423-1447
15. Wang, L.X.,J., Mendel M.: Generating Fuzzy Rules by Learning from Examples. IEEE Trans. on Syst. Man Cybernet., **22** (1992) 1414-1427

Improving Accuracy of Perceptron Predictor Through Correlating Data Values in SMT Processors

Liqiang He[1,2] and Zhiyong Liu[1]

[1] Institute of Computing Technology, Chinese Academy of Sciences
Beijing 100080, China
hlq@ict.ac.cn, zliu@nsfc.gov.cn

[2] College of Computer Science, Inner Mongolia University
Huhhot, Inner Mongolia 010021, China

Abstract. Simultaneous Multithreaded (SMT) processors improve the instruction throughput by allowing fetching and running instructions from several threads simultaneously at a single cycle. With the pipeline deepen and issue widths increase, the branch predictor plays a more important role in improving the performance of an SMT processor. Many predictors based on neural network, especially on perceptron, are proposed to provide a more accurate dynamic branch prediction than before in the literature. In this paper, we propose an effective method to improve the accuracy of a perceptron predictor through correlating data values in SMT processors. The key idea is using a dynamic bias input, which comes from some information independent on the branch histories (data values for example), to realize the objective of improving accuracy. The implementation of our method is simple, and the predicting latency is not lengthened. Execution-driven simulation results show that our method works successfully on improving the accuracy of a perceptron predictor and increasing the overall instruction throughput of SMT processors.

1 Introduction

Simultaneous Multithreaded (SMT) processors [1, 2] improve the instruction throughput by allowing fetching and running instructions from several threads simultaneously at a single cycle. In SMT processors, functional units that would be idle due to instruction level parallelism (ILP) limitations of a single thread are dynamically filled with useful instructions from other running threads. By allowing fetching instructions from other threads, an SMT processor can hide both long latency operations and data dependencies in a thread. These advantages increase both processor utilization and instructions throughput.

With the pipeline deepen and issue widths increase, the branch predictor plays a more important role in improving the performance of an SMT processor. Recently, some researches have shown that branch prediction is a machine learning problem. Based on the results, some new branch predictors based on neural network, especially on perceptron, are proposed to provide a more accurate prediction than the traditional predictors in the literature. The main obstacles that prevent these predictors being used in practice are the high implementation complexity and the high access latency because they need multiplication and sum operations during their prediction processes. There are researches that have addressed on these issues, and have provided several resolutions such as ahead pipelining [5] and using MAC representation [6]. The first is a technique that overrides the calculation through pipelining and starting ahead the operations. The second is to simplify the sum operations. All these techniques have made the neural network based branch predictors more and more practical.

In this paper, we focus our research on the perceptron predictor, a class of branch predictors that are widely used in today's branch prediction researches for their simplicity and effectiveness. Most of the perceptron predictors are learning machines that produce an output from the inputs through learning from the branch histories. We propose a new method to improve the accuracy of perceptron predictors through correlating data values in SMT processors. Rather than adding a new input vector and a corresponding weight to the original predictor, we use a dynamic bias input to show the effect of the new correlating information to the prediction result. This is because the new information (data values for example) does not depend on the branch histories, and needs not to learn from them. The dynamic bias input is set only according to some instantaneous information during a thread's running. In this means, our method is a technique that fuses an un-learning predicting technique and a learning technique to a single learning machine, and improves the machine's prediction accuracy to a higher level.

Although there are many researches that focus on perceptron predictors, they all use a static bias input to our knowledge. This is the first work that uses a dynamic bias input in a perceptron predictor explicitly.

The main contributions of this paper are: (1) We show that the accuracy of a preceptron predictor can be improved through correlating some information that does not depend on the branch histories, data values for example. And the new input information will not lengthen the predicting latency of the predictor. When the new predictor is used in SMT processors, the overall instruction throughput can be improved accordingly. (2) The new correlated information are realized through using a dynamic bias input to the predictor other than adding a new input vector and a new corresponding weight for the information in the predictor. This dynamic bias input is very successful in improving the accuracy of the predictor.

The rest of this paper is organized as follows. In section 2, we describe the rationale of perceptron predictor and the related works. In section 3, we present our method to improve the accuracy of perceptron predictor in detail. Section 4 and 5 describe the methodology and analyze the simulation results. Finally, section 6 concludes the paper.

2 The Perceptron Predictor and Related Work

2.1 The Idea of the Perceptron

The rationale of a perceptron predictor is shown in Figure 1. It is a very simple neural network. Each perceptron is a set of weights that are trained to recognize patterns or correlations between their inputs and the event to be predicted. A prediction is made by calculating the dot-product of the weights and an input vector. The sign of dot-product is then used as the prediction, each weight represents the correlation of one bit of history (global, path or local) with the branch to be predicted. In hardware, each

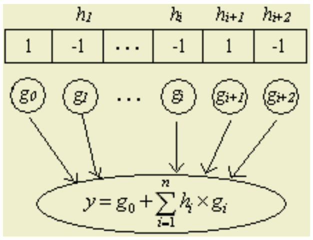


Fig. 1. The perceptron assigns weights to each element of the branch history and makes its prediction based on the dot-product of the weights and the branch history plus a bias weight to represent the overall tendency of the branch. Note that the branch history can be global, local or something more complex.

weight is implemented as an n-bit signed integer, where n is typically 8 in the literature, stored in an SRAM-Array. The input vector consists of 1's for taken and -1's for not taken branches. The doc-product can then be calculated as a sum with no multiplication circuits needed.

2.2 Related Works

The original predictor based on neural network was introduced by Calder [3] to be used in static branch prediction at a program's compiling time. Then D. Jiménez and C. Lin [4] use the perceptron predictor firstly in dynamic branch prediction, and show that the predictor is more accurate than any other known global branch predictors. Although the perceptron predictor has a high accuracy, the implementation of it is very complex due to the multiplication and sum operations needed in a prediction, and the complexity also lengthens the predicting latency of a single branch instruction. Recently, many orthogonal methods are proposed to simplify the complexity and shorten the latency. One is using the Multiply-Add Contribution (MAC) [6] to represent the weights of a perceptron and obtain a simpler hardware implementation. The other is ahead pipelining [5], which hides the most of the delay by fetching weights and computing a running sum along the path leading to each branch. Some other techniques also help to improve the accuracy of a perceptron predictor, such as using pseudo-tag to reduce aliasing impact, skewing weight tables to improve table utilization, and introducing redundant history to handle linearly inseparable data sets.

[7, 8] have shown that a traditional branch predictor can obtain a higher accuracy through combining a value predictor. The value predictor exploits the value locality to predict the value of a register or a memory location, and help to break the data dependence between instructions and let the control-flow to speculatively execute. A branch predictor combined with a value predictor can predict the inputs of branch instructions, and execute branches speculatively according to their predicted inputs.

3 Improving Perceptron Through Correlating Data Values

In this section, we present our method to improve the accuracy of perceptron predictor through correlating data values.

3.1 Rationale

[9] shows that the locality of branches does not come from the branch instructions themselves but the locality of their referenced values, such as the values stored in the registers or the memory. With the locality of data values, a value predictor can be used to predict the outcome of an arithmetic operation or the result of a load/store operation. In our work, we use a value predictor to predict the operands of a branch instruction, and then compute the branch result according to the predicted values. Information about the operator and the register names of operands of a branch instruction are cached in a small table BIT (short for Branch Instruction Table). Due to existing misprediction in a value predictor, the branch prediction result according to the values from the value predictor will not always correct. We use a variable *C*, the first letter of *Confirmation*, to reflect the degree of confirmation about the prediction result. The value of *C* is between 0 and 1.

Because the data values that a branch instruction referenced have no relationship with the histories of branches (neither the global nor the local), the prediction result according to the data values have also no relationship with the histories. So this branch prediction process needs not to learn from the branch histories, and also needs not any training process. The accuracy of the prediction only depends on the accuracy of the value predictor. This feature can be used in a perceptron predictor as a factor to change the bias input (g_0 in Fig.1) dynamically.

In all the existing perceptron predictors, the bias input is determined, and always being set to 1 as in Fig.1 that means the branch preferring to be taken. This static bias input setting is similar as a static branch prediction in the past. It is well known that a static branch prediction scheme has a lower accuracy than the dynamic prediction schemes. So we use a dynamic bias input in a perceptron predictor to improve its prediction accuracy.

A bias input in a perceptron predictor, whether being static or dynamic, must be independent with the branch histories, and needs not any training process. Otherwise it will be another input vector similar as the other vectors in the original predictor. A bias input according to the prediction result based on value predictor just meets this condition. And due to the variance of the prediction result, taken or not taken, confirming or not confirming, the bias input will change accordingly. Based on this feature, we named this input as a dynamic bias input to the perceptron.

With the prediction result of a branch based on value predictor, a dynamic bias input, still using g_0 as in Fig.1, can be calculated with equation (1). In equation (1), *C* is the confirmation parameter, and *P* is the prediction result of the branch based on the value predictor. The value of *P* can be unipolar or bipolar, that is 0/1 in unipolar, or -1/1 in bipolar. In general, the bipolar is used more often as in Fig.1.

$$g_0 = P * C \tag{1}$$

Using the dynamic bias input in a perceptron predictor, the branch prediction can be correlated with some information not depending on the histories of branch, data values for example, and obtains a higher accuracy than the original predictor. With the prediction result based on value predictor and the result based on perceptron of a same branch exist and work together, our method can also be seen as a technique that fuses the traditional branch prediction scheme and the new perceptron schemes.

Comparing with the other fusing techniques, our scheme makes the two candidates working together to produce a higher accurate prediction result rather than using a selector to choose a single result from the two candidates. With a selector, the accuracy of a predictor does not exceed any one of the group's accuracies, whereas with our scheme it can obtain a higher one.

3.2 Implementation Issue

The implementation of our method needs the help from value predictor. A circuit should read the predicted values needed by a branch from the value predictor, and calculate the branch result, *P*, according to the values. The value predictor must record the name of register whose value is predicted. And a new field, *C*, should be added into the value predictor to show the confirmation of a single predicted value. The value of *C* needs to be read out with the corresponding data values, and is used to calculate the result of bias input, g_0, together with the prediction result from the circuit. Other parts of perceptron predictor are implemented as the original one.

Since the value of *C* is between 0 and 1, the calculation of g_0 according to equation (1) needs a float point multiplying operation that will add more complexity to the hardware implementation. A small change of equation (1) can avoid the float point operation simply. For example, we can magnify the value scope of *P* firstly, such as 2 for taken and -2 for not taken, and then set the value of *C* to five levels accordingly, 0, -1, -2, -3, and -4, that represent the five degrees of confirmation (fully trust, a little of trust, unsure, a little of distrust, fully distrust). The degree of confirmation is set according to the accuracy of the value predictor. With these settings, the float point operation can be converted into a sum (for taken) or subtraction (for not taken) operation between two integers.

In addition, the BIT table can be implemented through an SRAM-Array. Once a branch instruction enters the pipeline, it uses the *PC* to index the BIT table, and checks if there is an entry for itself. If it is, the information about the branch, operator and register names of operands, are read out and used to predict the branch outcome. Otherwise, a new entry must be added into the table when the branch instruction enters the ”decode” pipeline stage at next cycle.

Although needing accessing the value predictor and BIT table, and calculating the sum/subtraction of *C* and *P*, our method does not lengthen the predicting latency of perceptron predictor because these operations can be done in parallel with the other sum operations in the original predictor.

4 Methodology

In this study, we focus on the heterogeneous multitasking mode of SMT processor. We modified the SMT simulator (SMTSIM) [10] to implement our new method and to gather detailed statistics of the experiments. This execution-driven simulator emulates unaltered Alpha executables, and models all typical sources of processor latency and conflict. The major simulator parameters are given in Table 1.

Table 1. Simulator parameters.

Parameter	Value
Functional Units	3 FP, 6 Int(4 LD/ST)
Pipeline depth	9 stages
Instruction Queue	32-entry FP, 32-entry Int
Latency	Based on Alpha 21264
Instruction Cache	64KB, 2-way, 64 byte/line
Data Cache	64KB, 2-way, 64 byte/line
L2 Cache (on-chip)	512KB, 2-way, 64 byte/line
L3 Cache (off-chip)	4MB, 2-way, 64 byte/line
I/DTLB, miss penalty	48/128 entry, 160 cycles
Latency(to CPU)	L2 6, L3 18, Mem 80 cycles
branch predictor, miss penalty	O-GEHL, 7 cycles

Table 2. Number of running instructions of each program in every workload.

No.	Workload	Inst. (billion)
1	art,perlbmk	0.3
2	crafty,mcf	1.5
3	equake,mesa	3
4	mgrid,ammp	4
5	bzip2,lucas	5.3
6	parser,twolf	5.3
7	applu,sixtrack	3
8	gcc,facerec	0.5
9	gzip,swim	2
10	art,perlbmk,crafty,mcf	1.5
11	applu,sixtrack,gzip,swim	3
12	bzip2,lucas,parser,twolf	5.3

The BIT table in our experiment has 1024 entries. And the value predictor we used is a 4096-entry stride 2-delta value predictor [11]. A 2-delta stride predictor computes strides as per the stride predictor, but only updates when the last two strides are equal. Due to the value predictor already existing in our simulator, we do not take its hardware cost into account in our experiment. So the total hardware budget needed in our method is the BIT table and a small circuit. The circuit reads data from the value predictor and makes the branch prediction based on them, then calculates the sum or subtraction between the prediction result and the confirmation parameter (C). The prediction result is set to 2 for taken and -2 for not taken, and the value of C is set to 5 levels as described in last section.

The perceptron predictor in our experiment is an O-GEHL branch predictor which is nominated the ”best practice” in the first championship branch prediction [12]. The hardware budget of the predictor is 64kbits.

Our workload consists of eight integer and ten floating point programs from the SPEC CPU 2000 benchmark suite [13]. We compiled each program with GCC with the -O4 optimization and produced statically linked executables. Each program runs with the reference (expect for *bzip2*, which uses the train) input set. From these eighteen benchmarks, we created nine two-thread, and three four-thread workloads. The running instructions each of the program in the workloads are listed in Table 2 according to the method in [14]. All the combinations are otherwise arbitrary. We run the workload in simulator with the original O-GEHL predictor and an updated version of it added by our dynamic bias input, and then compare the performances of them. In all the simulators, predictors are shared by all the threads running simultaneously.

5 Simulation Results

This section presents the simulation results of our experiments. It compares the performances between the original O-GEHL branch predictor and the updated version of it in SMT processors, and also presents the branch misprediction rates and the wrong path instruction fetched rates in the experiment. It uses the serial numbers listed in Table 2 to denote workloads in the figures of this section.

5.1 Processor Performance

Figure 2 shows the overall instruction throughput or IPC (Instruction Per Cycle) in our experiment. We can see that except the *bzip2-lucas-parser-twolf* and *crafty-mcf* workloads all the performances of other workloads with the updated version of O-GEHL predictor are better than that with the original one. The average IPC of the two predictors are 1.89 and 1.96 respectively. The speedup of our updated version over the original predictor is 3.7%.

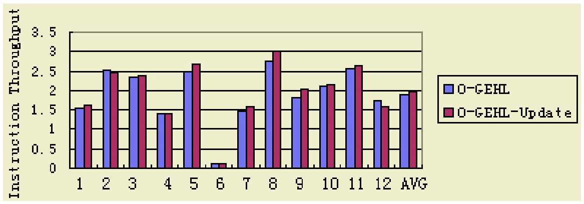


Fig. 2. The overall instruction throughput with the original O-GEHL predictor and our updated version.

The individual performance of every thread in the 2-thread workloads are shown in figure 3(a). The performances of *crafty* and *mcf* in our predictor are a little worse than that in the original one, whereas the performances of others are all improved which shows the fairness of our predictor to the threads in a workload. On average, the instruction throughput of the two predictor in the 2-thread workloads experiment are 0.91 and 0.95 respectively.

Figure 3(b) shows the individual performances of every thread in the 4-thread workloads experiment. The performances of near half of the threads, five in twelve, degrade with our predictor, and other's performances are improved. This shows the effect of

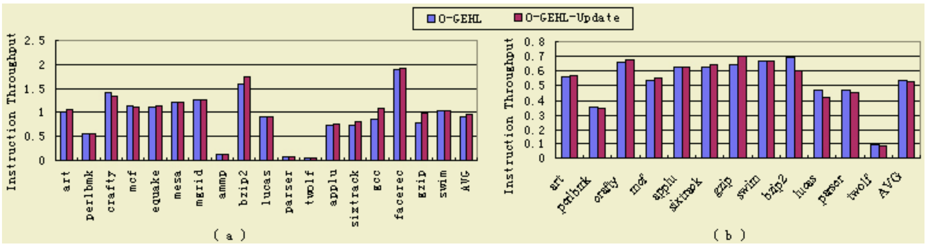


Fig. 3. The individual performance of every thread in 2-thread (a) and 4-thread (b) workloads experiment.

improved branch prediction accuracy on the performance of workload is not so distinct when the number of threads in a workload is too large (≥ 4). The average IPCs of the two predictors for the 4-threads workloads are 0.532 and 0.53. The performance of our predictor is a little worse than the original one.

5.2 Branch Misprediction Rate

Figure 4(a) shows the branch misprediction rates in our experiment with the two predictors. For all the workloads, our updated version of O-GEHL predictor degrades the misprediction rates effectively which also shows the fairness of our method. The highest improvement is obtained by *gzip-swim*, which is 3.43%. The lowest one belongs to *bzip2-lucas*, which is only 0.38%. On average, the misprediction rate of the two predictors are 9.39% and 7.64%.

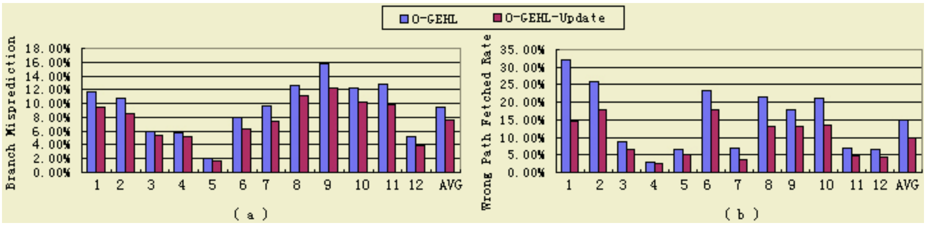


Fig. 4. The branch misprediction rates (a) and instruction fetched rates along the wrong path (b) in our experiment with the two predictors.

5.3 Wrong Path Instruction Fetched Rate

Figure 4(b) shows the instruction fetched rates along the wrong path (WP) in our experiment with the two predictors. Same as in Fig. 4(a), our predictor degrades the fetched rates of all the workloads effectively. The highest improvement, 17.71%, belongs to *art-perlbmk*, and the lowest one belongs to *mgrid-ammp* that is only 0.31%. The average WP fetched rate degrades from 15.02% with the original predictor to 9.76% with our updated version. With the lower WP fetched rate, the branch misprediction penalty in our updated predictor decreases, and more instruction slots can be used by other useful instructions which improves the overall performance of an SMT processor finally.

6 Conclusions

An SMT processor benefits the instruction throughput by allowing fetching and running instructions from several threads simultaneously at a single cycle. Branch predictors based on neural network, especially on perceptron, are proposed recently in the literature, and show the potential to increase the accuracy of prediction. We propose an effective method to improving the accuracy of perceptron predictor through correlating data values during its predicting process. Our contributions in this paper are that we show a perceptron predictor can be improved through correlating some information independent on branch histories, and the new correlated information and prediction process can be fused into the original perceptron through a dynamic bias input. The implementation of our method is easy and it does not lengthen the predicting latency of a perceptron predictor. The execution-driven simulation in SMT environments shows that our method works successfully on improving the accuracy of perceptron branch predictor and the performance of SMT processors.

Acknowledgements

This work is supported by National Natural Science Foundation of China (NSFC) Grant No. 60325205, 60303033, and 60373043.

References

1. Tullsen, D.M., Eggers, S.J., Levy, H.M., et al.: Simultaneous Multithreading: Maximizing On-Chip Parallelism. 22nd Annual International Symposium on Computer Architecture, (1995)
2. Tullsen, D.M.: Exploiting Choice: Instruction Fetch and Issue on a Implementable Simultaneous Multithreading Processor. 23nd Annual International Symposium on Computer Architecture, May (1996)
3. Calder, B., Grunwald, D., Jones, M., et al.: Evidence-based Static Branch Prediction Using Machine Learning. ACM Transactions on Programming Languages and Systems, 19(1) (1997)
4. Jiménez, D., and Lin, C.: Neural Methods for Dynamic Branch Prediction. ACM Transactions on Computer System, **20** (2002) 369-397
5. Jiménez, D.: Fast Path-Based Neural Branch Prediction. 36th International Symposium on Microarchitecture, (2003) 243
6. Seznec, A.: Revisiting the Perceptron Predictor. Technical Report, IRISA, May (2004)
7. Sato, T.: First Step to Combining Control and Data Speculation. In IWIA'98, Oct (1998) 53-60
8. Gonzalez, J., Gonzalez, A.: Control-Flow Speculation through Value Prediction. IEEE Transaction on Computer, **50** (2001) 1362-1376

9. He, L., Liu, Z.: A New Value Based Branch Predictor for SMT Processors. IASTED PDCS 2004, Nov (2004) 775-783
10. Tullsen, D.M.: Simulation and Modeling of a Simultaneous Multithreading Processor. 22nd Annul Computer Measurement Group Conference, December (1996)
11. Hu, S., Bhargava, R., and John, L.K.: The Role of Return Value Prediction in Exploiting Speculative Method-level Parallelism. Journal of Instruction- Level Parallelism, **5**, (2003) 1-C21
12. Seznec, A.: The O-GEHL Branch Predictor. Championship Branch Prediction, The Journal of Instruction Level Parallelism, (2004) http://www.jilp.org/cbp
13. Henning, J.L.: SPEC CPU 2000: Measuring CPU Performance in the New Millennium. IEEE Computer, July (2000)
14. Sherwood, T., Perelman, E., Hamerly, G., and Calder, B.: Automatically Characterizing Large Scale Program Behavior. 10th ASPLOS, October (2002)

A Genetic-Algorithm-Based Neural Network Approach for Short-Term Traffic Flow Forecasting

Mingzhe Liu[1], Ruili Wang[1], and Jiansheng Wu[2], and Ray Kemp[1]

[1] Institute of Information Sciences and Technology, Massey University
Private Bag 11222, Palmerston North, New Zealand
{m.z.liu,r.wang,r.kemp}@massey.ac.nz

[2] Department of Mathematics and Computer, Liuzhou Teachers College,
Guangxi 545000, China

Abstract. In this paper, a Genetic-Algorithm-based Artificial Neural Network (GAANN) model for short-term traffic flow forecasting is proposed. GAANN can integrate capabilities of approximation of Artificial Neural Networks (ANN) and of global optimization of Genetic Algorithms (GA) so that the hybrid model can enhance capability of generalization and prediction accuracy, theoretically. With this model, both the number of hidden nodes and connection weights matrix in ANN are optimized using genetic operation. The real data sets are applied to the introduced method and the results are discussed and compared with the traditional Back Propagation (BP) neural network, showing the feasibility and validity of the proposed approach.

1 Introduction

The demand of predictive traffic conditions to date is becoming more and more essential for both travelers and intelligent transportation systems (ITS) such as route guidance system, signal control, and variable message signs (VMS). This kind of advanced information can provide a real-time way to redistribute traffic resource and reschedule the transportation system temporally and spatially. In this case, the accurate forecasting of short-term traffic conditions is a main concern to researchers. The target of short-term traffic prediction is to forecast traffic flow variables such as volume, speed, or travel time in the range from 5 to 30 minutes [1] into the future.

The study of short-term traffic flow forecasting has attracted attention from various disciplines, both inside and outside of traffic and transportation areas. Various methodologies and techniques have been reported in the literature, including time series models [2],[3], Kalman filter theory [4], regression analysis [5], chaotic theory [6], Markov chain model [7], artificial neural networks (ANN) [8],[9], etc.

Among these models, the use of ANN is expected to be valuable due to its capabilities of approximating a given numeral at any desired accuracy [10] and of solving nonparametric regression problems [11]. The focus on the application of ANN is how to select the network structure [12]. Some input parameters, such as the number of hidden nodes, connection weights, learning rate, activation function, etc. are usually determined empirically.

More recently, an approach of utilizing genetic algorithms (GA) to optimize ANN has become popular [12],[13],[14],[15],[16],[17]. However, evolving both connection weights matrix and hidden neurons with GA has not been found in the literature so far, especially in short-term traffic prediction. Thus, in this paper we employ GA to optimize both connection weights matrix and the number of hidden nodes in order to predict traffic flow. The real data sets are used in the introduced method and the results are discussed and compared with the traditional BP neural network, showing the feasibility and validity of the proposed approach.

2 Methodology

The GAANN model has three layers with m nodes in the input layer, h nodes in the hidden layer, and n nodes in the output layer. Firstly, the model is implemented in order to determine a basic state space of connection weights matrix. Secondly, the number of hidden nodes and connection weights matrix are encoded into a mixed string which consists of integer value and real value. In this paper the used data are divided into three parts: training sample Φ_1, cross-validation sample Φ_2 and testing sample Φ_3. Here we introduce our scheme:

Step 1: Initialize connection weights which are within [-1, 1] for training sample Φ_1. Adjust the weights until the desired tolerance of error ε_1 is obtained. The maximum and minimum of weights are denoted as $u_{\max}$ and $u_{\min}$, respectively. The value of weights are taken within $[u_{\min} - \delta_1, u_{\max} + \delta_2]$, where δ_1, δ_2 are adjustment parameters.

$$\min\ E_i = \frac{1}{2}\sum^{\phi_i} [y_k(t) - \hat{y}_k(t)]^2 < \varepsilon_i \tag{1}$$

where i = 1, 2, 3 which corresponds to three data sets. $\hat{y}_k(t)$, $y_k(t)$ are the desired output and real data, respectively.

Step 2: Encode connection weights and number of hidden nodes. The hidden nodes are encoded as binary code string, 1 with connection to input and output nodes and 0 with no connection. The weights are encoded as float string, with string length $H = m{*}h + h + h{*}n + n$ (m is the number of input nodes, n is the number of output nodes, h is the number of hidden nodes). Each string corresponds to a chromosome, which consists of some gene sections, tabulated as follows:

Table 1. Schematic diagram of encoding chronosome. Part A is encoded in binary type, and other parts in real value. These values change during training period.

1, …, 1	0.2, …, 0.7	0.3, …, 0.1	0.2, … , 0.3	0.9, … , 0.8
A	B	C	D	E

Here: A stands for the number of hidden neurons; B stands for weights between input and hidden neurons; C stands for threshold of hidden neurons; D stands for weights between hidden and output neurons; E stands for threshold of output neurons.

Step 3: Initialize a population of chromosomes. The length L of each chromosome equals to $G + H$, where G is the length of binary code of the number of hidden nodes and H is the length of real-valued code of connection weights.

Step 4: Calculate fitness individually according to Equation 2 below.

$$F = 1/(1 + \min E) \tag{2}$$

Step 5: Copy the highest fitness individual directly to a new offspring and select other individuals by means of the method of spinning the roulette wheel [9].

Step 6: Use basic crossover and mutation operations to the control code, namely, if a hidden node is deleted (added) according to mutation operation, the corresponding control code is encoded 0 (1). The crossover and mutation operators of weights are encoded as follows:

- Crossover operation with probability p_c

$$\begin{aligned} X_i^{t+1} &= c_i \cdot X_i^t + (1 - c_i) \cdot X_{i+1}{}^t \\ X_{i+1}{}^{t+1} &= (1 - c_i) \cdot X_i^t + c_i \cdot X_{i+1}{}^t \end{aligned} \tag{3}$$

where X_i^t, $X_{i+1}{}^t$ are a pair of individuals before crossover, X_i^{t+1}, $X_{i+1}{}^{t+1}$ are a pair of individuals after crossover, c_i is taken as random value within [0, 1].

- Mutation operation with probability p_m

$$X_t^{i+1} = X_t^i + c_i \tag{4}$$

where X_i^t is individual before mutation, X_i^{t+1} is individual after mutation, c_i is taken as random value within ($u_{\min}$ - δ_1 - X_i^t , $u_{\max}$ + δ_2+ X_i^t).

Step 7: Generate the new population and replace the current population. The above procedures (step 4 ~ 7) are repeated until convergence conditions (min $E_2 < \varepsilon_2$ and min $E_3 < \varepsilon_3$) are satisfied.

Step 8: Decode the highest fitness individual, obtain corresponding number of the hidden nodes and connection weights, and output the prediction results.

3 Model Implementation and Results

In order to evaluate the performance of our model, field observation is performed to count traffic volume in the double-lane urban road. In our research we have recorded 10 hours of traffic data during the morning peak period (7:50 – 8:50) between 16 August 2004 and 27 August 2004. The average traffic volume has been obtained and applied to predict short-term traffic flow into the future. Traffic volume is normally viewed as a time series, namely, the problem of prediction can be formulated as the estimation $d(m + t)$, where t is a small time interval, given a set of historical data series say, $d(1)$, $d(2)$, …, $d(m)$, where $d(i)$ represents the value at the i^{th} time step, $1 \leqslant i \leqslant m$. As to short-term traffic flow forecasting, the value of t is recognized less than 30 minutes [1], even equals to 20 seconds into the future for predicting vehicle velocity [20]. Since a shorter time interval and acceptable prediction accuracy are very necessary for traffic information systems, in this paper the time interval t is set as 1 minute. Thus we can validate performance of our model more microscopically.

In the simulations a three-layered BP neural network is first employed to estimate basic state space of connection weights. The minimum and maximum values of weights are obtained, -1.21 and 0.96, respectively. Let δ_1 = -0.09 and δ_2 = 0.04, therefore, the range of weights is assumed to be within [-1.3, 1.0]. The number of input neurons are 4 and the range of hidden nodes is assumed to be within [2, 6]. The activation function adopted here from input to hidden layer is Sigmoid, while from hidden to output layer is Purelin function. For the proposed hybrid neural network, the following system parameters in Table 2 are applied to training sample and prediction:

Table 2. Tabulated required model parameters.

Name of Variables	Value	Name of Variables	Value
Training Sample Φ_1	40	Number of Output Nodes	1
Validation Sample Φ_2	12	Learning rate	0.05
Testing Sample Φ_3	8	Momentum	0.04
Error $\varepsilon_1$of Sample Φ_1	0.05	Crossover Probability p_c	0.80
Error $\varepsilon_2$of Sample Φ_2	0.05	Mutation Probability p_m	0.05
Error $\varepsilon_3$of Sample Φ_3	0.05	Population	50
Number of Input Nodes	4	Iteration Number	1000

For the purpose of comparison with other neural network models, such as the basic BP neural network, four types of errors, which are commonly found in many papers discussing these models, are also used here. Four types of errors are the mean root of squares error (MRSE), the mean absolute percentage error (MAPE), the maximum absolute percentage error (MAXAPE), the minimum absolute percentage error (MINAPE) [18].

The values of errors are tabulated in table 3. The used parameters are the same in both two methods. The MRSE of the GA-based neural network and the traditional one are 1.16 and 1.82, respectively. The MAPE of the GA-based neural network is 2.05, which is better than that of the traditional one. We can observe that the capability of approximation of this model is better than the traditional one. In Fig. 1 the prediction of the next 60 minutes by the GAANN is compared to the true value and the traditional BP model. It shows that the GAANN model has a better capability of approximating the true data than that of the traditional BP method. However, in peak values GAANN method is not satisfied. It suggests that our model should be improved and prediction horizon should be extended to 2, 5, or 15 minutes in order to validate prediction capability in our model.

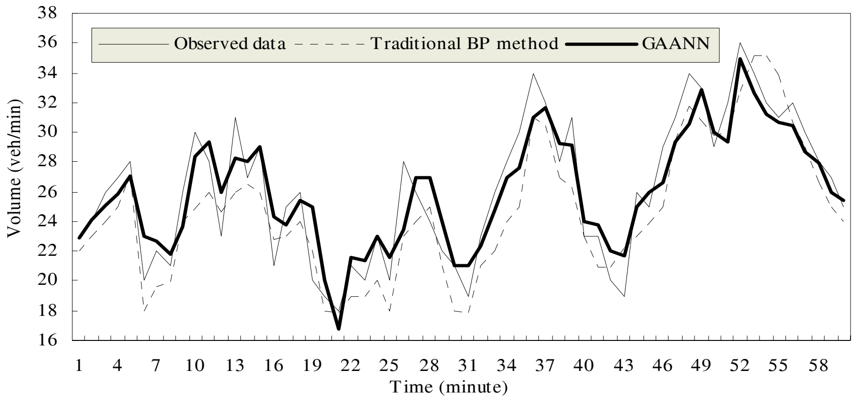


Fig. 1. Traffic volume plot of actual, GAANN and traditional BP method for a 1 minute prediction horizon.

Table 3. Error comparisons of GA-based neural network and traditional neural network.

	GAANN	Traditional BP Network
MRSE	1.16	1.82
MAPE	2.05	3.97
MAXAPE	2.67	7.04
MINAPE	0.032	0.085

4 Conclusions

In this paper, a GA-based neural network approach has been proposed for short-term traffic flow forecasting. Network structure is optimized and connection weights are adjusted through the implementation of genetic operators. The experiment with real traffic data shows that the predictive performance of the proposed model is better than that of the traditional BP neural network. However, in peak values our model does not exhibit perfectly. The consideration of further improvements in model performance should include the following factors such as different time windows (input nodes, in this paper the number of input nodes equal to 4), different prediction horizon (2, 5, or 15 minutes), crossover and mutation operators, classification of data sets, etc. In fact, this work is now under progress.

Acknowledgements

The support of the Massey University Research Fund and the ASIA 2000 Foundation High Education Exchange Programme (HEEP) is a gratefully acknowledged.

References

1. Vlahogianni, E.I., Golias, J.C. and Karlaftis, M.G.: Short-term Traffic Forecasting: Overview of Objectives and Methods. Transport Review, **24** (2004) 533-557
2. Smith, B.I. and Demetsky, M.J.: Traffic Flow forecasting: Comparison of Modeling Approaches. Journal of Transportation Engineering, **4** (1997) 261-265
3. William, B.M.: Modeling and Forecasting Vehicular Traffic Flow as a Seasonal Stochastic Time Series process. Doctoral dissertation. Department of Civil Engineering, University of Virginia, Charlottesville (1999)
4. Okutani, I. and Stephanides, Y.I.: Dynamic Prediction of Traffic Volume through Kalman Theory. Transportation Research Part B, **1** (1984) 1-11
5. Sun, H., Liu, H., Xiao, H., He, R., and Ran, B.: Short Term Traffic Forecasting Using the Local Linear Regression Model. Journal of Transportation Research Board, **1836** (2003) 143-150
6. Hu, J., Zong, C., Song, J., Zhang, Z. and Ren, J.: An Applicable Short-term Traffic Flow Forecasting Method Based on Chaotic Theory. Proceedings of the 2003 IEEE Intelligent Transportation Systems, **1** (2003) 608 – 613
7. Yu, G., Hu, J., Zhang, C., Zhuang, L. and Song, J.: Short-term Traffic Flow Forecasting Based on Markov Chain Model. Proceedings of The 2003 IEEE Intelligent Vehicles Symposium (2003) 208 - 212
8. Dochy, T., Danech-Pajooh, M. and Lechevallier, Y.: Short-term Road Forecasting Using Neural Network. Recherché-transports-Securite. English issue, **11** (1995) 73-82

9. Messai, N., Thomas, P., Lefebvre, D. and Moudni, A.E.: A Neural Network Approach for Freeway Traffic Flow Prediction. Proceedings of The 2002 IEEE International Conference on Control Applications, Glasgow, Scotland, U.K. (2002) 18-20
10. Hornik, K., Multilayer Feedforward Networks Are Universal Approximates. Neural Networks, 2 (1989) 359-366
11. White, H.: Connectionist non-parametric regression: Multilayer Feed Forward Networks Can Learn Arbitrary Mapping. Neural Networks, **3** (1990) 535-549
12. Gallant, P.J. and Aitken, J.M.: Genetic Algorithm Design of Complexity-controlled Time-series Predictors. Proceedings of the 2003 IEEE XIII Workshop on Neural Networks for Signal Processing
13. Tian, L. and Noore, A.: Evolutionary Neural Network Modeling for Software Cumulative Failure Time Prediction. Reliability Engineering and System Safety, 87 (2005) 45-51
14. Pedrycz, W.: Heterogeneous Fuzzy Logic Networks: Fundamentals and Development Studies. Neural Networks, IEEE Transactions on, **15** (2004) 1466-1481
15. Ahmad, A.L., Azid, I.A., Yusof, A.R., Seetharamu, K.N.: Emission Control in Palm Oil Mills Using Artificial Neural Network and Genetic Algorithm, Computers and Chemical Engineering, **28** (2004) 2709-2715
16. Niska, H., Hiltunen, T., Karppinen, A., Ruuskanen, J. and Kolehmainen, M.: Evoling The Neural Network Model for Forecasting Air Pollution Time Series. Engineering Application of Artificial Intelligence, **17** (2004) 159-167
17. Kim, G., Yoon, J., An, S., Cho, H. and Kang, K.: Neural Network Model Incorporating a Genetic Algorithm in Estimating Construction Costs. Building and Environment, **39** (2004) 1333-1340
18. http://www.spss.com/la/productos/DecisionTime/decisiontime2.htm, last accessed on 20 October (2004)
19. Hopfield, J.J.: Neural Networks and Physical Systems with Emergent Collective Computation Abilities. Proceedings of The National Academy of Science, USA (1982) 2554-2558
20. Dia, H.: An Object-Oriented Neural Network Approach to Short-Term Traffic Forecasting. European Journal Of Operational Research, **131** (2001) 253-261

Self-organizing Map Analysis Consistent with Neuroimaging for Chinese Noun, Verb and Class-Ambiguous Word

Minghu Jiang[1,3], Huiying Cai[1], and Bo Zhang[2]

[1]Lab of Computational Linguistics, Dept. of Chinese Language
Tsinghua University, Beijing 100084, China
[2]Dept. of Computer, Tsinghua University, Beijing 100084, China
[3]Creative Base of Cognitive Science, Tsinghua University, Beijing 100084, China
jiang.mh@tsinghua.edu.cn

Abstract. In the paper we discussed the semantic distinction between Chinese noun, verb, and class-ambiguous word by using SOM (self-organizing map) neural networks. Comparing neuroimaging method with neural network method, our result shows neural network technique can be used to study lexical meaning, syntax relation and semantic description for the three kinds of words. After all, the response of human brain to Chinese lexical information is based mainly on conceptual and semantic attributes, seldom uses Chinese syntax and grammar features. Our experimental results are coincident with human brain's neuroimaging, our analysis will help to understand the role of feature description and relation of syntax and semantic features.

1 Introduction

An important feature of human intelligence is the use of language, which makes human brain grow upward. Kim divided the process of language acquisition into four parts [1], early speech perception, word recognition to word learning and the acquisition of grammatical inflections. Because languages consist of ordinal words which are based on syntax and semantic rules, many researchers are exploring the representation and operation of words in human brain, seeking their construction principle and computational theory. Although neuroimaging methods by using localization of cognitive operations within the human brain can be applied to studies of neural networks, conventional syntax techniques are ineffective in natural language processing due to a lack of semantic understanding of relevance. In order to construct the language hierarchical structure model for natural language understanding, we need to know-how the lexical conceptual map can be imitated with the aid of neural networks; how it simulates the conceptual learning, conceptual and semantic relations, word order and grammar rules; how background knowledge is expressed; how conceptual description of lexical features appropriates with human brain's processing.

In fact, no matter neuroimaging or neural network method, the displayable analysis for the lexical feature data can reveal the similarity extent among object-attribute, relation structure and semantic topology relation. This paper addresses the comparison of both methods, we discuss the semantic distinction between Chinese conceptual words by using SOM neural networks, it shows neural network technique can be used to study lexical meaning, syntax relation and semantic description for nouns, verbs, and class-ambiguous words. Our analysis will help to understand the role of feature description and relation of syntax and semantic features.

2 The Self-organizing Map (SOM)

SOM is based on research of physiology and brain science which is proposed by Kohonen [2]. By using self-organized learning, the network enables the similar nerve cell in function to be nearer, the different nerve cell in function to be more separate. During learning process, some ruleless inputs are sorted automatically and enable the weight distribution to be similar to input's probability density distribution. SOM consists of input layer and output layer, which is constructed by competitive learning algorithm. Each nerve cell in input layer is linked by the weight $W_{i,j}$ to each nerve cell of output layer, the nerve cells within the area $N(r)$ around the winner nerve cell r in output layer obtain excitement in different degree, the nerve cells besides $N(r)$ are restrained. The area of $N(r)$ is iteration number t's monotony descend function, finally there only remains one nerve cell, it reflects the attribute of a kind of samples. SOM learning process [2]: When t=0, input sample $X = \{X_i \in \Re^p : i = 1,2,\cdots,n\}$, initial weight is put: $\{W_{i,j}, i, j=1,2,\ldots,m\}$. When $t<T_{\max}$, randomly select $X_i(t)$ in X set:

$$\text{Find out: } r = \arg\min_s \{\|X_i(t) - W_s(t)\|\} \tag{1}$$

$$\begin{aligned} \text{Iteration: } & W_s(t+1) = W_s(t) + \alpha_t \cdot e^{-dist(r,s)^2/\sigma_t^2}[X_i(t) - W_s(t)], && \forall s \in N_t(r), \\ & W_s(t+1) = W_s(t), && \forall s \notin N_t(r). \end{aligned} \tag{2}$$

$$\text{Update: } t{+}{=}1,\ N_t = N_0 - t(N_0-1)/T_{\max},\ \alpha_t = \alpha_0(1 - t/T_{\max}),\ \sigma_t = \sigma_0 - t(\sigma_0 - \sigma_f)/T_{\max}. \tag{3}$$

Here, m is output array size, $T_{\max}$ is the max iterative number, N_0 is initial neighbor threshold, α_0 is initial learning rate, σ_0 and σ_f are the control parameter of step length, dist(r, s) is a distance between node r and node s in the output array. N(r) and α_t are iteration number t's monotony descend function.

3 Experimental Results

Description for experiment data: Experimental data for input features of 68 words are based on "Dictionary for Modern Chinese Syntax Information" [3] and "Hownet"[4], which are described according to their syntax and semantic attributes. By using SOM neural network to train the 68 Chinese words including nouns, verbs and class-ambiguous words, we observed the learning map results for the three kinds of words , and compared our map results of neural networks with the fMRI experimental results of Li Ping et al [5].

Experiment 1: 50-dimension syntax features: the features are extracted from Dictionary for Modern Chinese Syntax Information [3] , which is shown as follows: count n., amount n., front n., back n., front v., subject, object, adv., location, copulative, auxil. v., trend v., form v., double objects, class-ambiguous sentence, back time measure word, plural subject, only as predicate, only as complement, molde v., etc. Experimental parameters of SOM are set as: α_0=0.9, N_0=29, m=30, 50000 iterations, the experimental result of SOM for nouns (tag: ▲), verbs (tag: ○) and class-ambiguous words (tag: ●) is shown in Fig. 1.

Table 1. Experimental data: nouns, verbs and class-ambiguous words [5]

23 nouns (serial number: 1-23)	22 verbs (serial number: 24-45)	23class ambiguous words (serial number: 46-68)
Changliang, daolu, dianying, ertong, fangan, feiji, gongzi, guanzhong, gushi, haiguan, hetong, jiaoshi, jiemu, lvke, meitan, shangchang, shangdian, shanqu, shipin, waihui, wewnzhang, zuqiu	Biancheng, chaoguo, chulai, danren, dapo, dida, duode, gaohao, gaosu, jiancheng, jiaohuan, kandao, liuxia, qianding, quxiao, shuli, xiajiang, yudao, zhidao, zhongzhi, zhuazhu, zuochu	Bianhua, bianji, daibiao, daoyan, heying, huafei, huihua, jianyi, jiaoxun, jihua, jilu, lingdao, mingling, renshi, tongzhi, weixie, xuyao, yaoqiu, yingxiang, zhishi, zhubian, zongjie, zuzhi

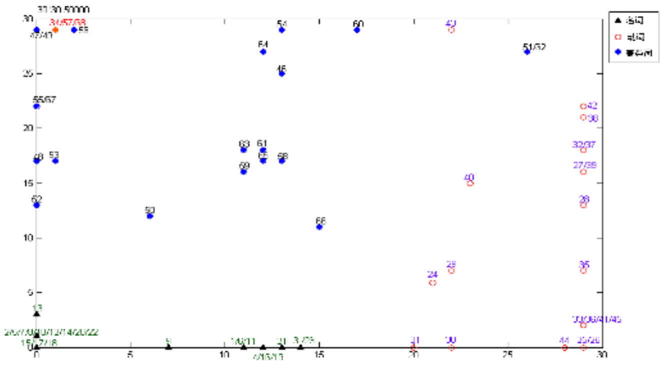


Fig. 1. SOM of 50-dimension syntax features (50*50000*68).

Experiment 2: 132-dimension semantic feature: classified ability of saytax features is too strong, however, described ability of saytax features is too rough. On one hand many different words are described in same form and are mapped to same nerve cell, on the other hand, there is an obviously discrepancy between the POS (part-of-speech) of neural networks recognised words and human brain. Therefore, we incorporated and cancelled some features in saytax, at the same time, we selected appropriate conceptal layers from Hownet extracting semantic features. Chinese semantic attributes include semantic sort, basic acceptation, explicative acceptation and semantic arrangement in pairs or groups etc. According to Hownet, we adopted more specific and extended semantic features and obtained 132-dimension semantic features, which learning result of SOM for nouns, verbs and class-ambiguous words is shown in Fig. 2. The parameters of SOM is set the same as experiment 1 of 8000 iterations.

Experiment 3: 64-dimension features: We incorporated and cancelled some saytax and semantic features, and extracted 29-dimension syntax features and 35-dimension semantic features, thus obtained 64-dimension features, which learning result of SOM for nouns, verbs and class-ambiguous words is shown in Fig. 3. The parameters of SOM is set the same as experiment 1 of 10000 iterations.

Experiment 4: 140-dimension features: based on experiments above, we farther weaken th role of syntax features, and strengthen the role of semantic features, which

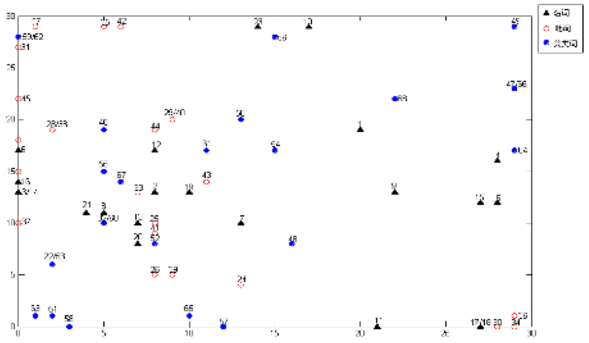


Fig. 2. SOM of 132-dimension semantic features (132*8000*68).

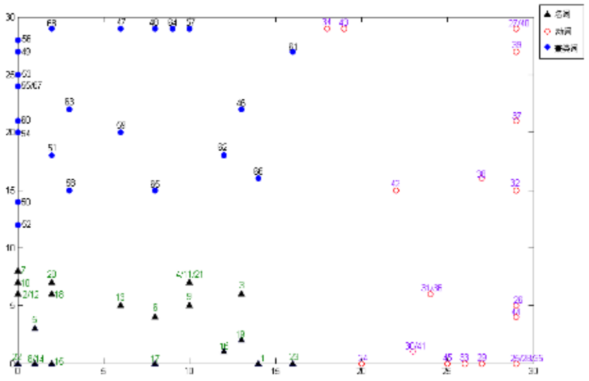


Fig. 3. SOM of 64-dimension merging features (29-dimension syntax features, 35-dimension semantic features, 64*10000*68).

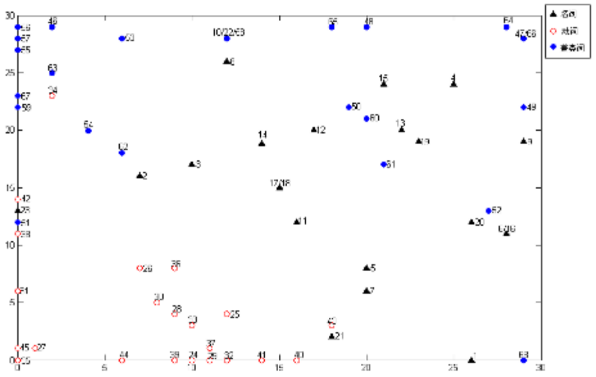


Fig. 4. SOMof 140-dimension merging features (132-dimension semantic features, 8-dimension syntax feature, 140*12000*68).

experimental data are 132-dimension semantic features and 8-dimension syntax features, the learning results for nouns, verbs and class-ambiguous words is shown in Fig. 4. The parameters of SOM is set the same as experiment 1 of 12000 iterations.

Analysis and discussion: our experimental results show that feature description plays an important role in the map area of the three kinds of words: if we selected completely the syntax features (Fig. 1), then the three kinds of words are mapped to three different (non-overlap) areas. If we completely selected the semantic features (Fig. 2), then the mapped areas of the three kinds of words are overlapped mutually, which can not be classified simply. Chinese is different from English, it has the following attributes and characteristics:

Chinese sentence has not the variation of tense and morphology. The sentence structure and phrase structure are basically identical in Chinese, which is "realization relation" between phrase and sentence. There is no simple relationship in one corresponding to one between Chinese POS and syntax components. Chinese syntax components are layer upon layer, the variations of words and phrases in sequence have agility and levity. In fact the response of human brain to Chinese lexical information is based mainly on conceptual and semantic attributes, seldom uses Chinese syntax and grammar features, which is coincident with our experiments. When we strengthen the role of syntax features, and weaken the role of semantic features, the overlapping of the mapped distributing areas for the three kinds of words can disappear as shown in the Fig. 3. Whereas, when we weaken the role of syntax features, and strengthen the role of semantic features, the overlapping of the mapped distributing areas for the three kinds of words is increased as shown in the Fig. 4. According to LiPing's experiments [5], the response of human brain to Chinese nouns, verbs and class-ambiguous words, especially the response to Chinese nouns and verbs is not the same as English words which have a obvious difference, there is a stronger response to the class-ambiguous words. If we adpoted artificial neural networks to simulate the human brain's recognition of nouns, verbs and class-ambiguous words, the distributing areas of mapped results for which should be overlapped. According to experiments above, our experimental results as shown in Fig. 2 and Fig. 4 are closer to that of LiPing [5]. When human brain recognises some word, much information is processed, our experiments are only simple simulations. Therefore, we are seeking for more proper feature descriptions and algorithms to approximate human brain's processing.

4 Conclusions

Chinese is different from English, Chinese sentence has not the variation of tense and morphology. The response of human brain to Chinese lexical information is based mainly on conceptual and semantic attributes, seldom uses Chinese syntax and grammar features. Our experimental results show that feature description plays an important role in the map area of nouns, verbs and class-ambiguous words: If the semantic features are only selected, then the mapped areas of the three kinds of words are overlap mutually, they can not be classified simply. When we strengthen the role of syntax features, and weaken the role of semantic features, the overlapping of the mapped distributing areas for the three kinds of words is decreased or nearly disappear. If the syntax features are only selected, then the three kinds of words are completely mapped at three different (non-overlap) areas. Our experiments is coincident with human brain's neuroimaging.

References

1. Kim, P.: Theories of Early Language Acquisition. Trends in Cognitive Sciences, **1** (1997) 146-153
2. Kohonen, T.: The Self-organizing Map. Proceedings of the IEEE, **78** (1990) 1464-1480
3. Yu, S.: Dictionary Explanation in Detail for Modern Chinese Syntax Information. 2nd edn Tsinghua University Press, Beijing (2003)
4. Dong, Z., Dong, Q.: Hownet. http://www.keenage.com
5. Li, P., Jin, Z., Tan, L.H.: Neural Representations of Nouns and Verbs in Chinese: an fMRI Study, **21** (2004) 1533-1541

Self-organizing Map Analysis of Conceptual and Semantic Relations for Noun

Minghu Jiang[1,2], Chengqing Zong[3], and Beixing Deng[4]

[1] Lab of Computational Linguistics, Tsinghua University, Beijing, 100084, China
[2] Creative Base of Cognitive Science, Tsinghua University, Beijing, 100084, China
jiang.mh@tsinghua.edu.cn
[3] State Key Lab of Pattern Recognition, CAS, Beijing, 100080, China
[4] Dept. of Electronic Eng., Tsinghua University, Beijing, 100084, China

Abstract. In this paper, we analyzed self-organizing map of conceptual and semantic relations for noun, discussing the semantic distinction between conceptual nouns for natural language processing and syntax acquisition, summarizing the lexical meaning and a detailed description of semantic lexical tagging of nouns. Our result reflects the noun-attribute associations and focuses on the conceptual relationships. By using several features, map models provide an operational definition of the conceptual nouns distinction.

1 Introduction

An important feature of human intelligence is the use of language consisting of symbols (words). Many researchers are exploring the lexical representation and operation in human brain and seeking their construction principle and computational theory. It is very useful for natural language understanding if there is semantic understanding of relevance. We need to know how the learning of concepts is simulated, how conceptual and semantic relations can be mapped with the aid of neural networks, and how the multi-disciplinary perspectives of cognitive neuroscience and experimental psycho-linguistics are used to construct neural network models to deal with the hierarchical structures based on the grammar and semantic rules. As a rule, the semantic features of conceptual nouns are described in high dimension vectors. For expedient comparison of neuroimaging localization of cognitive operations [1] and syntactic processing with pattern recognition technology, we adopted the feature compression technologies which map high dimension features to low dimension, and hold enough main information to distinguish the sorts among conceptual nouns. Multi-principal component analytical method is the classical statistical technology of data analysis and feature compression, which is of the function of extraction principal features, restraint noisy and drop dimension [2]; Kohonen's SOM (self-organizing map) nonlinear drop dimension processing enables weight vector to approximate the probability distribution of feature data and displays a topography structure ordinal logic diagram in a 2-dimension array plane [3], which displayable analysis for the feature data can reveal the similar extent among objects, relation structure and semantic topology relation.

2 Feature Data Compress and Map

2.1 The Self-organized Map [3]

Kohonen's SOM is based on research of physiology and brain science [3], the self-organized learning of which enables the similar kinds of nerve cells in function to be nearer and the different kinds of nerve cells in function to be more separate. SOM consists of input layer and output layer, the training of weight $W_{i,j}$ is finished by competitive learning. The black nerve cell r in the center is the winner, around r the nerve cells within the area $N(r)$ obtain excitement in different degree, the nerve cells besides $N(r)$ are restrained, as shown in Fig. 1(a).

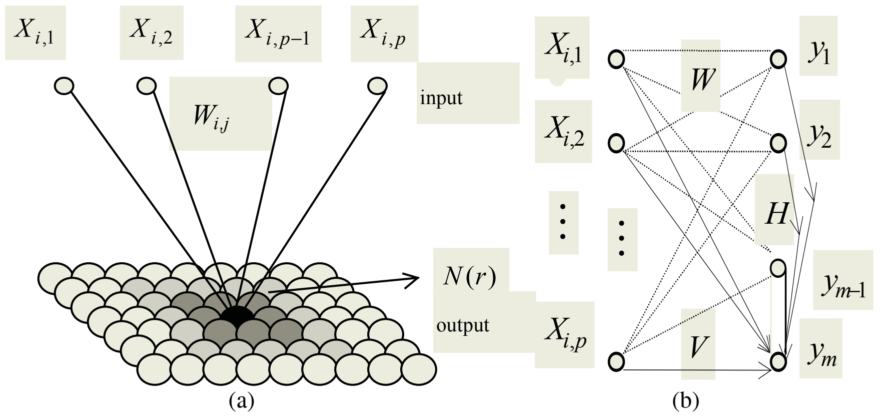


Fig. 1. (a) SOM Neural Network (b) APEX Neural Network.

When t=0, input classless sample $X = \{X_i \in \Re^p : i = 1,2,\cdots,n\}$, initial weight is put: $\{W_{i,j},\ i, j=1,2,\ldots,m\}$. When t<$T_{\max}$, randomly select $X_i(t)$ in X set:

$$\text{Find out: } r = \arg\min_s \{\|X_i(t) - W_s(t)\|\} \tag{1}$$

$$\begin{aligned} \text{Iteration: } & W_s(t+1) = W_s(t) + \alpha_t \cdot e^{-dist(r,s)^2/\sigma_t^2}[X_i(t) - W_s(t)], \quad \forall s \in N_t(r), \\ & W_s(t+1) = W_s(t), \qquad\qquad \forall s \notin N_t(r). \end{aligned} \tag{2}$$

$$\text{Update: } t\text{+=}1,\ N_t=N_0-t(N_0-1)/T_{max},\ \alpha_t = \alpha_0\,(1-t/T_{max}),\ \sigma_t = \sigma_0 - t(\sigma_0 - \sigma_f)/T_{max}. \tag{3}$$

Here, m is output array size, $T_{\max}$ is the max iterative number, N_0 is initial neighbor threshold, α_0 is initial learning rate, σ_0 and σ_f are the control parameter of step length, $dist(r, s)$ is a distance between node r and node s in the output array. $N(r)$ and α_t are iteration number t's monotony descend function.

2.2 Principal Component Feature Extraction

Adaptive principal component extraction (APEX) [2] remains eigen vectors which correspond to m maximum eigenvalues of covariance matrix of input data, canceling the eigen vectors which correspond to lesser eigenvalues, and maps n dimensions feature to m dimensions. Assumed that Mx is mean vector of input samples, the covariance (positive definite) matrix of samples is C_x, P order eigenvalues $\lambda_1 > \lambda_2 > \cdots > \lambda_p > 0$, e_h is an orthogonal unit-eigen vector, $e_h^T e_h = 1$, $C_x e_h = \lambda_h e_h$, $h = 1,2,\cdots,p$. APEX is its statistic property unknown by self-organnized learning to extract principal features and by weight of the former m-1 neural nodes to recursively calculate weight of m^{th} neural node, as shown in Fig.1 (b), which is shown as follows:

$$Y(t) = W(t)X(t) \tag{4}$$

$$y_m(t) = V(t)X(t) - H(t)Y(t) \tag{5}$$

here, input X is a p-dimension sample, $Y=[y_1,\cdots,y_{m-1}]^T$ is the former m-1 output nodes, $W=[e_1,\cdots,e_{m-1}]^T$ is a weight matrix which connects input X to the former m-1 output nodes, V is weight vector which connects input nodes to m^{th} neural node, H is weight vector which connects the former m-1 output nodes to m^{th} output node, $\lambda_i = e_i^T C_x e_i = E\{y_i^2\}$, when W converges, only V and H are trained, the t^{th} iteration is shown as follows [2]:

$$V(t+1) = V(t) + \beta(t)(y_m(t)X^T(t) - y_m^2(t)V(t)) \tag{6}$$

$$H(t+1) = H(t) + \beta(t)(y_m(t)Y^T(t) - y_m^2(t)H(t)) \tag{7}$$

Here, $\beta = 1/(M \cdot E\{y_{m-1}^2(t)\})$ (M is a constant). When W has converged to the former m-1 principal components $[e_1, e_2, \cdots, e_{m-1}]^T$, then V will converge to e_m. The training step is: m=1, V and H are iterated by Eqs(4)~(7) until the changes of V and H is less than a threshed, m is increased and continues iteration Eqs(6), (7) until the number of dimensions are satisfied. After convergence $Y=[W, V]X$ maps p dimensions' vector to m dimensions' one.

2.3 Self-organizing Semantic Maps (SOSM) [5]

SOSM is self-organizing feature maps for the class-extended input feature [5]. Assumed that X is a p-dimension sample, after the classes are extended, the samples mode is: $\hat{X} = \{\hat{X}_i = \begin{bmatrix} \alpha X_{s,i} \\ X_i \end{bmatrix}\} \in \Re^{c+p}$. Here, X_i is the p-dimension mode vector, $X_{s,i}$ includes the class information (c classes), and $\alpha < 1$ is to weaken the importance of the class-extended information.

2.4 Self-organizing Maps for Detailed Feature (SOMDF)

Because the described noun attribute is quantized too rough for the training data of SOM, for example, although aircraft carrier (G) and frigate (I) are classified into one class, the difference of both volumes is very large. Therefore, their feature attributes should be described more detailed as shown in Table 1.

3 Experimental Results

Our feature description of 16 kinds of conceptual nouns in detail and experimental data for SOMDF are shown in Table 1, the other experiment data transform the data of Table 1 into binary values, and the parameters are set α_0=0.9, σ_0=4.0, σ_f=0.5, N_0=9, m=10, $T_{\max}$=10000, α=0.2.

During the learning process of SOM, weight updating is not only for the excited nerve cell, but also for those nerve cells within around N_r area at the same time, within which the more neighboring nerve cells can be reflected in the N_r area. Therefore, the network has a high ability to tolerate noisy and aberrance of the samples, which learning result enable the nearer samples of the input space to be mapped to nearer nodes in output layer. Our experimental results show that if the input samples have several classes, then according to their probability distribution, these samples are mapped to different area in output layer, one area representing one same class samples. We can learn from comparison as shown in Fig 2, APEX algorithm is similar to SOM for clustering ability, reflecting the high dimension features of the input samples are mapped to some low dimension spatial area and these similar features' samples in high dimension space are mapped to neighboring nodes in output layer. Because the conventional binary feature descriptions are too rough, many objects are mapped to the same output node, for example, aircraft carrier (G), chaser (H) and frigate (I), which features are described the same for the training data of conventional SOM, and output is mapped to the same node, as shown in Fig 2(a). The class extended information (16 classes) is added to the input features, as shown in Fig2(c), showing the same input features are also mapped to different output nodes. SOMDF can obtain fine classification for the described object as shown in Fig. 2(d).

Table 1. Detailed Feature Description for Output Objects (A: bicycle, B: motorcycle, C: microbus, D: coach, E: train, F: tank, G: aircraft carrier, H: chaser, I: frigate, J: huge passenger liner, K: passenger liner, L: lugger, M: helicopter, N: scout, O: huge airliner, P: airliner).

Feature Description \Output Object	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
Small	0.8	1.0	1.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
Middle	0.0	0.0	0.0	1.0	0.0	0.9	0.0	0.0	0.0	0.0	1.2	0.0	0.9	0.9	0.0	0.0
(Very)Big	0.0	0.0	0.0	0.0	1.1	0.0	1.2	1.0	1.0	0.9	0.0	0.0	0.0	0.0	1.0	0.9
Seaway	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0
Landway	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Airway	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0	1.0
Used in Army	0.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0
Used in Civilian	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0	1.0	0.0	1.0	1.0
Drive	0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	1.0	1.0	1.0	1.0
Non-Drive	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
Cheap	0.8	0.9	1.0	1.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.9	0.0	0.0	0.0	00
Moderate	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.1	0.8	0.0	1.0	1.2	0.0	1.2
Expensive	0.0	0.0	0.0	0.0	0.9	0.0	1.2	1.0	0.9	0.0	0.0	0.0	0.0	0.0	1.1	0.0
High Speed	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.2	0.8
Middle Speed	0.0	1.2	1.1	1.0	1.1	1.0	0.8	0.9	1.1	0.8	0.8	0.0	1.1	0.0	0.0	0.0
Slow Speed	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	00

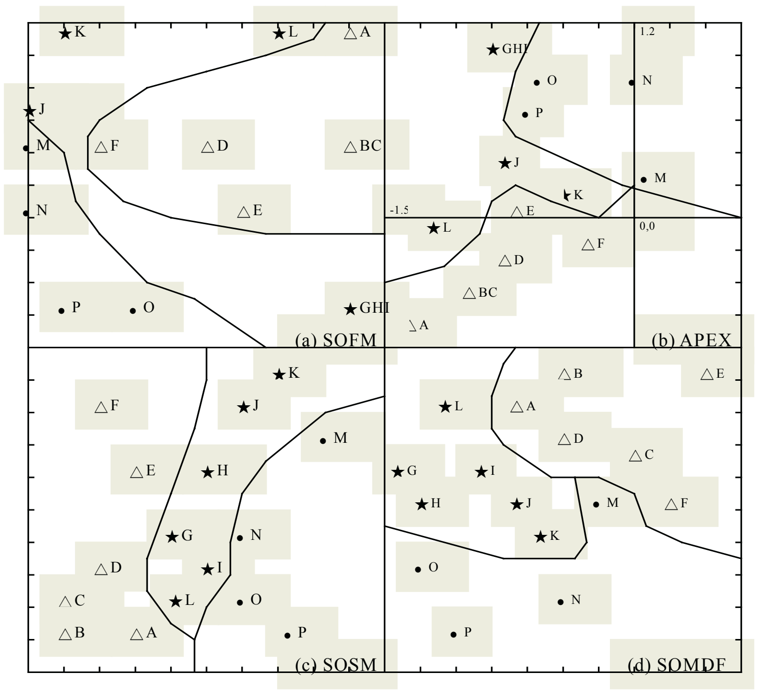


Fig. 2. Several SOMs of Conceptual and Semantic Relations for noun.

4 Conclusion and Discussion

Although the APEX algorithm is a non-linear processing, principal component feature extraction belongs to a linear operation, which only adopts a non-linear neural network method to realize its linear algorithm. The class extended feature of SOSM is a supervisory learning [5] which based on brain learning rules. A supervisory learning of class extended mode can improve systemic performance and increase systemic memory ability. Above experiments show that if the feature description includes its feature content, symbolic class information and its attribute relations, then after feature map the semantic relation can certainly be responded. Even if two objects' features are the same, but the class extended SOSM distinguishes them into different classes, and forces map to different output nodes. However, SOMDF is different from SOSM, the dimensions of the former input features are not increased, the objects can also be fine classified. Neural networks can imitate the organize structure and function mechanism of brain nerve in some extent, by learning to obtain the cognition of some impersonal objects. Because human brain can save plentiful information including speech, image, figure and text etc., and is good at integrating all information for judgment of fuzzy, misshapen and aberrance objects, to reach the aim of different objects correctly recognized. High-level cerebral running depends mainly on the extraction of concept which is expressed by sign and language. During learning process, sign is expressed by its content and similitude degree between signs which is expressed by similitude degree of their content. The feature description used in neural network models is based on digital information, needs to translate all kind of information from real world into a digital style. Human brain can deal with some feature descriptions by fuzzy or simulative mode, for example, big and small, gentle and simple etc. However, neural networks can only deal with this information by using a digital expression. Learning methods of attribute-based description's neural networks have a limited expression of the background knowledge and make the concept description language appropriate to natural language understanding in some degree. At present about research of feature expression is very insufficiency. The other way about research of all kind of algorithms is very plenty, we remedy the feature expressive shortcoming by algorithms' predominance. When there is a lack of the content tag, class information of signs and their attribute relation in feature expression, it is difficult to reach an expectant aim if we only depend on the algorithms' predominance.

References

1. Li, P., Jin, Z., Tan, L. H.: Neural Representations of Nouns and Verbs in Chinese: An FMRI Study, **21** (2004) 1533-1541
2. Kung, S. Y., et al.: Adaptive Principal Component Extraction (APEX) and Application. IEEE Transactions on Signal Processing, **42** (1994) 1202-1217
3. Kohonen T.: The Self-Organized Map. Proceedings of the IEEE, **78** (1990) 1464-1480
4. Dong, Z., Dong, Q. Hownet. Kim, P.: Theories of Early Language Acquisition. Trends in Cognitive Sciences, **1** (1997) 146-1535
5. Bezdek, J. C., Pal, N. R.: A Note on Self-Organizing Semantic Maps. IEEE Transactions on Neural Networks, **6** (1995) 1029-1036

Artificial Neural Network for Prediction of Rockburst in Deep-Buried Long Tunnel*

Xiaohong Li, Xinfei Wang, Yong Kang, and Zheng He

Key Lab of Ministry of Education for the Exploitation of Southwest Resources & the Environmental Disaster Control Engineering, College of Resources & Environmental Sciences, Chongqing University, Chongqing 400044, China
xhli@cqu.edu.cn, {wangxinfei781231,icorn}@163.com, hzcq@vip.sina.com.cn

Abstract. Rockburst is a main engineering geological problems in deep-buried long tunnels. Rockburst phenomena have been analyzed, assumptions and criteria have been presented from the perspectives of strength, stiffness, energy, steadiness, fracture, damage, fractal and catastrophe, etc. Considering individual factors only among some assumptions and criteria will cause unilateral and limited results. Based on the assumptions and criteria of rockburst and real examples of tunnel engineering, a neural network model is proposed to predict rockburst. The prediction results show that it is feasible and valid to use artificial neural network for predicting rockburst.

1 Introduction

Rockburst is a dynamic destructive phenomenon in the process of deep-buried long tunnel construction. Rockburst can lead to surrounding rock destructions in great area abruptly. Surrounding rocks will release energy, which may lead to the destruction of the tunnel[1].At present, experts and scholars have analyzed rockburst phenomenon and raised assumptions and criteria from the angles of strength, stiffness, energy, steadiness, fracture, damage, fractal and catastrophe, etc. Considering individual factors only among some assumptions and criteria will cause unilateral and limited results. While considering all the factors will make the problem more complex [2]. In addition, the relationship between the concerned factors in rock engineering, which has only relative accuracy, and rockburst cannot be assessed simply by true or false. Non-linear ANN just suits to such problems because a clear function relationship between them is not required. Once the main factors are determined, the rest is to train and predict correctly.

2 Application of Artificial Neural Network

2.1 Neural Network Model for Prediction of Rockburst in Deep-Buried Long Tunnel

A typical three-layered BP network is selected here. The structure of a typical three-layered BP network is shown in Fig1. It uses forward three-layered back propagation

* Funded by National Natural Science Foundation Project (No.50334060).

learning algorithm. It is consisted of input layer, hidden layer and output layer. The simulating function of units is S-type function. This type of network is of great nonlinear mapping ability. And it is a steady pattern recognition method[3].

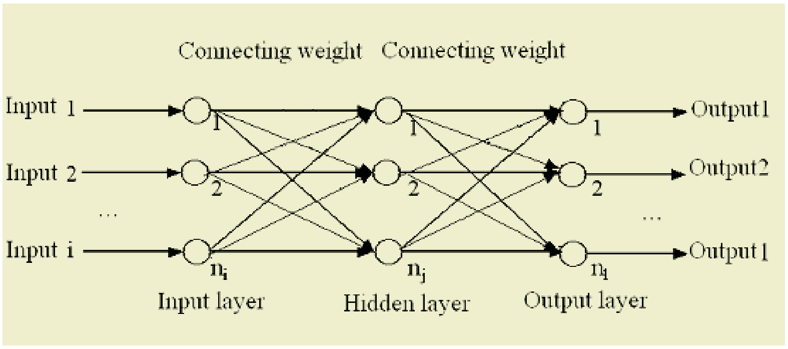


Fig. 1. Structure of three-layered BP network.

On the basis of previous researching criteria of rockburst, it choose maximum tangential stress σ_θ, uniaxial compressive strength σ_c, uniaxial tensile strength σ_t, ratio of maximum tangential stress to uniaxial compressive strength σ_θ/σ_c, ratio of uniaxial compressive strength to uniaxial tensile strength σ_c/σ_t and *Wet* as the input features[1]-[5]. The maximum tangential stress σ_θ reflects the strata stress character of rockburst. The rock, which occurs rockburst, is compact. The main characteristics of rock which impact rockburst are uniaxial compressive strength σ_c and uniaxial tensile strength σ_t. *Wet* reflects rock's ability storing elastic energy. Research Russenes, Turchaninov and Hoek's experiential criteria synthetically, σ_θ/σ_c is selected as an input feature. The criteria of rockburst presented by Lu J. indicate that when σ_θ/σ_c is bigger than K_s rockburst will occur. And the value of K_s depends on the ratio of uniaxial compressive strength to uniaxial tensile strength σ_c/σ_t. So the grade of rockburst depends on the ratio of uniaxial compressive strength to uniaxial tensile strength σ_c/σ_t. When the ratio of uniaxial compressive strength to uniaxial tensile strength σ_c/σ_t is small, rockburst occurs acutely. When the ratio of uniaxial compressive strength to uniaxial tensile strength σ_c/σ_t is big, rockburst occurs lightly. So σ_θ/σ_c is selected as an input feature.

If the grade of rockburst is divided to 4 degrees, the number of output layer's units is set 4. And (1, 0, 0, 0) is defined as intense degree, (0,1,0,0) is defined as middle degree, (0, 0,1,0) is defined as light degree, and (0, 0, 0, 1) is defined as no rockburst. The number of hidden layer's unit is 15.

2.2 Training of Network

The training sample patterns are 20 typical rockburst examples home and abroad, the data is in table 1. The learning rate η is set as 0.85 and momentum factor α *is* set as 0.68 , and maximum allowed system error is set as 1×10^{-5}, and maximum allowed pattern error is set as 1×10^{-6}.Training finish when number of iteration is 55406, and the system error is less than 1×10^{-5}.

Table 1. Rockburst Examples Home and Abroad.

Engineering	σ_0 /MPa	σ_c /MPa	σ_t /MPa	Wet	σ_0/σ_c	σ_c/σ_t	Desired output	Training Output			
Diversion Tunnels of Jingping Hydropower Station	98.6	120	6.5	3.8	0.822	18.46	0,1,0,0	0.002626	0.997599	0.000000	0.000000
Underground power-house of Laxiwa Hydropower Station	55.4	176	7.3	9.3	0.315	24.11	0,1,0,0	0.000021	0.996643	0.002443	0.001757
Qinling Tunnel of Xikang Railway Dyk77+176	56.1	132	9.4	7.44	0.425	13.98	0,1,0,0	0.002702	0.996443	0.002076	0.000000
Qinling Tunnel of Xikang Railway-Dyk72+440	60.7	111	7.9	6.16	0.544	14.19	1,0,0,0	0.997802	0.002481	0.000000	0.000000
Diversion Tunnels of Swedish Vietas Hydro-power Station	80.0	180	6.7	5.5	0.444	26.87	0,0,1,0	0.000010	0.003174	0.997111	0.000066
Soviet Rasvumchorr Workings	57.0	180	8.3	5.0	0.317	21.69	0,1,0,0	0.000001	1.000000	0.000000	0.000026
Japanese Guanyuk Tunnel	89.0	236	8.3	5.0	0.377	28.43	0,1,0,0	0.000000	0.998670	0.001535	0.000082
Italian Raibl Lead Zinc Sulfide Working	108.4	140	8.0	5.5	0.774	17.50	1,0,0,0	0.996853	0.002757	0.000000	0.000000
Diversion Tunnels of Tianshengqiao-II Hydropower Station	30.0	88.7	3.7	6.6	0.338	23.97	0,1,0,0	0.000003	0.998715	0.000759	0.001964
Sub Tunnel of Ertan Hydropower Station	90.0	220	7.4	7.3	0.409	29.73	0,0,1,0	0.000068	0.000333	0.998720	0.000169
Underground Cavern of Longyangxia Hydro-power Station	18.8	178	5.7	7.4	0.106	31.23	0,0,0,1	0.000000	0.000115	0.000000	0.999884
Underground Cavern of Lubuge Hydropower Station	34.0	150	5.4	7.8	0.227	27.78	0,0,0,1	0.000001	0.001971	0.000000	0.997651
Underground Power-house of Norwegian Sima Hydropower Station	48.8	180	8.3	5.0	0.271	21.69	0,1,0,0	0.000000	1.000000	0.000000	0.000815
Norwegian Heggura Road Tunnel	62.5	175	7.3	5.0	0.357	24.14	0,1,0,0	0.000001	0.999942	0.000471	0.000115
Norwegian Sewage Road Tunnel	75.0	180	8.3	5.0	0.417	21.69	0,1,0,0	0.000095	0.999866	0.000080	0.000000
Cooling Diversion Tunnels of Swedish Forsmark Nuclear-Power tation	50.0	130	6.0	5.0	0.385	21.67	0,1,0,0	0.000033	0.998217	0.001183	0.000033
Diversion Tunnels of Yuzixi Hydropower Station	90.0	170	11.3	9.0	0.529	15.04	0,1,0,0	0.001573	0.998155	0.001852	0.000000
Underground Cavern of Taipingyi Hydropower Station	62.6	165	9.4	9.0	0.379	17.55	0,0,1,0	0.000993	0.003436	0.996551	0.000010
Underground Cavern of Lijiaxia Hydropower Station	11.0	115	5.0	5.7	0.096	23.00	0,0,0,1	0.000000	0.001685	0.000000	0.999255
Underground Cavern of Pubugou Hydropower Station	43.4	123	6.0	5.0	0.353	20.50	0,1,0,0	0.000021	0.999927	0.000154	0.000044

3 Predicting Rockburst of Tongyu Tunnel by BP Neural Network

The Surrounding rocks of K21+680 900 meters deep in Tongyu Tunnel are limestone of P_2W, its mechanic parameters [6] and predicting result is in table 2.According the results, rockburst will occur lightly there. When working face arrived, the surface of surrounding rocks will break lightly, and became more and more obvious. So the predicting results are feasible and credible.

Table 2. Prediction of Rockburst at K21+680 in Tongyu Tunnel 900 meters deep.

Engineering	σ_θ /MPa	σ_c /MPa	σ_t /MPa	Wet	σ_θ/σ_c	σ_c/σ_t	Desired output	Training Output			
Tongyu Tunnel K21+680	47.56	58.5	3.5	5	0.338	23.97	0,1,0,0	0.000003	0.998715	0.000759	0.001964

4 Conclusions

Rockburst is caused by many factors. Choosing the key factors correctly, the right criteria of predicting rockburst can be obtained. On the basis of analysis of rock-burst's cause of formation and calculating of examples, six indexes are chosen including maximum tangential stress σ_θ, uniaxial compressive strength σ_c, uniaxial tensile strength σ_t, ratio of maximum tangential stress to uniaxial compressive strength σ_θ/σ_c, ratio of uniaxial compressive strength to uniaxial tensile strength σ_c/σ_t and *Wet*. Results indicate that to predict rockburst with ANN is feasible.

References

1. Bai, M. et al.: Study on a Neural Network Model and its Application Predicting the Risk Rock Blast. China Safety Science Journal, **12** (2002) 65-69
2. Wang, Y. et al.: Method of Fuzzy Comprehensive Evaluations for Rockburst Prediction. Chinese Journal of Rock Mechanics and Engineering, **17** (1998) 493-501
3. Wang, X.: Research on Pre-Splitting Blasting Design Expert System and its Application in Cutting Excavation of Expressway. Chongqing University, Chongqing (2003)
4. Chen, H., et al.: A Model for Prediction of Rockburst by Artificial Neural Network. Chinese Journal of Geotechnical Engineering, **24** (2002) 229-232
5. Ding, X., et al.:Artificial Neural Network for Forecasting and Classification of Rockbursts.Journal of Hohai University (Natural Sciences), **31** (2003) 424-427
6. Wang, Q., Li, X., et al.: Numerical Simulating of Surrounding Rocks' Deformation and Rockburst in Tongyu Tunnel. Underground Space, **23** (2003) 291-295

Implementation of Brillouin-Active Fiber Based Neural Network in Smart Structures

Yongkab Kim[1], Sunja Lim[1], Hwan Y. Kim[1], Sungkwun Oh[2], and Chung Yu[3]

[1] Departments of Electrical Engineering, Wonkwang University, 570-749, Iksan South Korea
ykim@wonkwang.ac.kr
http://www.forlab.wonkwang.ac.kr
[2] Department of Electrical Engineering, The University of Suwon,
Gyeonggi-do 445-743, South Korea
ohsk@Springer.de
[3] Department of Electrical Engineering, North Carolina State University, Greensboro, USA
chungyu@ncat.edu

Abstract. We propose a novel application of SBS in embedded optical fibers as a building block for the implementation of optical neural net in smart structures. By employing well established technology in fiber embedding in structures, and exploiting the ability of SBS as a highly versatile ambient sensor, and its nonlinear optical property of energy addition and subtraction to perform optical arithmetic, we predict the possible integration of sensing and actuation for smart structures. In this paper, the fiber based SBS neuron is explained. Optical arithmetic schemes are shown to demonstrate the simplicity of operation of these neurons. Typical implementation schemes are illustrated.

1 Introduction

It is well known that optical fibers have potential for various [1] uses other than in the communications field, such as recent research on using optical fibers as versatile sensors for various measured. Our research has also focused on integrating fiber optic sensors with actuation materials to create a system that is capable of sensing, and controlling shape or orientation with respect to its environment, as a first step in creating a smart sensor structure. Specifically, we have focused on configuring and developing a Stimulated Brillouin Scattering (SBS) sensing system that behaves as a neural network, in order to acquire the ability to learn by experience, predict future reactions to environmental changes, and execute as prescribed.

Such a smart sensor system would implement a massively parallel computational architecture with its attendant reduction in processing time while managing the complexity of the system, i.e. the sensing/actuation grid. Our SBS network would learn the correct "algorithms" by example during training and have the ability to generalize to untrained inputs after training is completed. The inputs to the network are the fiber optic sensor signal outputs, and the network outputs are the control signals for actuation controls. The true advantage of this system for application to smart sensor structures lies both in its capability to analyze complex sensor signal patterns and its speed in generating the appropriate control signal for the actuators. The key lies in the implementation of a neuron operation using SBS in optical fibers.

2 SBS Based Neuron

Nonlinear effects in optical fibers, specifically stimulated Brillouin scattering(SBS), has emerged as a versatile approach to the construction of active optical devices for all-optic in-line switching, channel selection, amplifiers and oscillators in optical sensing, and optical communications[2], [3], [4]. The backward scattering nature of Brillouin scattering, which is the light reflection by laser induced acoustic wave in the fiber, has long been viewed as an ultimate intrinsic loss mechanism in long haul fibers, since Brillouin threshold decreases with increasing effective fiber length[5], [6]. The very backscattering nature of this nonlinear process and the existence of a threshold provide potential optical device functions, such as optical switching, arithmetic and neural functions in networks. An artificial neuron, used in neural network research, can be thought of as a device with multiple inputs and a single or multiple outputs. The inputs to a neuron are weighted signals. The neuron adds the weighted signals, compares the result with a preset value, and activates if the sum exceeds threshold. In the nonlinear optical phenomenon, the system combined weighted signals also produces an output if the weighted sum is greater than the threshold. A typical neuron is illustrated in Fig. 1.

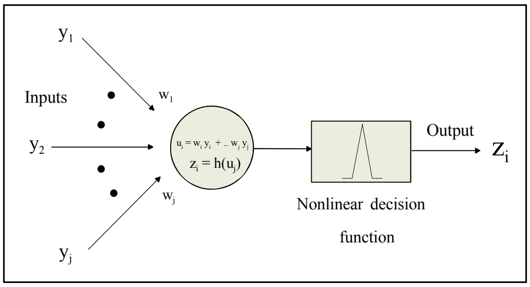


Fig. 1. A simplified multi-layered feedward neural network; the processing node between interconnects, where weighted sums are subjected to threshold decision-processing element.

In the nonlinear optical phenomenon of SBS, the system through mixing combines weighted signals to produce an output if the weighted sum exceeds the threshold. The threshold decision is made by an individual neuron in conjunction with weighted inputs from other neurons. A theoretical SBS based neural network, utilizing SBS threshold sensing with an embedded sensor is seen in Fig.2.

The arithmetic building block of energy addition and subtraction, as in Fig.2, can conceivably be accomplished by the SBS process, which involves energy transfer between waves. Thus, if two waves at a frequency difference equal to the stokes shift of the fiber propagate in the fiber in opposite directions, then energy is "subtracted" from the higher frequency wave and "added" to the lower frequency wave. If three waves are present in a fiber with equal stokes shifts, then the wave at the middle frequency will receive energy from the higher frequency wave and lose energy to the lower frequency wave. Practical implementation of this scheme calls for all the waves to be generated by the same laser.

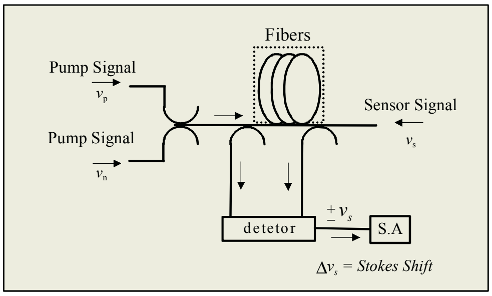


Fig. 2. SBS implementation of threshold logic.

3 SBS Based Threshold Logic Implementation

A practical implementation of theoretical neuron calls for all the waves to be generated by the same laser. We are very familiar with this method and a scheme is devised in Fig.3 [7]. A laser was used as the pump to the system through a coupler. An isolator is installed to prevent any reflected signal back into the laser cavity that may disrupt the performance of the laser. The pump wave travels through the long fiber to an embedded sensing fiber. If the pump signal launched into the fibers exceeds some critical threshold level, then SBS occurs. In this process, the input pump traveling through the fibers may be converted into a Stokes wave, shifted in frequency, and traveling backward towards the laser source.

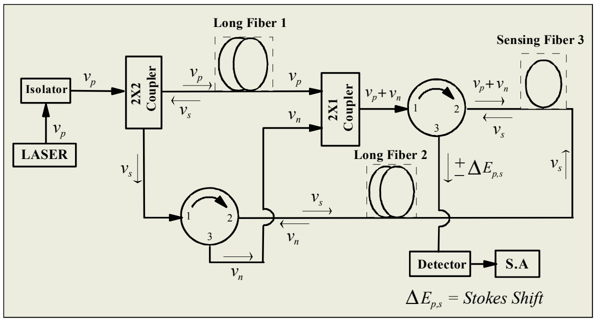


Fig. 3. Practical Implementation SBS based threshold logic. Optical fibers are used as the medium for providing SBS gain to the stokes wave. Here $\nu_p > \nu_s > \nu_n$.

Three identical fibers are used, two in the oscillator format, and one in the amplifier format. Fiber 1 is used to provide the v_s generated by the laser pump ν_p. The v_s wave is then used as a pump to generate ν_n in fiber 2. The residual v_s and ν_p proceed to fiber 3, where all three waves will mix. Since all three signals are present, the third fiber can be in the amplifier format. The sensor signal v_s wave will act as a stokes wave for ν_p and as a pump wave for ν_n, with ν_p - ν_s =$\Delta\nu_p$, and ν_s - ν_n =$\Delta\nu_p$, where $\Delta\nu_p$ is the Brillouin shift. The enhanced Stokes signal, and pump signal of the SBS oscillator fiber and other backwards signals due to the sensing fiber will be used for the multipoint threshold sensing technique in the SBS neural net. The 'sensed' signal will be viewed on a Spectrum Analyzer for comparison. This sensed signal energy can be added to and/or subtracted from the sensor wave by its interaction with v_p and v_n via SBS. In our implementation, the threshold of the device can be set to any suitable output power level of the "sensor" signal. We take the threshold to be reached when the sensor signal emerges from the fiber device with no net gain or loss (0dB gain). The SBS mechanism can also serve as input weight controller, since energy can be added or subtracted from the input signal. The SBS oscillator / amplification scheme without wave mixing is shown in Fig. 4, where the Brillouin-shifted energies are $\Delta E_p = h\Delta\nu_p$, and $\Delta E_s = h\Delta\nu_s$. This energy can be added or subtracted from the sensor wave ν_s when it interacts with waves at ν_p and ν_n via SBS.

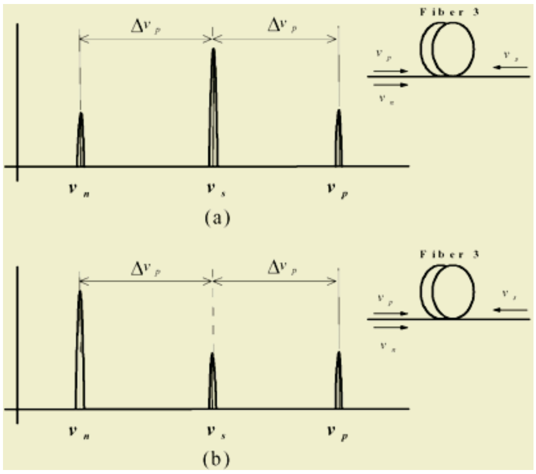


Fig. 4. (a); Addition = Energy from ν_p added to ν_s and (b); Subtraction = Energy from ν_s added to ν_n.

As an example in temperature sensing, the sensor system be at a temperature T_1, signal energy is subtracted from ν_p by ν_s and then subsequently added to ν_n. If the temperature is changed to a new value T_2, different from T_1, then the ν_p and ν_s waves will create new stokes signals shifted to frequencies ν_s' and ν_n'. Due to the presence of these new frequencies, ν_p and ν_s, and ν_s and ν_n , respectively, the arithmetic changes and no signal energy is added and/or subtracted as shown in Fig. 5. The experimental result of a practical implementation for a SBS based neural networks shows two narrow single-frequency outputs. The Brillouin shifted output, which 'adds' signal energy (large-amplitude signal) and/or 'subtracts' signal energy (small signal), from the sensor wave through interaction (Δv_s) of v_p and v_n via SBS observed on the spectrum analyzer. For the case where the signals add, the Stokes signal is downshifted in frequency by approximately 12.9GHz to 12.8GHz, with the display centered at 12.85 GHz. Linewidth resolutions was better than 30 MHz.

The narrow gain spectrum and relatively small frequency shift of the SBS process will allow the use of the same oscillator format (identical fibers) for signals. In the case of subtraction, the Stokes signal is up-shifted in frequency by about 30 MHz from the sensing signal in an amplifier format.

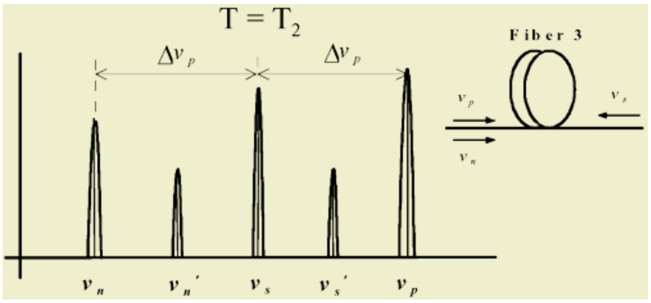


Fig. 5. SBS Oscillator-Amplifier Amplification scheme, with the presence of ν_s' and ν_n' between ν_p, ν_s, and ν_n, at fiber 3 when temperature is changed from T_1 to T_2.

4 Conclusions

Embedded fiber sensors have been extensively deployed in structures normally as passive light conduits. However, Brillouin active fibers not only propagate light, but also provide device functions in the fiber. The ability of SBS to perform both sensing and optical arithmetic render such a scheme as the simplest building block for neural network based smart structures.

Acknowledgement

This paper was supported by Research Institute of Engineering Technology Development, WonKwang University.

References

1. Grossman, B., Alavie, T., Ham, F., Franke, F., Thursby, M.: Fiber-optic Sensor and Smart Structures Research at Florida Institute of Technology. SPIE., **1170** (1989) 213-218
2. Koyamada, Y., Sato, S., Nakamura, S., Sotobayashi, H., Chujo. W.: Simulating and Designing Brillouin Gain Spectrum in Single-Mode Fibers. J. of Lightwave Tech., **22** (2004) 631-639
3. Bernini, R., Minardo, A., Zeni. L.: Stimulated Brillouin Scattering Frequency-Domain Analysis in a Single-Mode Optical Fiber for Distributed Sensing. Optics Letters., **29** (2004) 1977-1979
4. Tanemura, T., Takyshima, Y., Kikuchi. K.: Narrowband Optical Filter, with a Variable Transmission Spectrum, Using Stimulated Brillouin Scattering in Optical Fiber. Opt. Lett., **27** (2002) 1552-1554
5. Cotter, D.: Stimulated Brillouin Scattering in Monomode Optical Fiber. J. Opt. Com., **4** (1983) 10-19
6. Agrawal**,** G, P.**:** Nonlinear Fiber Optics, 3rd, Academic press, London (2001)
7. Tariq, S., Habib, M, K.: Neural Operation Using Stimulated Brillouin Scattering in Optical Fiber. Opt. Eng., **37** (1998) 1823-1826
8. Yong, K, Kim., Choon, B, Park.: Study Of Chaotic and Instability Effect of Optical Fiber Using on the Internet. SPIE., **5246** (2003) 648-655

Inelastic Simulation of Insect Cuticle Using Artificial Neural Network

Bin Chen[1], Gang Chen[2], Hongtao Liu[1], Xianghe Peng[1], and Jinghong Fan[1]

[1] College of Resources and Environmental Science, Chongqing University, Chongqing 400044, China
bchen@cqu.edu.cn
[2] College of Automation, Chongqing University, Chongqing 400044, China

Abstract. Neural networks have been availably applied to the simulations of the mechanical behaviors of many materials. In this work, a neural network material model is built for the simulation of the inelastic behavior of biocomposite insect cuticle. Radial basis function neural network is adopted in the simulation for that the neural network has the characteristic of fast and exactly completing the simulation. In the construction of the neural network, the network is trained based on the experimental data of the load-displacement relationship of a chafer cuticle. A strain-controlled mode and the iterative method of data are adopted in the training process of the neural network. The obtained neural network model is used for the simulation of the inelastic behavior of another kind of insect cuticle. It is shown that the obtained material model of the radial basis function neural network can satisfactorily simulate the inelastic behavior of insect cuticle.

1 Introduction

The computer simulations of material behaviors have distinct advantages than the experimental tests in many cases due to their more economical, timesaving and convenient. In addition, the computer simulations of material behaviors can also get more information than experimental tests. Nevertheless, up to today, most of the computer simulations of material behaviors are limited in the analysis of elastic solid because there are considerable model errors in present inelastic or elastoplastic material models. An alternative is to use a neural network to model material behavior. The main benefit in using the approach of a neural network is that all material behaviors can be represented within a unified environment of a neural network and that the network is built directly from experimental data using the self-organizing capabilities of the neural network, i. e., the network is presented based on the experimental data and the "learning" of the real relationships between the stresses and strains of the materials. Such a modeling strategy has important implications for modeling the inelastic or elastplastic behaviors of modem, complex materials, such as various composites [1].

The simulations and descriptions of material behaviors with the models of neural networks have been reported by some researchers [1]-[5]. Ghaboussi et al. [1] modeled the behaviors of concrete in the state of plane stress under monotonic biaxial loading and compressive uniaxial cycle loading with a back-propagation neural network. Abendroth and Kuna [2] described an approach to identify plastic and failure properties of ductile materials by means of a small punch test and a feed forward neural network. Furukawa and Hoffman [3] presented a material model using a multi-layer perceptron neural network that has the ability to describe plasticity and cyclic plasticity of a material. Genel et al [4] built a multiple-layer-feed-forward artificial neural network model for modeling the tribological behaviors of short alumina fiber reinforced zinc-aluminum composite. Liu Q et al [5] acquired the constitutive relationship of a thermal viscoplastic material using a back-propagation neural network. In this paper, a radial basis function neural network is constructed and trained for describing the inelastic behavior of biocomposite insect cuticle. It is shown that the obtained material model of the neural network with radial basis function can well describe the inelastic behavior of insect cuticle.

2 Inelastic Behavior of Biocomposite Insect Cuticle

Through untold centuries of evolutionary development, insect cuticle has become a kind of typical biocomposite that has highly optimized fiber-reinforced microstructure and excellent strength, stiffness and fracture toughness [6]. The investigation on the material behavior of biocomposite insect cuticle can give a profitable guidance for the research of man-made advanced biomimetic composites. In this section, the relationship between the load and the displacement of an insect cuticle is investigated with a small tensile experiment. The obtained test data will be transferred to a radial basis function neural network to obtain the material model of the neural network of insect cuticle.

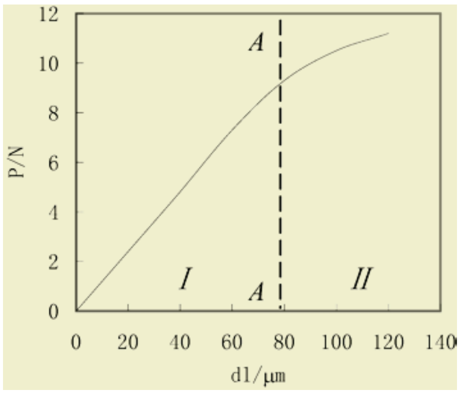


Fig. 1. Load-displacement curve of elytra.

The insect cuticle selected for the experiment is the elytra of chafer because the cuticle has higher strength, stiffness, fracture toughness and larger size. Firstly, the cuticle was peeled from the insect, and then the cuticle was fabricated to small tensile specimens. At last, the specimens were tested with a small material testing system.

The measurable input is the load applied on the specimen and the measurable output is the displacement of the cuticle. The relational curve of the load-displacement provides the main information of the material behavior and can be easy transferred into stress-strain curve. Figure 1 shows the curve of the load-displacement of the cuticle specimen. The load-displacement curve can be divided into two parts: one is the elastic section (Part I), another is the inelastic section (Part II). *A*-*A* is the dividing line between the elastic and plastic section of the material. From Fig. 1 it is clear that the cuticle possesses inelastic properties, which will be simulated with a radial basis function neural network model in following section.

3 Simulation of Material Behavior with Neural Network

3.1 Construction of Neural Network Model

Multi-layered feed-forward neural networks are most suitable for the research of material behaviors [7]. In multi-layered feed-forward neural networks, the artificial neurons (also referred to as processing units or nodes) are arranged in layers and all the neurons in each layer connect to all the neurons in the next layer. Signals received at the input layer pass through the hidden layers and reach the output layer, producing the output of the neural network. Specially, the neural network used here is a radial basis function neural network (RBFNN) that is belongs to the multi-layered feed-forward neural networks. Though more neurons are needed in the RBFNN, but the RBFNN has well ability in approaching targets and higher "learn" speed than general multi-layered feed-forward neural networks [8]. There are two kinds of model forms in RBFNN, one is its regularized network and another is its extended network. In the research, the regularized network is adopted.

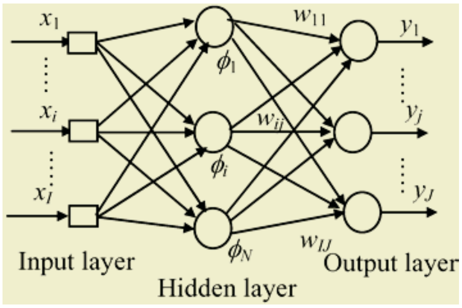


Fig. 2. Radial basis function neural network.

The RBFNN consists of three layers: input layer, hidden layers and output layer (Fig. 2). The input layer has *I* neurons and anyone of the neurons can be expressed as *i*. The hidden layer includes *N* neurons and anyone of the neurons can be expressed with *n*. The output layer has *J* neurons and anyone of the neurons can be expressed as *j*. The "basis function" of the network is $\phi(X, X_i)$ which is the inspiriting output of *i*-th hidden unit. The connection weight between the hidden layer and the output layer is w_{ij} $(i = 1,2,...I;\ j = 1,2,...,J)$. When a straining sample X_k of the network is inputted, the output of *j*-th outputting neuron is

$$y_{kj}(X_k) = \sum_{i=1}^{N} w_{ij}\phi(X_k, X_i),\ \ (j = 1,2...,J)\ . \tag{1}$$

Green function is adopted as the basis function of the network

$$\phi(X_k, X_i) = G(X_k, X_i)\,. \tag{2}$$

specially it is Gaussian function

$$\varphi(r) = \exp\left(-\frac{(r-t)^2}{2\sigma^2}\right), \qquad (\sigma > 0, \quad r \in R)\,. \tag{3}$$

where t is the center of the Gaussian function, σ is its square error, then the $G(X_k, X_i)$ can be expressed as

$$G(X_k, X_i) = G(\|X_k - X_i\|) = \exp\left(-\frac{1}{2\sigma_i^2}\sum_{m=1}^{M}(x_{km} - x_{im})^2\right). \tag{4}$$

where $X_i = [x_{i1}, x_{i2}, \ldots, x_{iM}]$ is the center of the Gaussian function and σ_i is the square error of the Gaussian function.

There are three parameters needed to be "learned" in the RBFNN: the center $t_i\ (i = 1,2,\ldots I)$ of the basis function, the square error $\sigma_i\ (i = 1,2,\ldots I)$ and the weight value $w_{ij}\ (i = 1,2,\ldots I,\ j = 1,2,\ldots J)$. The center of the basis function can be obtained through following equation

$$i(X_k) = \arg\min\|X_k - t_i(n)\|\,, \quad (i = 1,2,\ldots,I)\,. \tag{5}$$

where n is the iterative times. The square error can be calculated as follows

$$\sigma_1 = \sigma_2 = \ldots = \sigma_I = \frac{d_{\max}}{\sqrt{2I}}\,. \tag{6}$$

where I is the of hidden-element number and $d_{\max}$ is the maximal distance between the selected centers.

3.2 Training and Test of Neural Network Model

Constructed RBFNN was trained using obtained experimental curve of stress-strain relationship of the specimen of the chafer cuticle. A strain-controlled training mode was adopted [1]. In the training process of the network, the strain increments of the specimen of the insect cuticle were presented to the network as input and the stress increments of the specimen as output. The network would be trained to predict stress increments given the current state of stress and strain and a strain increment. This is done by starting at a known stress-strain state, incrementing small strains, and using the RBFNN to predict the stress increments of the specimen. These stress increments can then be added to get the new state of stress, which can be used to predict the stress increment for another strain increment of the cuticle specimen. The predicted result of the curve of the load-displacement of the chafer cuticle is shown in Fig. 3. From Fig. 3 it is clear that the predicted result is very closely to the experiment result and exactly describes the inelastic behavior of the cuticle.

The trained material model of RBFNN was also used to predict the inelastic behaviors of another kind of insect cuticle, the Rutelidae elytra. The experimental curve of the load-displacement of the cuticle was obtained with the same method as that of the chafer cuticle. The experimental result of the Rutelidae elytra was predicted using the trained RBFNN. When the new inputs are presented to the network, the outputs will be predictable. Figure 4 shows the comparison between the predicted values acquired from the neural network and the experimental results of the Rutelidae cuticles. From Fig. 4 it can be seen that the experimental and predicted values of the insect cuticle are very close to each other and the RBFNN can satisfactorily predict the inelastic behavior of Rutelidae elytra with satisfactory precision.

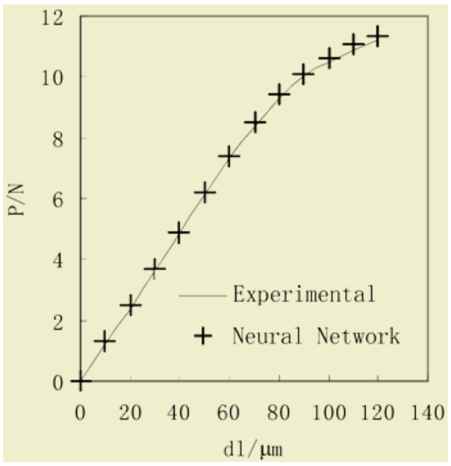


Fig. 3. Load-displacement curve of Chafer elytra.

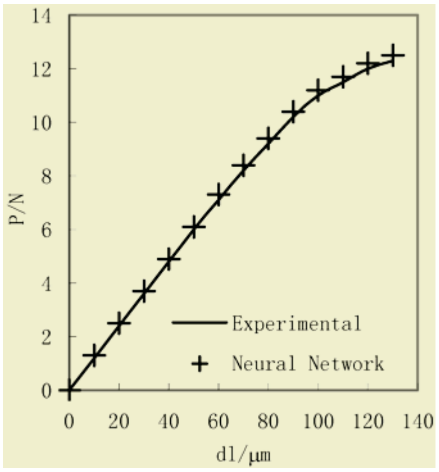


Fig. 4. Load-displacement curve of Rutelidae elytra.

4 Conclusions

As a kind of biocomposite, insect cuticles have inelastic material property. In the paper, the inelastic material property of chafer cuticles was simulated using the technique of artificial neural network. A radial basis function neural network was adopted in the research for the network has the characteristic of fast and exactly completing the simulation. In the construction of the neural network, the network was trained based on the experimental data of the load-displacement relationship of the chafer cuticles. A strain-controlled mode and a data-iterative method were adopted in the training process. The obtained neural network model was used for the simulation of the inelastic behavior of insect cuticles. It was shown that the obtained material model of the radial basis function neural network can satisfactorily simulate the inelastic behavior of insect cuticles.

Acknowledgements

The authors gratefully acknowledge the financial support to this work from the Natural Science Foundation of China (Grant no. 10272120) and Chongqing (Grant no. 8417).

References

1. Ghaboussi, J., Garrett, J. H., Wu, X.: Knowledge-based Modeling of Material Behavior with Neural Networks. J. Engineering Mechanics, **117** (1991) 132-153
2. Abendroth, M., Kuna, M.: Determination of Deformation and Failure Properties of Ductile Materials by Means of the Small Punch Test and Neural Networks. Computational Materials Science, **28** (2003) 633-644
3. Furukawa, T., Hoffman, M.: Accurate Cyclic Plastic Analysis Using a Neural Network Material Model. Engineering Analysis with Boundary Elements, **28** (2004) 195-204
4. Genel, K., Kurnaz, S., Durman, M.: Modeling of Tribological Properties of Alumina Fiber Reinforced Zinc-aluminum Composites Using Artificial Neural Network. Materials Science & Engineering A, A363 (2003) 203-210
5. Liu, Q., Ji, Z., Liu, M., Wu, S.: Acquiring the Constitutive Relationship for a Thermal Viscoplastic Material Using an Artificial Neural Network. Material Processing Technology, **62** (1996) 206-210
6. Chen, B., Peng, X., Wang J., Fan, J., Wu, X.: Investigation of Fiber Configurations of Chafer Cuticle by SEM, Mechanical Modeling and Test of Pull-out Forces. Computational Materials Science, **30** (2004) 511-516
7. Ghaboussi, J., Sidarta, D.: New Nested Adaptive Neural networks (NANN) for Constitutive Modeling, **22** (1998) 29-52
8. Efe, M., Kaynak, O.: A Comparative Study of Neural Network Structures in Identification of Nonlinear Systems. Mechatronics, **9** (1999) 287-300

Applying Neural Networks and Geographical Information Systems to Airport Noise Evaluation

Yingjie Yang[1], David Gillingwater[2], and Chris Hinde[3]

[1] Centre for Computational Intelligence, De Montfort University,
The Gateway, Leicester, LE1 9BH, UK
yyang@dmu.ac.uk

[2] Transport Studies Group, Department of Civil and Building Engineering,
Loughborough University, Loughborough, LE11 3TU, UK
D.Gillingwater@lboro.ac.uk

[3] Department of Computer Science, Loughborough University,
Loughborough, LE11 3TU, UK
C.J.Hinde@lboro.ac.uk

Abstract. The assessment of aircraft noise is becoming an increasingly important task in ensuring sustainable airport development. Aircraft noise is influenced by many complex factors and traditional laboratory models are not sufficient to assess the exposure to noisy flights of specific local communities in proximity to an airport. In this paper neural network and fuzzy set methods have been integrated with Geographical Information Systems to provide an alternative method to evaluate airport noise.

1 Introduction

In airport operations, identifying and monitoring noise disturbance caused by a specific aircraft movement at a specific airport on a specific local community is very important when considering financial penalties, compensation claims and social costs [1]. The standard methodologies available follow one of two classes: (i) the 'laboratory model' - based on laboratory-type experiments and standardized in-situ tests undertaken in given conditions; or (ii) the 'replication/simulation model' - based primarily on in-situ test data. However, in practice it is not feasible to monitor each impacted locality around an airport and the interactions between the key factors are too complicated to enable reliable mathematical models to be developed. As a result, laboratory models are in fact the dominant models in use. For instance, aircraft noise calculations around airports are dominated by calibrated models based on standard condition tests and aircraft engine manufacturers' data, such as the US FAA integrated noise model, the INM [2]. These standard models are very useful in simulation analysis at a general level, but they do not have the capacity to incorporate specific local conditions; thus the reliability of their results are subject to standardised assumptions regarding, for example, aircraft engine power settings, geographical factors and weather conditions. From this point of view, a location-specific model established with data from that site is likely to yield more realistic results although its generalizability is likely to be poor when compared with the laboratory model.

As part of our work on devising a decision support system for planning sustainable airport development [3], [4], we applied neural networks [5] and fuzzy sets [6] as a mapping tool to establish the impact of aircraft noise on localities around an airport in a Geographical Information System [7] environment.

2 System Framework

A neural network excels at learning from data and does not require the prior specification of a mathematical model. This feature makes it an ideal candidate in environmental analysis where a large amount of monitoring data exists but where the interactive mechanisms are too complicated or little understood to specify an accurate mathematical model. As a spatial analysis tool, GIS is ideal in dealing with environmental simulation and analysis. Hence, it is a logical step to integrate both into one system. Neural networks have mainly been applied to spatial interpolation [8], spatial attribute mapping [9], [11], [12] and error simulation [13]. In airport operations, however, a clearly defined result or numerical value is often required as an understandable output, and a map generated from a GIS may provide only distributional information rather than a clear conclusion such as the number of disturbed people or affected properties. To overcome this problem, we propose to adopt fuzzy sets incorporated with neural networks. The general structure of this integration is shown in Figure 1.

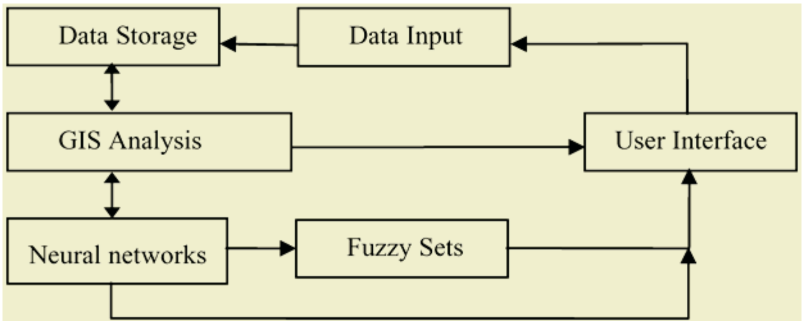


Fig. 1. A framework for linking neural networks and fuzzy sets with GIS.

Here neural networks and fuzzy sets provide a powerful function to GIS, but it is embedded within the functionality of a GIS system - there is no perceived difference to the users of GIS from the point of view of operation as long as a trained neural network is already installed. The system only calls a neural network and its associated fuzzy logic operation when no other method can meet a requirement or where the user selects the involvement of a neural network. The data required as inputs to the neural networks are fed from the GIS operation, from which neural networks map out an intuitive solution [9], [10], [11], [12]; these results are then considered as fuzzy sets and a fuzzy analysis is carried out to generate outputs as attributes and images as well as corresponding numerical values. These results can then be fed into the map algebra function together with other maps to produce synthesised results or provide direct results to assist the final decision making process. In the process overall, neural networks play an advanced function to solve those otherwise difficult mapping problems.

3 The Role of Neural Networks and Fuzzy Sets

Considering the spatial and temporal distribution of aircraft noise, all locations in proximity to the same airport experience similar weather conditions and the same noise sources; their spatial difference is the main factor in determining the distribution of aircraft noise impacts. Therefore, we establish a set of neural networks using distance to consider the macro spatial relationships between aircraft noise and its distribution. If the input noise level disturbs a local community, then the output frequency of the neural networks can be taken as an indicator for the disturbance caused by its corresponding noise level. If we consider this local community as a fuzzy subset, the normalised frequency value could be considered as the fuzzy membership of those properties or persons disturbed by the corresponding aircraft type. In this way, the output of a neural network becomes a set of fuzzy subsets. Their members are the properties or people in the corresponding locality, and their memberships are the normalised frequencies. Unlike other approaches [14], here we do not consider the shape as fuzzy, and we can calculate the corresponding number of disturbed properties or people based on a fuzzy area [15]. A simplified diagram is shown in Figure 2.

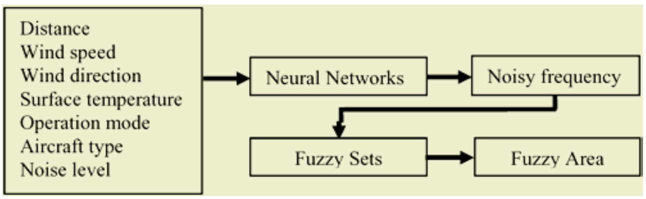


Fig. 2. Neural network inputs and outputs.

For the sake of simplicity, fine grain spatial details have not been included in our model since, following trial tests, we found that aircraft noise data are sufficiently complicated when considering only one aircraft type (e.g., Boeing 747-400). It is difficult to establish a satisfactory neural network with the inclusion of different aircraft types, hence we keep it separate from the input factors and establish an embedded network when considering aircraft types. The output is based on the frequency of aircraft noise rather than absolute noise levels. For example, flights regarded as ‘really noisy’ are those departures or arrivals with a noise level higher than some threshold. In this sense, the same aircraft may not necessarily be as noisy for each of its operations, since its noise output will depend on many other complex interactions within the system. It is very difficult to monitor all factors which influence aircraft noise levels, but it is relatively apposite to test the probability of an aircraft event being noisy for each kind of aircraft during its operation at the same airport. By the probability of an aircraft event being noisy, we mean the probability for the aircraft noise level to go beyond some given threshold for a specific geographical location at the airport. Given the noise monitoring level $\{L_1(\mathrm{x}), L_2(\mathrm{x}), L_3(\mathrm{x}), \ldots, L_n(\mathrm{x})\}$ when the distance between an aircraft flight trajectory and the location under scrutiny is within $[d_1,d_2]$ for the same aircraft in the same operation mode under similar weather conditions, the 'noisy probability' $p(x,t)$ is calculated as

$$p(x,t)=\frac{|\{L_i(x):L_i(x)>t\}|}{n} \quad . \tag{1}$$

Here, x is the specified location and n is the cardinality of the data monitored for that location with distance in $[d_1,d_2]$. Note that $[d_1,d_2]$ is a 'grey number' [16] and its degree of greyness determines the resolution of the noisy probability. L_t is the threshold for the 'noisy' noise level. However, a lower degree of greyness has less data available, hence it has to be balanced between the high resolution and reliable probability. With noisy probability, the noisy frequency $F(x,t)$ is easy to derive:

$$F(x,t) = p(x,t) \times N_f \quad . \tag{2}$$

where N_f is the number of flights operated at the airport during the given time interval. It is clear that $p(x,t)$ is in fact the normalised frequency, and therefore it can serve as the fuzzy membership.

The notion of the area (AR) of a fuzzy set can now be introduced (15). For a fuzzy set E, formed by discrete pixels referenced by a coordinate pair (x, y) with area $A(x, y)$,

$$AR(E) = \sum_x \sum_y \mu_E(x,y) A(x,y) \quad . \tag{3}$$

Each locality can be considered as a fuzzy entity consisting of properties or people. We consider the area here as the number of properties or people in a given area, hence the area of the entity could be approached using a series of tiny grids. The fuzzy membership of properties or people in each tiny grid is considered to be the same. Note that if we divide the entity using large grids, then the influence of the fuzzy boundary could be neglected. Therefore, Equation (3) could be adapted to calculate the number of properties or people who suffer a disturbance.

$$N_E(x,t) = \sum_i p_E(x,t,i) N_E(i) \quad . \tag{4}$$

Here, $N_E(x,t)$ is the equivalent number of disturbed properties or people in the corresponding locality, $p_E(x,t,i)$ is the fuzzy membership of grid i, and $N_E(i)$ is the number of properties or people in grid i.

For simplicity, we consider each flight takes a similar amount of time to take-off from or approach the airport. Hence each flight can be counted only once in the frequency calculation. For the situation where we have more than one trajectory, the probability is calculated separately for each trajectory. In the final analysis, a general probability is then calculated according to their weight in terms of number of operations.

4 Example Output Maps

The training of a neural network is often a difficult task, especially for problems like noise where the data have such large fluctuations. Partly in response to such problems, we have undertaken extensive work in improving the quality of training using different methods and analysing the different impacts of input factors [17], [18], [19]. For a trained neural network with data from a single aircraft type, output based on a number of virtual flight tracks is shown in Figure 3. The network was trained using aircraft noise monitoring data from a large UK international airport for a single aircraft type – the Boeing 757. The figure to the right highlights those localities and their populations subject to noise disturbance by this aircraft type's movements.

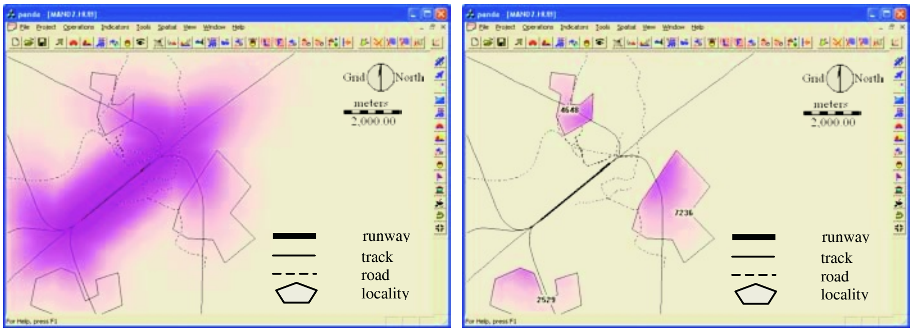


Fig. 3. Noisy frequency distribution derived from neural networks.

5 Conclusions

Artificial neural networks and fuzzy sets are typically two mainstream applications currently in use to overcome the uncertainty and complex interactions associated with solving real world problems. With the increasing awareness among airport operators of the need to ensure the sustainable development of their airports, ever more monitoring data are being recorded around the major airports of the world. This has laid a solid foundation for neural networks to be applied to the environmental evaluation of airport operations where huge uncertainties and complex interactions exist. The interpretation of the results from these neural networks could however be further enhanced using fuzzy sets. In this paper, we have demonstrated a system that integrates both neural networks and fuzzy sets with Geographical Information Systems as a method to assess the exposure to noisy flights of specific local communities in proximity to an airport. Our results show that their integration can provide an alternative to existing modelling methods.

References

1. Morrell, P., Lu, C.H.: Aircraft Noise Social Cost and Charge Mechanisms - A Case Study of Amsterdam Airport Schiphol. Transportation Research, Part D: Transport and Environment, **5** (2000) 305-320
2. Connor, T.S.: Integrated Noise Model - The Federal Aviation Administration's Computer Program for Predicting Noise Exposure Around an Airport. Proceedings of the International Conference on Noise Engineering. INTER-NOISE80 (1980) 127-130
3. Yang, Y., Gillingwater, D., Hinde, C.: An Intelligent System for the Sustainable Development of Airports. Proceedings of the 9th World Conference on Transportation Research (WCTR) (2001b) F5-02
4. Yang, Y., Gillingwater, D., Hinde, C., Upham, P.: A Scaled Approach to Developing a Decision Support System for Sustainable Airport Development. Proceedings of the UK Sustainable Cities and Aviation Network (SCAN-UK) Conference, Manchester (2001d)
5. Hechi-Nielson, R.: Neurocomputing. Addison-Welsley, Reading MA (1990)
6. 6. Zadeh, L.: Fuzzy Sets. Information and Control, **8** (1965) 338–353
7. Burrough, P.A., McDonnell, A.M.: Principles of Geographical Information Systems. Oxford University Press, Oxford (1998)

8. Rigol, J.P., Jarvis, C.H., Stuart, N.: Artificial Neural Networks as a Spatial Interpolation. International Journal of Geographical Information Science, **15** (2001) 323-343
9. Yang, Y., Rosenbaum, M.: Spatial Data Analysis with ANN: Modelling to Manage Geoenvironmental Problems Caused by Harbour Siltation. Proceedings of International Symposium on Spatial Data Quality (ISSDQ'99) (1999) 534-541
10. Yang, Y., Rosenbaum, M.: Artificial Neural Networks Linked to GIS for Determining Sedimentology in Harbours. Journal of Petroleum Science and Engineering, 29 (2001) 213-220
11. Yang, Y., Rosenbaum, M.: Artificial Neural Networks Linked to GIS. In: Nikravesh, M., Aminzadeh, F., Zadeh, L.A. (Eds.): Developments in Petroleum Science, 51: Soft Computing and Intelligent Data Analysis in Oil Exploration. Elsevier Science (2002)
12. Yang, Y., Rosenbaum, M., Burton, C.: An Intelligent Database for Managing Geoenvironmental Change within Harbours. Environmental Geology, **40** (2001c) 1224-1231
13. Brunsdon, C., Openshaw, S.: Error Simulation in Vector GIS Using Neural Computing Methods. In: Worboys, M.F. (eds): Innovation in GIS. Taylor & Francis, London (1994)
14. Fonte, C., Lodwick, W.: Areas of Fuzzy Geographical Entities. International Journal of Geographical Information Science, **18** (2004) 127–150
15. Rosenfeld, A.: The Diameter of a Fuzzy Set. Fuzzy Sets and Systems, **13** (1984) 241–246
16. Liu, S., Guo, T., Dang, Y.: Grey System Theory and Its Application. The Science Press of China, Beijing (2000)
17. Yang, Y., Hinde, C., Gillingwater, D.: A New Method to Evaluate a Trained Artificial Neural Network. Proceedings of the International Joint Conference on Neural Networks 2001 (IJCNN'01) (2001a) 2620-2625
18. Yang, Y., Hinde, C., Gillingwater, D.: A New Method in Explaining Neural Network Reasoning. Proceedings of the International Joint Conference on Neural Networks 2003 (IJCNN'03) (2003) 3256-3260

An Artificial Neural Network Method for Map Correction[*]

Yi Chai[1,2], Maoyun Guo[1], Shangfu Li[3], Zhifen Zhang[3], and Dalong Feng[1]

[1] College of Automation, Chongqing University, Chongqing 400044, China
[2] The Key Laboratory of High Voltage Engineering and Electrical New Technology of Ministry of Education, Chongqing University, Chongqing 400044, China
chaiyi@cqu.edu.cn, gomorning@sina.com
[3] Xichang Satellite Launch Center, Xichang, Sichuang 615000, China

Abstract. Raster map should be corrected after scanned because of the errors caused by paper map deformation. In the paper, the deficiency of the polynomial fitting method is analyzed. The paper introduces an ANN (Artificial Neural Network) correcting method that utilizes the advantage of its function approximation ability. In the paper, two types of ANNs, BP and GRNN, are designed for the correcting. The comparing experiment is done with the same data by the polynomial fitting and ANN methods, utilizing the MALAB. The experiment results show that the ANN methods, especially the GRNN method, performances far better than the polynomial fitting method does.

1 Introduction

The errors are caused after the paper map is scanned because of the deformation and the errors of the scanners. So the map needs correcting. The classical method of the map correction is the polynomial fitting method that utilizes the polynomial to fit the error curse.[1] But since the map deformation is hard non-linear, if a finite order polynomial is use to correct hard non-linear deformation, the correction errors may be considerable. So a new correcting method should be developed.

ANN, as a new tool, has already been used to correct the errors in instrument measuring [2],[3]. Essentially, the map correcting is also errors correcting. Since the ANN is good at correcting the errors, it also can be good at correcting the map errors.

2 Map Deformation and Correction

The main cause of the map deformation is that the map is moisten and that the map is not level when it is scanned. The deformation of the map includes flexing and circumvolving. In this paper, the flexing deformation is considered only.

As the Fig. 1 showing, the control point C that contains the information of the image coordinates and the geographic coordinates are added into the map for correction. The control points' image coordinates can be read from the paper map directly, which can change with the map deformation. And control points' geographic coordinates are

[*] This work was supported by 863 Program Project of China (Grant No.2003AA132050) and China Postdoctoral Science Foundation (Grant No.2004035154).

the coordinates of control points in the world coordinate system, which do not change with the map deformation. When a map is made, the control point's geographic coordinates are set. And the mapping relation of any point's image coordinates and its geographic coordinates in the map are set too. So the image coordinates of any point in the map are equal to its geographic coordinates.

As Fig.1 (a) and (b) showing, when the map is deformed, the points' image coordinates are changed. The point whose image coordinates are (x_{i0}, y_{i0}) when the map is deformed is not the control point C. The control point C whose image coordinates were (x_{i0}, y_{i0}) move to the position (x_i,y_i). That's the mapping relation set before is damaged. The map needs correcting.

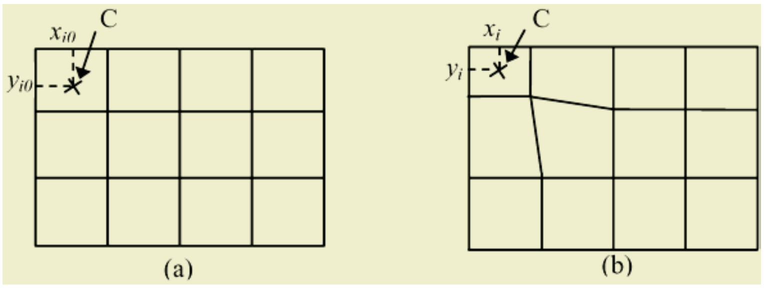


Fig. 1. The Illustration of the Map's Flexing Deformation.

As Fig.1 showing, the image coordinates of control point C in the map is changed (from (x_{i0}, y_{i0}) to (x_i,y_i)). But the geographic coordinates of them do not change, remaining (x_f,y_f). So a new mapping relation $f(\cdot)$ of the deformation map's image coordinates and its geographic coordinates can be got by comparing the control point's geographic coordinates $(x_f,y_f)$and image coordinates(x_i,y_i). This comparing is the map correcting.

The basic steps of the correcting are showed as following.

(1) Get the control points' geographic and image coordinates (x_f,y_f), (x_i,y_i);
(2) Get the mapping $f(\cdot)$of (x_f,y_f) and (x_i,y_i);
(3) Utilize the mapping $f(\cdot)$ to correct the map deformation;

The essential of the correcting is to get the mapping relation of the geographic coordinates and image coordinates. That's, the veracity of correcting is determined by that of the mapping relation.

3 The Classical Method for Map Correction

As stated in 2, the map correction is to get a mapping by the control points' geographic coordinates and image coordinates. Essentially, The mapping is a curve fitting problem.[1]

Also as stated in 2, the map deformation is non-linear, so the deformation can be represented by a polynomial. And in the paper, only the flexing deformation is discussed. So, if a map have K control points, which contains the geographic coordinates $(x_{f,l}, y_{f,l})$ and image coordinates $(x_{i,l}, y_{i,l})$ $(l<=K)$.

Then:

$$x_f = f_x(x_{i,l}) = a_{x0} + \sum_{m=1}^{K-1} a_{xm} x_{i,l}^m . \tag{1}$$

$$y_f = f_y(y_{i,l}) = a_{y0} + \sum_{m=1}^{K-1} a_{ym} y_{i,l}^m . \tag{2}$$

The coefficient ($a_{x0}, a_{x1}, ..., a_{xK-1}$) and ($a_{y0}, a_{y1}, ..., a_{yK-1}$) of (1) and (2) can be obtained by the least square method. Since (1) and (2) is similar, in the paper, only equation (1) needs discussing. For *k* control points, *k-1*-order polynomial, equation (3), can be got by the least square method[1],[4].

$$x_{f,l}^* = f_x^*(x_{i,l}) = a_{x0}^* + \sum_{m=1}^{K-1} a_{xm}^* x_{i,l}^m . \tag{3}$$

And the error e_p is:

$$e_p = x_{f,l}^* - x_{f,l} . \tag{4}$$

Theoretically, *K-1*-order polynomial can be obtained by the least square method with the *k* groups of data corresponded with *k* control points.

4 The ANN Method for Map Correction

The map deformation is a hard non-linear mapping from the geographic coordinates space to the image coordinates space. if with the correcting method of polynomial fitting, the correcting error is bigger. Since ANN has a perfect ability of describing non-linear mapping, the ANN can have a nice performance at map correcting.[2],[3],[5] The Fig. 2 shows a map correction model based on ANN.

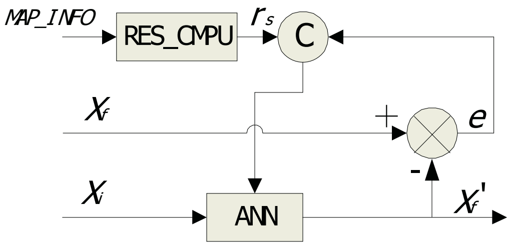


Fig. 2. A Map Correction Model Based on ANN.

There are 3 parts in the model. ANN is used for correcting. C is a comparator used for the comparing the correcting errors and the map resolving rate and adjust the ANN with the comparing result. RES_CMPU is resolving rate computing module whose inputs are the information of the map, such as the scale of the map, the coverage of the map, and etc.

In order to get better correcting effect, the correcting errors should not be greater than the resolving rate. That is:

$$e \le r_s . \tag{5}$$

r_s is the map resolving rate whose scale is *s*. r_s is the output of RES_CMPU whose input is MAP_INFO(including scale of the map, the coverage of the map, and etc.). r_s has the same unit as the correcting errors *e'*s

The quality of the map correction depends on the correcting errors. For a map with *n* control points, the correcting errors can be defined as following:

$$e = \left\| X_f' - X_f \right\| . \tag{6}$$

$X_f = (x_{f,0}, \cdots, x_{f,n})$, $x_{f,i}$ is the geographic coordinates of the control point *i*, $i = 1, \cdots, n$. $X_f' = (x_{f,0}', \cdots, x_{f,i}')$, $x_{f,i}'$ is the corrected coordinates of he control point *i*, $i = 1, \cdots, n$.

$\|\cdot\|$ is a normal. It is the mean absolute error (mae)

$$\|(y_1, \ldots, y_n)\| = 1/_n \sum_{i=1}^{n} |y_i| . \tag{7}$$

The following is the work procedure of the model showed as the figure 2.

Step 1: Make the X_i as the input of the ANN, single input and single output and the X_i the target of the ANN, set certain parameters of the ANN. and then train the ANN

Step 2: When training finished, compute the correcting errors *e*. and then with the comparator C, compare the *e* and r_s which is obtained from the RES_COMP. If the comparing result shows that the equation (5) is not satisfied, adjust the ANN's parameters. And retrain and readjust the ANN until the equation (5) is satisfied.

Step 3: Correct the map with ANN obtained in step 2.

5 Experiment

5.1 Experiment Environment and Data

In order to compare the correction method based on ANN with that based on polynomial fitting, a experiment is done with MALAB. In this experiment, a map whose scale is 1/50000 has 42 control points. So,we can get the approximate $r_{s,}$ as the equation (8):

$$r_s = \frac{r_{50000}}{R/360} . \tag{8}$$

r_{50000} is the resolving rate of the map whose scale is 1/50000. $r_{50000} = 5m$.*R* is the perimeter of the earth. $R \approx 4 \times 10^7 m$. So, $r_s \approx 4.5 \times 10^{-5}$.

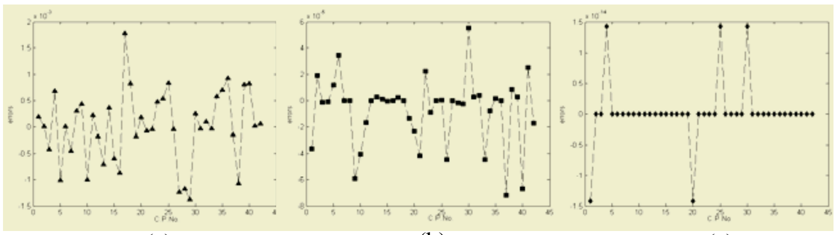
And the 42 control points provide 42 couples of the image and geographic coordinates showed in Table 1.

Table 1. The 42 couples of the image and geographic coordinates.

No.	Img. Coor.	Geo. Coor.	No.	Img. Coor.	Geo. Coor.	No.	Img. Coor.	Geo. Coor.
1	7584.052404	102.293980	15	4933.489499	102.191880	29	5651.489064	102.220720
2	105.529725	102.002740	16	1035.505405	102.040530	30	6740.499390	102.262230
3	2417.506804	102.092930	17	1180.482484	102.043910	31	7252.512877	102.282230
4	5817.488534	102.224600	18	3687.515694	102.142700	32	7540.482644	102.293000
5	6920.500409	102.270470	19	6495.485127	102.252490	33	7510.497292	102.292280
6	640.452647	102.023300	20	7570.489869	102.293670	34	7036.520483	102.273190
7	3604.482196	102.140740	21	563.497200	102.021470	35	5008.517511	102.193650
8	5788.490256	102.223930	22	540.511781	102.020910	36	5021.482497	102.193950
9	6752.503425	102.262510	23	2238.504787	102.084720	37	2685.495387	102.103230
10	3329.493194	102.130300	24	3667.489024	102.142210	38	1282.506049	102.050350
11	678.500360	102.024190	25	5830.486743	102.224910	39	4239.509027	102.163630
12	2432.486498	102.093280	26	45.505123	102.001350	40	2980.502066	102.114140
13	6011.496102	102.233140	27	4896.499901	102.191020	41	15.484890	102.000650
14	4754.486402	102.183700	28	4096.502785	102.160280	42	7669.489096	102.295980

5.2 The Polynomial Fitting Method Experiment

Theoretically, the maximum order of polynomial fitted is 41 with the 42 couples of data by the least square method. Computed with MALAB, when the order of the polynomial is 21, the mae, the correcting errors *e*, is minimum. It's 5.1979e-004.Since $e>r_s$, the correction quality is not good. And the Fig.3(a) gives the correction error of each control point.



(a) (b) (c)

Fig. 3. The Correcting Error With different Method. (a):the Polynomial Fitting Method.(b): the BP ANN method (c): the GRNN ANN method.

5.3 The ANN Method Experiment

In order to compare the correction performance, A 3-layered BP and a generalized regression neural network (GRNN) are designed.

For the 3-layered BP[3][4], a BP_{10} ANN is designed which has single neuron in input, output layer and 10 neurons in the medium layer. Its epochs are set to 10000. And its goal is mse (mean square error)<1e-9. With the data given in table 1, when the epoch reaches 10000, the goal >1e-9. The training is failure. And the mae, the correction errors *e*, is 2.7582e-004. Since $e>r_s$, according to the 4th section of the paper, the BP_{10}ANN need adjusting.

So, the number of the neurons in medium layer is set to 40. And the other parameters are not changed. Then a BP ANN, BP_{40}, is obtained. When the epoch reaches 10, the goal <1e-9. The training is succeeded. The mae is 1.6616e-005. Since $e<r_s$, according the 4, the BP_{40} is the ANN wanted. And the train stops. And each control point 's correcting error with BP_{40} method is showed in Fig.3(b).

For the GRNN based on radial base function (RBF), A GRNN is designed, whose SPREAD is 2, input is image coordinates and the target is geographic coordinates. The mae, the correction errors e, is 1.6918e-015. It *is* far less than the r_s. And each control point 's correcting error with the GRNN method is showed in Fig.3(c).

6 Conclusions

Correcting raster map is the important procedure of map digitalization. The precision of digital map depends greatly on the precision of the correction. Traditional correction method is polynomial correction method whose correcting ability is limited for the large-scaled map deformation which is a hard non-linear. A new correction method needs to be developed to satisfy the requirement of the high precision.

Since the ANN performs well at the non-linear mapping, it can obtain good effect if applied to the map correction. The experiment results stated in Fig.3 show that ANN method, especially the GRNN method, performs far better than the method of polynomial by one order of magnitude. The correcting performance is improved.

References

1. Wu. X.: MAPGIS Geography Information System, Press House of Electronic Industry Beijing (2004)
2. Massicotte, D.; Megner, B.M: Neural-Network-Based Method of Correction in a Nonlinear Dynamic Measuring System. Instrumentation and Measurement Technology Conference, 1999. IMTC/99. Proceedings of the 16th IEEE Vol.3, (1999) 1641 – 1646
3. Arpaia, P., Daponte, P., and Grimaldi, D.: ANN-Based Error Reduction for Experimentally Modeled Sensors, IEEE Transactions on Instrumentation and Measurement, **51** February (2002) 23-30
4. Yi, D., Shen, Y., and Li, Y.: Computation Methods. Zhejiang University Press, Hangzhou (1989)
5. Yuan, Z.: Artificial Neural Network and Its Applications, Tsinghua University Press Beijing (1999)

An Effective Two-Stage Neural Network Model and Its Application on Flood Loss Prediction

Li Yang, Chun Zuo, and Yuguo Wang

Institute for Software, Chinese Academy of Sciences, Beijing 100080, China
{yangli,zuochun,wyg}@sinosoft.com.cn

Abstract. In this study, a two-stage radial basis function neural network based model is employed to develop flood loss prediction model for insurance company. In the first stage, self-organizing map clustering is used to measuring the similarity of the input data, unlike k-means approach, number and centers of clusters can be determined dynamically and automatically. The Value-related Self Organizing Map (VSOM) is proposed to improve predicting accuracy for high loss subject. During the second stage, the weights from the hidden layer to output layer are determined by multivariate linear regression method. Simulation results show that the proposed approach can be applied successfully to build flood loss prediction models and provide higher accuracy and more direct decision support compared to current approaches.

1 Introduction

The flood hazard and the potential flood loss are of great interest of insurance companies. However the Hydrology and Geography based flood loss prediction systems are marvelously complex and notoriously nonlinear. In order to build such a model for insurance companies one must know the behavior of hydrological processes under different terrain conditions. Some meteorological, physiographic and human factors such as rainfall, terrain and drainage etc. could affect the actual flood loss. Hence, the relation of various factors and flood loss has always received great attention. Flood loss prediction methods vary from fairly simple relationships between water depth and loss to rather complex models[1]. One of the disadvantages of using these models is that the parameters of the models are usually difficult to determine from observed data. Another lack of existing approaches is that they often have some assumptions such as it is a simple linear model and rainfall is uniformly distributed in time and space. Unfortunately, these assumptions cannot be satisfied in most regions. Owing to lack of practicality and difficult in use, the application of such sophisticated flood loss prediction models in China, which has a changefully climate and diverse terrain situations, remains some distance away.

Even with all these difficulties, neural networks (NNs) can make important contributions to flood loss prediction. A very important feature of NNs is their adaptive nature, which learns from the historical data, in dealing with nonlinear problems. Constructing an input-output mapping without physical models is also another useful property of NNs. Thus, NNs can be viewed as nonlinear self-organizing model. Recently, they have been used for modeling hydrological processes[2]. By using NNs, the hydrological processes can be viewed as black boxes. Thus poorly defined or misunderstood relationships can be described by NN. Among many types of NN, RBFNN is selected in this study for its fast training speed[3]. Unlike traditional approach, the Self-Organizing Map (SOM) instead of the K-means is employed in this study to determine the number and center dynamically and automatically.

In flood loss data, the majority is about low loss, while the minority is about high loss data, which is of particular interest of insurance companies because it can directly guide the loss reduction work. Most algorithms generate models that minimize the overall mean square error. When dealing with unbalanced data like flood loss data, this leads to either trivial models that completely ignore the minority data or models with many small neurons that tend to over-fit the training sample. In order to tackle this problem, this paper proposed a novel value-related self-organization maps (VSOM) to make the neurons clustered by SOM moved to desired location. Experiments show proposed approach is effective in flood loss prediction model construction.

The rest of this paper is arranged as follows: in section 2, RBFNN is briefly introduced, section 3 describes the proposed SOM and VSOM network training approach, in section 4, the simulation results of proposed model and compared model is presented. The conclusion is given in the last section.

2 Radial Basis Function Neural Network

RBFNN have currently been widely accepted in many areas, such as control and classification, for nonlinear input-output mapping. An important property of these RBFNNs is that high-dimensional space nonlinearity can be transformed by a set of nonlinear radial basis functions by linear combination. Requiring less training time is another important feature of an RBFNN. For the purpose of faster training speed, RBFNN with the hybrid learning scheme applied herein have a feed-forward structure that involves three layers, as shown in Fig. 1. The input layer is composed of *n* input nodes. The only hidden layer consists of *k* locally tuned units and each unit has a radial basis function acting like a hidden node. The hidden node output $R_i(X)$ calculates the closeness of the input and projects the distance to an activation function. The activation function of the *i*th hidden node is given by

$$R_i(X) = e^{-\frac{\|X-C_i\|^2}{2\sigma_i^2}} \quad i = 1,2,\cdots,k . \tag{1}$$

where X is the input vector with n dimensions; C_i is the center of the radial basis function for hidden node *i*; and σ_i is a parameter for controlling the smoothness properties of the radial basis functions. The third layer of the network is the output layer with L nodes that are full interconnected to each hidden node. The output of the network is the sum of linear weighted $R_i(X)$

$$y = \sum_{i=1}^{k} w_i R_i(X) . \tag{2}$$

where y is the output value of the network, w_i is the synaptic weight between the *i*th node of hidden layer and the output layer.

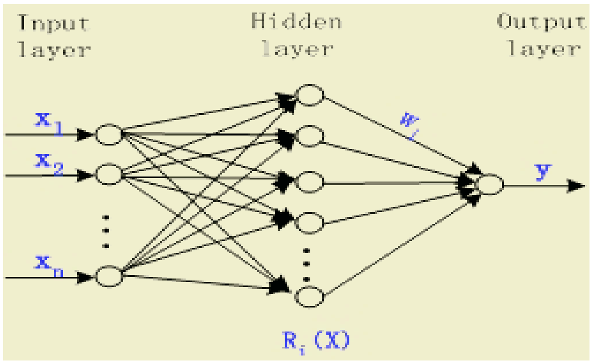


Fig. 1. Topology of RBFNN used in this study.

3 Network Training

Training a RBFNN occurs in two stages. During the first stage, the unsupervised training scheme employs clustering for locating hidden unit's receptive field centers. Several algorithms including K-means and genetic algorithm are employed in determining the hidden layer structure. Instead of the K-means clustering method, the Self-Organizing Map (SOM) method is employed in this study. The advantage of SOM method is that the number, centers and σ_i of radial basis functions can be determined dynamically and automatically. SOM is one of the most popular data mining techniques, which is especially suitable for high dimensional data visualization and clustering[4]. It uses an unsupervised learning algorithm for creating a set of prototype vectors representing the data. Moreover, a topology preserving projection of the prototype from the original input space onto a low dimensional lattice (usually a 2D lattice) is carried out.

A SOM is formed by an arbitrary number of clusters $C_1,\ldots,C_m$, located on a 2D lattice($C_k=(x,y)$ represents the position of the cluster on the lattice); each of the cluster C_k has an associated prototype vector $c_k=(c_{k1},\ldots, c_{kd})$, which describes the position of the cluster's center on the d-dimensional data space.

The vectors of the SOM are first initialized to random values. The goal of the training algorithm is iteratively adapting the prototype vectors. The batch implementation of the training proceeds in cycles; on each cycle, all data vectors are considered iteratively, one at a time(v_i), and the best-matching(or "winning") prototype c_{ki} is obtained as the one minimizing the Euclidean distance to the data vector:

$$\| v_i - c_{ki} \| = \min_k \| v_i - c_k \|, k = 1,\ldots,m. \tag{3}$$

After each cycle the prototypes are moved according to the closer data vectors and also to its neighbors:

$$c_j = \frac{\sum_{i=1}^{n} v_i h(\| c_j - c_{ki} \|)}{\sum_{i=1}^{n} h(\| c_j - c_{ki} \|)}, j = 1,\ldots,m. \tag{4}$$

where the function $h(\| c_1 - c_2 \|)$ is a neighborhood kernel, which determines the rate of change around the winner unit (usually a Gaussian function is considered: $h(x)=exp(-x/s(t))$). The variance of the Gaussian neighborhood kernel $s(t)$ decrease monotonically with time, softening the topologic constraints- a linear decay to zero is usually chosen for these functions. The neighborhood adaptation mechanism is what makes SOM different from other clustering algorithms, so neighboring clusters in the 2D lattice space are quite similar, while more distant clusters become increasing diverse. For a detailed description of different implementations of the method, the reader is referred to [4].

Although the numbers and centers of clusters can be determined by SOM adaptively, the unbalanced nature of the relationship between the flood loss data and loss reduction operations is not considered. The majority of flood loss data is about low loss, the number of high loss data which have important operational value to insurance companies is relatively small. This phenomenon makes the prediction accuracy for high loss data is unsatisfying and unable to provide valuable support for loss reduction work. To solve this problem, this paper proposed a Value-related Self-Organizing Map (VSOM) to adapt to the characteristic of the flood loss data. The main idea is reinforcing learning based on SOM clustering process, training data which have desired results will be picked up, after the whole clustering process has finished, the selected data is used to reinforce the clustering. The purpose is to make the whole neurons of VSOM move towards the reinforcing vector space. The clustering result will move towards the desired direction after several reinforcing process and is not constrained by the original data distribution. The detailed step can be described as follows:

1. initialize the vector of VSOM to random values
2. determine the size of network topology and learning rates
3. determine the stopping criterion of VSOM processes, i.e. iteration number of calculation and convergence error
4. for every training example, calculate the winning neuron based on Euclidean distance $\| v_i - c_{ki} \| = \min_k \| v_i - c_k \|, k = 1,...,m.$
5. centered with the wining neuron, modify the weights of neurons in the neighbor area $w_{ki}(t+1) = w_{ki}(t) + \Delta w_{ki}$
6. adjust the learning rate and shrink the neighbor area in the topology accordingly.
7. if the stopping criterion is not met, go to step 4.
8. select the data (i.e. high loss data in this study) need to be reinforced from the training examples.
9. determine the stopping criterion of second calculation, i.e. iteration number of computation and convergence error
10. iterate the process of step 4~7 based on current clustering results and reinforcing data.

After the clustering is completed, the radial basis functions are kept fixed. During the second stage, the weights w_i are determined to let the output of network y approximate to the actual value $\tilde{y}$. The supervised training algorithm aims to minimize the following root mean square error,

$$RSME = \sqrt{\frac{\sum_{i=1}^{k} (\tilde{y}_i - y_i)^2}{k}} \,. \tag{5}$$

where $\tilde{y}_i$ and y_i are ith set output and target respectively. k is the number of neurons in hidden layer. Since the outputs of the network are linear combinations of the outputs of the hidden layer, the multivariate linear regression can be used to determine the weights. By using the multivariate regression method to minimize RSME, the RBFNN will have the best approximation property. It can be demonstrated the multivariate linear regression method also consumes less time for model training when compared with those of other three-layer NNs.

4 Experiments

4.1 Study Area

The model developed in this paper was applied at a study area on Shenzhen City (Fig 2), a coastal city in southern Guangdong province. Typhoon and heavy rainfall is the main cause of flood hazard influencing Shenzhen.

4.2 Model Construction

The factor used in this study is listed below (Table 1), the terrain factor is derived from famous TOPMODEL[5] based on DEM (Digital Elevation Model), and the drainage factor is derived from our partner's work[6], The Rainfall factor and Typhoon factor can be acquired from meteorological forecasting, The insurance subject loss factor is derived from historical insurance loss data, The output variable y is insurance loss value.

Table 1. Input factors used in this study.

Factor name	Formula	Description
x_1=Terrain factor (wetness index)	$ln(\frac{a}{tan\beta})$	a : specific contributing area $tan\beta$:gradient slope
x_2=Drainage factor	$\frac{1}{n}\frac{\pi d^2}{4}S^{1/2}$	n, d, S: coarse degree, diameter and slope degree of drainage pipeline
x_3=Rainfall factor	Precipitation*duration	from meteorological forecasting
x_4=Typhoon degree	0~12	from meteorological forecasting
x_5=Insurance subject loss factor	0~1	from insurance historical loss data e.g. home appliances=0.4,bicycle=0.2 …

4.3 Results

Performance of proposed models is compared with logistic regression, RBFNN using normal k-means clustering algorithms. Insurance flood loss data of People Insurance Company of China from 1992 to 2003 is devoted to training the model, data collected from Dujuan typhoon(September 2003) is used for testing the constructed model. The RMSE is used for approaching to comparison of the performance of the proposed model. The data is characterized as low loss data if the loss value is measured below 10,000 yuan (US$1205) and high loss data otherwise.

Table 2. Simulation results of different models.

Model	RMSE	RMSE for low loss data	RMSE for high loss data
Logistic Regression	23.28	22.68	25.13
k-means based RBFNN(k=8)	19.13	18.59	23.71
SOM based RBFNN	14.18	13.56	19.79
VSOM based RBFNN	14.55	15.96	12.77

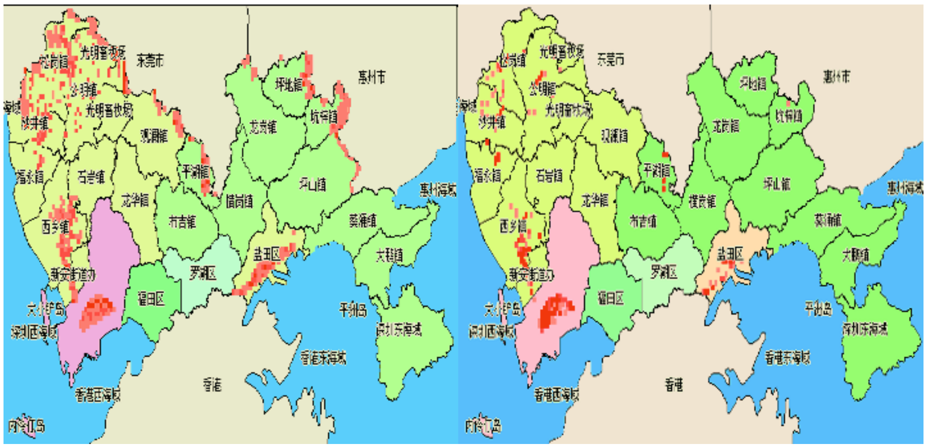


Fig. 2. Prediction result of proposed model (left) and actual flood loss map (right).

Table 2 shows the simulation results of different model. It can be observed that the SOM based model have best overall prediction ability (lowest RMSE) on all data, but VSOM based model exhibits best prediction ability on high loss data, So it can provide more direct support for insurance companies to reduce loss.

The prediction results of VSOM based RBFNN model can be visualized as left part of Fig. 2, the red dot means the location where the insurance subject would be damaged, whereas the shade of color means the damage degree. Compared with actual loss map (right part of Fig. 2), the prediction results is relatively larger than actual loss, this is partly due to that some defense methods had already been taken to reduce loss. The prediction results can be used effectively to guide the loss reduction work.

5 Summary and Conclusions

In this study, an effective two-stage radial basis function neural network based model is employed to develop flood loss prediction model. During the first stage, the commonly used K-means clustering method is replaced by SOM clustering for determining the characteristics of the radial basis functions automatically. In order to provide higher prediction accuracy for high loss data and more direct support for insurance companies to reduce loss, VSOM is proposed to make the clustering result move towards the desired direction. The weight between the hidden layer and output layer are chosen by multivariate linear regression method. Experiments have demonstrated that the proposed approach can not only successfully model nonlinear flood loss prediction systems for insurance company, but also provide higher accuracy and more direct decision support compared with current approaches.

References

1. Hazus Hazus: Flood Loss Estimation Model. http://www.fema.gov/hazus/fl_main.shtm
2. Chang, F.J., Chen, Y.J.: A Counterpropagation Fuzzy Neural Network Modeling Approach to Real-Time Streamflow Forecasting. Journal of Hydrology, **254** (2001) 153-164
3. Chang, F.J., Liang, J.M., Chen, Y.C.: Flood Forecasting Using Radial Basis Function Neural Networks. IEEE Transactions on Systems, Man, and Cybernetics, Part C, **31** (2001) 530-535
4. Oja, E., Kaski, S.: Kohonen Maps. Amsterdam, Elsevier (1999)
5. Wang, L., Qin, Q., Li, J., et al.: Study on the Disaster Analysis Model of City Water-Logging Based on GIS. Science of Surveying and Mapping, **29** (2004) 48-51
6. Beven, K.J., Lamb, R., Quinn, P., et al.: TOPMODEL. IN: Singh, V.P. Computer Models of Watershed Hydrology. Water Resources Publication (1995) 627-668

An Artificial Neural Network Model for Crop Yield Responding to Soil Parameters

Gang Liu, Xuehong Yang, and Minzan Li

China Agricultural University, Beijing 100094, China
pac@cau.edu.cn

Abstract. This paper presents an artificial neural network model for crop yield responding to soil parameters. The experimental data had been obtained via a precision agriculture experiment, which is carried out by PAC in a demo farm locating in Shunyi district, Beijing in 2000. The model has been established by training a back propagation neural network with 58 samples and tested with other 14 samples. The model consists of 6, 11 and 1 processing units in the input, hidden and output layers, and the step length is 0.05, the momentum coefficient is 0.5. The training was terminated after 20000 times and the convergence effect was very good. The training results are that the correlation coefficient is 0.916 and the average error value is 2.8×10-2. It shows that the model can precisely describe crop yield responding to soil parameters.

1 Introduction

Precision agriculture(PA) has become a main aspect of agriculture sustainable development, which could make us mostly free the potential of farming land, increase the using efficiency of water and fertilizer, prevent our environment from being polluted, and greatly improve crop yield and quality[1],[2]. Then there is a growing perception that more researchers had concentrated themselves on precision agriculture, especially on the spatial and temporal variability in field over the past two-decades. Given the crop growth condition in the field context is complicated and the agronomic factors involving in crop production are numerous, it is important to research on the key soil parameters affecting the crop yield and the relationship between the soil parameters and the crop yield.

It is naturally nonlinear agricultural system for most of the constitutive elements of biological functions [3],[4]. So there are most of models structured via nonlinear analysis. But they have many linear assumptions taking simplify research into account. For example, the correlation analysis and regression analysis are mostly used to establish the mathematical model between crop yield and soil parameters. A perfect model usually needs a mass of surveyed data (such as >100 samples). Otherwise, the model may make a biased estimate or an invalid forecast[5],[6],[7].

In this paper, an Artificial Neural Network (ANN) model for crop yield responding to soil parameters was established by training a Back Propagation (BP) neural network with 72 samples, which is obtain via experiment.

2 Materials and Methods

2.1 Field Experiment

This experiment had been carried out in the demo farm, which locates in Shunyi district, Beijing. The field is about 360m*300m, 11 hectare. The crop rotation is winter wheat and summer maize. The field was divided into 40m*40m grids using Trimble AgGPS 132, and 72 soil samples were collected at 0-20 cm depth. Most of soil properties had been analyzed in 2000. They are soil moisture, N(Nitrogen), P(Phosphorus), K(Potassium), SOM(Soil Organic Matter) and N_{total}. The yield data of winter wheat were collected in the same position. A spatial map on field boundary and sampling site had been made via GIS ArcView. The 72 sets of data had been as the training data of BP for acquainting the weights of nodes, which are distributed in the input, hidden and output layers.

2.2 The Construction of the ANN

A BP-ANN usually consists of several consecutive layers, i.e. an input layer, a number of hidden layers and an output layer. For producing a nonlinear relationship between input and output, each of nodes in the input layer is connected with other nodes in the subsequent layer by weighted connections. It depends on domain problem requirements there are how many nodes in input and output layers. On the contrary, it is arbitrarily there are how many hidden layers and nodes in them[8],[9],[10].

The proposed ANN model has adopted the structure of three-layer BP network and there is only one hidden layer. The input nodes are soil moisture, N, P, K, SOM and N_{total}, the output layer includes only one node, i.e. crop yield. All weights and thresholds are assigned random values at the beginning and updated based on the trained results of ANN. Figure 1 illustrates the architecture of the proposed model based on BP ANN.

3 The Network Training and Forecast Results

3.1 The Pretreatment of Sample Data

Sigmoid function had adopted in the hidden layer of the proposed BP ANN model, which has the characteristic that the middle part is sensitive to the change of input, but the response from both of endpoints is slow. Then the network input sample should be compressed. Furthermore, taking the input samples are from different dimensions and discernment accuracy should be not affected into account, each of input data needs to be transformed into the format of 0-1. The process is called normalization. The calculation method lies in equation 1.

$$x' = (\max - x) / (\max - \min)\ . \tag{1}$$

Where x represents the raw data, max represents the maximum value of the raw data, and min represents the minimum value of the raw data.

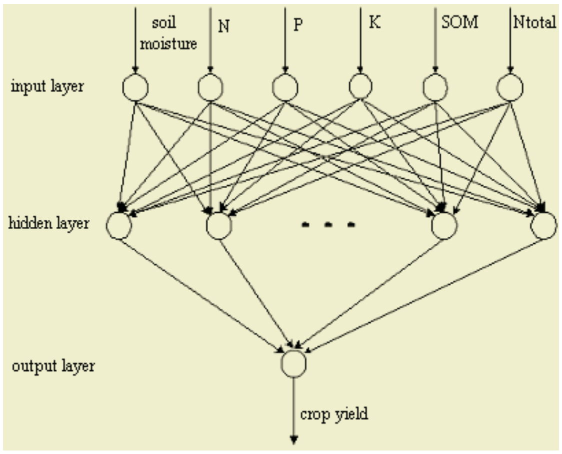


Fig. 1. The architecture of the proposed model based on BP AN.

3.2 The Ascertainment of the ANN's Training Parameters

Based on the above analysis, the proposed model is named with BP-ANN (the 3-q-1 structure: three layer-q nodes in the hidden layer-one output node), which contains one hidden layer with activated function of logarithm S type and the output layer uses linear function. The node of hidden layer q is related to the concrete problem, and the node of hidden layer q is 11 in this paper.

NeuroShell 2 software of neural network was used to train network. 58 out of 76 sets of training data were used. The step length is usually within 0.01-0.8 and 0.05 is selected in order to the error value of the neural network going least. The momentum coefficient is usually within 0-1 and 0.5 is selected in order to the neural network having a good stability and a good convergence. The training was terminated after 20000 times of iterative when there had very good convergence effect. 14 sets of validation data were used to calculate the prediction error. The training results show that the correlation coefficient is 0.916 and the average error value is 2.8×10-2. It proves that the model can precisely describe crop yield responding to soil parameters.

3.3 Using BP Model for Crop Yield Responding to Soil Parameters

Crop yield can be estimated based on the proposed model. In order to further analyze the model, one parameter is selected randomly from the six parameters, 72 sample values are divided equally into six groups from minimum to maximum, every group value is calculated averagely and six simulating values can be obtained. Other five soil parameter values are calculated averagely among these 72 sample values. Figure 2 shows the relationship between crop yield and soil parameters.

It shows that not all of relationship between crop yield and soil parameters is linear, so traditional analysis methods, such as correlation analysis and regression analysis, are not appropriate for establishing a mathematical model between crop yield and soil parameters. It also shows that crop yield ascends differently for different soil parameters, but the crop yield ascends quickly with soil moisture, N and P. It indi-

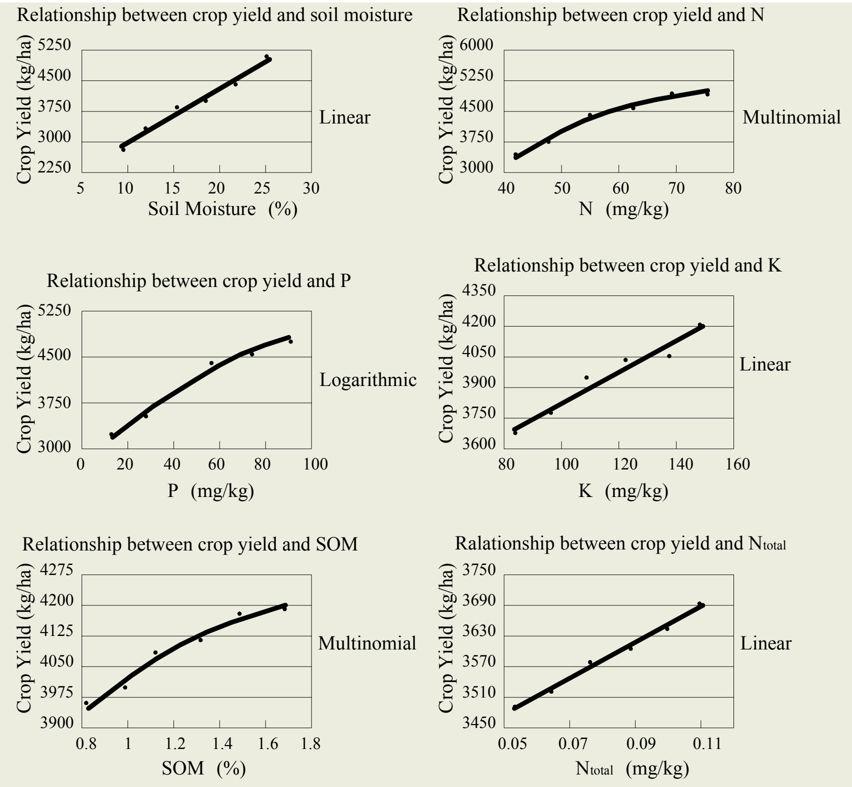


Fig. 2. The relationship between crop yield and soil parameters.

cates that the key soil parameters affecting crop yield are soil moisture, N and P. The order of these soil parameters that affect crop yield is soil moisture, N, P, K, SOM and N_{total}. Table 1 shows the relationship between crop yield and soil parameters.

4 Conclusions

The proposed BP ANN model describes the relationship between crop yield and soil parameters, the former as input factors and crop the later as output factor. It shows that the proposed model can more effectively describe the nonlinear relationship than the one based on correlation analysis and regression analysis. The impact of soil parameters on crop yield is analyzed via the proposed model. It shows that crop yield ascends with soil parameters ascending and the stages sensitivity order of crop yield is variable with different soil parameter. The order of soil parameters that affect crop yield is soil moisture, N, P, K, SOM and N_{total}.

It indicates the proposed model is a good supplement to traditional model of crop yield responding to soil parameters and provides a new method for studying crop yield responding to soil parameters. Furthermore, the proposed model has certain adaptability range given the model parameters are obtained by experiment under a certain natural condition and at the farm management level.

Each of soil parameters has variable values during different crop growing stage. Furthermore except for the above six kinds of soil parameters, there maybe other soil parameters, which affect crop yield and are not focused. This will be further research.

Table 1. Functional relationship between crop yield and soil parameters.

Soil parameters	Functional relationship	Min and max of soil factor	Min and max of crop yield	Increasing degree of crop yield
Soil Moisture	Linear	8.9 - 24.9	190 - 336	146
N	Multinomial	42 - 74	222 - 329	107
P	Logarithmic	16 - 88.6	212 - 308.6	96.6
K	Linear	84 - 148.8	243 - 280	37
SOM	Multinomial	0.86 - 1.65	263 - 280.2	17.2
N_{total}	Linear	0.06 - 0.11	234.7 - 245	10.3

Acknowledgements

This program is supported by Chinese National Funded High Science and Technology Research Project (863 project) under Grand No. 2003AA209040 and 2002AA243031. The research project is founded by the 863 Program-Study on precision agriculture platform and its application 2003AA209040 and the long-distance diagnosis platform of plant diseases and insect pests based on network 2002AA 243031.

References

1. Wang, M.: The Roadmap of ICT for Agriculture and Precision Farming In Less Developed Regions. 2004 CIGR International Conference. Beijing, China (2004)
2. Kuang, P.: The Precision Agricultural Development Approach of Developing Countries (Areas). 2004 CIGR International Conference. Beijing, China (2004)
3. Zhou, M., Chen, H.: Artificial Neural Network Model for Soil Moisture Forecast in Deficit Irrigation Rice Field. 2004 CIGR International Conference. Beijing, China (2004)
4. Huo, Z., Shi, H., Qiao, D.: Study on Artificial Neural Network Model for Crop Response to Soil Water-Salt. 2004 CIGR International Conference. Beijing, China (2004)
5. Zhang, X, Engel, B.A., Benady, N.: Locating Melons Using Artificial Neural Networks. ASAE Paper (1992)
6. Fraisse, C.W., Sudduth, K.A., Kitchen, N.R.: Evaluation of Crop Models to Simulate Site-Specific Crop Development and Yield. Proceedings of the 4th International Conference on Precision Agriculture, St. Paul, MN (1998) 1297-1308.
7. Zaidi M.A., Murase, H.: Evaluation of Seeding Vigour Using Neural Network Model under Clinostated Conditions. International Conference on Agricultural and Science and Technology, Beijing, China (2001) 504-507
8. Li, X., Qiao, X., Ye, T.: A New Fuzzy Neural Network Controller Applied in The Greenhouse. Progress of Information Technology in Agriculture, Beijing, China (2003) 546-549
9. Liu, G., Kuang, J.: A Study on Spatial Variability of Soil Nutrient within Field. Proceedings of International Conference on Engineering and Technological Sciences. Beijing, China (2000) 189-193
10. Yuan, H., Xiong, F.: A Novel Approach For Extracting Rules from Trained Neural Network. International Symposium on Intelligent Agricultural Information Technology. Beijing, China (2000) 305-309

Research on Reservation Allocation Decision Method Based on Neural Network*

Ancheng Pan[1], Yongqing Yang[2], and Hanhui Hu[1,**]

[1] School of Economics & Management, Southeast University, Nanjing, Jiangsu 210096, China
pankergy@163.com hhh@seu.edu.cn

[2] School of Science, Southern Yangtze University, Wuxi Jiangsu 214063, China

Abstract. The paper describes management-decision functions in a reservation allocation problem (RAP), and the purpose is to discuss how some recently developed techniques are combined with neural network to provide a firm's management with important tools for this storage control. The research herein is not sophisticated from the mathematical point of view , as a matter of fact, borrows heavily from work already published in neural-network-research literature. An optimization technique using linear programming is essential in solving the discrete requirements case, and it has been used to solve this problem mathematically. Consequently, a reservation distribution is defined in the domain of storage management. We propose an exact and a heuristic way that transforms the RAP into a linear programming model, and compute the optimal solution in some methods in neural network.

1 Introduction

With the explosion of information technologies over last decades, increasingly, it is important for firms to use effectively information from customers. Optimization models offer the promise of a powerful instrument for processing reservation information as soon as it is received. Online models for real-time operation planning face a host of implementation issues in order to minimize the cost of logistics. Moreover, in the conditions that a firm's storage capacity is constant, the firm must deal rationally with the relationship between reservation information and its own storage capability so as to maximize its revenue. Therefore, optimally solving the subproblem of determining a delivery schedule by balancing revenue and storage-holding-cost considerations can be very important to a firm. In this paper, we give the definition of RAP. After the detail of this description, we propose the linear programming model of RAP, and further compute solution in some methods in neural network.

2 Problem

In order to make a reservation allocation decision efficient, the RAP considered in this paper is modelled as a two-period problem, consisting of a reservation period followed

* This work was jointly supported by the National Natural Science Foundation of China under Grant 79970097.

** Corresponding author

by a service one. At the initial of the reservation period, we assume that for each reservation class $i(i = 1, 2, ..., n)$, there is $\xi = (\xi_1, \xi_2, ..., \xi_n)$ reservations currently on hand; the problem is to decide how many reservations requests to accept. The vector $\eta = (\eta_1, \eta_2, ..., \eta_n)$ is the number of class i to hold at the end of the reservation period, which we call the ordering levels . We assume the reservation demand is sufficient to allow any set of ordering levels η to be chosen, that is, the ordering levels are not constrained by future demands.

Let the vector $z = (z_1, z_2, ..., z_n)$ be the number of customers, where z_i is the number of customers from reservation class i that actually shows up for service. These are the accepted customers who survive from the reservation period to the service one. The number of $z = (z_1, z_2, ..., z_n)$ is a function of the ordering level η, and we further assume these reservation classes are independent respectively, so denote $z_i = z_i(\eta_i)$.Thenceforth, the number of reservation and no-shows is $\eta - z(\eta)$, where $z(\eta) = (z_1(\eta_1), ..., z_n(\eta_n))$. Because the fact that a customer does not survive to the service period could be due to either one, we suggest that cancellations are equal to no-shows in the model. The revenue gained by accepted a reservation of class i is denoted r_i, the refund associated with a cancellation of class i is denoted q_i , and we assume $q_i \leq r_i$ and $z(\eta) = b\eta$, where b is constant. Consequently, because of the cancellations, the total revenue and cost in the first period are $\sum_{i=1}^{n} r_i(\eta_i - \xi_i)$ and $\sum_{i=1}^{n} q_i(\eta_i - z_i(\eta_i))$ respectively, where ξ_i is the possible accepted ordering level, and satisfies $\xi_i \leq \eta_i$.

During the service period, surviving customers are allocated to storage classes to maximize the total net benefit. Suppose δ_{ij} be the net benefit of assigning the customer of reservation class i to the reservation class $j(j = 1, ..., m)$ during the service period; $c_j(j = 1, ..., m)$ be the capacity of the storage class j; z_i be the number of customers of the reservation class i that shows up at the service period (number of survivors); and ϵ_{ij} be the number of customers of the reservation class i assigned to the storage class $j(j = 1, ..., m)$ during the service period (decision variables). So the revenue is $\sum_{i=1}^{n} \sum_{j=1}^{m} \delta_{ij}\epsilon_{ij}$ during this period.

Let $\mathit{\Pi}$ be the expected value of future net benefit as a function of the ordering level η , and this is a transportation problem in which the supplies are the available storages, demands are the customers requesting service, so we are maximizing the objective function rather than minimizing. Therefore, RAP is:

$$\max \mathit{\Pi} = \sum_{i=1}^{n} r_i(\eta_i - \xi_i) - \sum_{j=1}^{m} q_i(\eta_i - z_i) + \sum_{i=1}^{n} \sum_{j=1}^{m} \delta_{ij}\epsilon_{ij}$$

$$s.t. \begin{cases} z_i = b\eta_i, \\ \sum_{j=1}^{m} \epsilon_{ij} = z_i, \\ \sum_{i=1}^{n} \epsilon_{ij} \leq c_j, \\ \eta_i \geq \xi_i, \\ \epsilon_{ij} \geq 0, \end{cases} \tag{1}$$

By further transformation, we can convert model (1) into a linear programming model (2).

$$\max \Pi = \sum_{i=1}^{n}\sum_{j=1}^{m}(\frac{r_i}{b} - \frac{b-1}{b}q_i + \delta_{ij})\epsilon_{ij} - \sum_{i=1}^{n} r_i\xi_i$$

$$s.t. \begin{cases} \sum_{i=1}^{n} \epsilon_{ij} \le c_j, \\ \sum_{j=1}^{m} \epsilon_{ij} \ge b\xi_i, \\ \epsilon_{ij} \ge 0, \end{cases} \tag{2}$$

Because the penultimate term of the objection function in the model(2) is often constant in the practice of solving the problem, in fact, we further simplify model(2), and transform it into model(3) as follows:

$$\max W = B^T Y$$

$$s.t. \begin{cases} A^T Y \le C, \\ Y \ge 0, \end{cases} \tag{3}$$

where $C \in R^n, X \in R^n, A \in R^{n\times m}, B \in R^m, Y \in R^m$.

3 The Neural Network Model of RAP

In this Section, we presented the neural network for solving RAP. Considering the following general linear programming problem

$$\max W = B^T Y$$

$$s.t. \begin{cases} A^T Y \le C, \\ Y \ge 0, \end{cases} \tag{4}$$

By the dual theory, the dual problem of (4) is as follows:

$$\min Z = C^T X,$$

$$s.t. \begin{cases} AX \ge B, \\ X \ge 0, \end{cases} \tag{5}$$

Theorem 1: (Duality Theorem) Suppose that X^* is an optimal solution of LP, then there exists Y^*, such that $(X^*, Y^*)^T$ is an optimal solution of DLP and satisfies the following complementary slackness condition,

$$\begin{cases} X \ge 0, \;\; Y \ge 0, \;\; C^T X = B^T Y \\ AX \le B, \;\; A^T Y \ge C. \end{cases} \tag{6}$$

We construct the following energy function.

$$E(X,Y)=\frac{1}{2}\|(AX-B)^{-}\|_2^2+\frac{1}{2}\|(A^TY-C)^{+}\|_2^2+\frac{1}{2}(C^TX-B^TY)^2. \quad (7)$$

Obviously, $(x^*,\ y^*)^T$ is an optimal solution of (6) if and only if $(x^*,\ y^*)^T$ is an optimal solution of (7) and $E(X,Y)=0$, that is all inequalities in (6) are satisfied. Since the problem (7) is convex unconstraint programming, it exists the optimal solution. If its optimal solution $(x^*,\ y^*)^T$ satisfies $E(X^*,Y^*)>0$, then the problem (4) has no solution, namely, (4) either has infeasible solution solution or is unbound solution. Because of this, we only design a neural network for solving (4).

Now, we construct the neural network. The state vector of neural network is U, V, which is determined by the following differential equation.

$$\begin{cases} \frac{dU}{dt}=-\nabla_X E(X,Y)=-[A^T(AX-B)^{-}+C(C^TX-B^TY)], \\ \frac{dV}{dt}=-\nabla_Y E(X,Y)=-[A(A^TY-C)^{+}-B(C^TX-B^TY)], \\ x_i=g(u_i),\quad i=1,2,...,n, \\ y_j=g(v_j),\quad j=1,2,...,m, \end{cases} \quad (8)$$

where $X\in R^n$, $Y\in R^m$ is the output variable of the network corresponding to the optimal variable, $g(\cdot)$ is a neuron action function which may be a sigmoid function.

4 Stability and Convergency of the Neural Network Model

The neural network described by (8) is globally stable and converge to the optimal solution of (7).Considering the time derivation of the energy function $E(X,Y)$ as follow:

$$\begin{aligned}\frac{dE(X,Y)}{dt}&=\sum_{i=1}^{n}\frac{\partial E}{\partial x_i}\frac{dx_i}{dt}+\sum_{j=1}^{m}\frac{\partial E}{\partial y_j}\frac{dy_j}{dt}=\sum_{i=1}^{n}\frac{\partial E}{\partial x_i}\frac{dx_i}{du_i}\frac{du_i}{dt}+\sum_{j=1}^{m}\frac{\partial E}{\partial y_j}\frac{dy_j}{dv_j}\frac{dv_j}{dt}\\&=\nabla_X E(X,Y)G'_u\frac{du}{dt}+\nabla_Y E(X,Y)G'_v\frac{dv}{dt}\\&=-\nabla_X E(X,Y)G'_u\nabla_X E(X,Y)-\nabla_Y E(X,Y)G'_v\nabla_Y E(X,Y)\le 0,\end{aligned}$$

where

$$G'_u=diag(g'(u_1),g'(u_2),...,g'(u_n))>0,$$

$$G'_v=diag(g'(v_1),g'(v_2),...,g'(v_m))>0.$$

Therefore, the energy function $E(X,Y)$ of system (7) is reduced with the time t increase until the stable state of the system is reached. If the initial point x^0,y^0 is not an equilibrium point of (8), then $X(t),Y(t)$ ensures $\frac{dE(X,Y)}{dt}\le 0$, this is, the dynamic system is globally stable and $X(t),Y(t)$ is convergent.

5 Simulation Example

Assume a firm's storage capacity is $c = (c_1, c_2, c_3, c_4) = (4500, 5200, 4200, 6250)$, and the threshold of reservations is $\xi = (1500, 2500, 3500, 2800, 1990)$. By careful analysis, the relation data about RAP could be interpreted. Namely, the coefficient is $b = 0.60$, the cancellation of reservation classes is $q = (q_1, q_2, q_3, q_4, q_5) = (0.08, 0.06, 0.07, 0.02, 0.15)$, and the revenue rates of reservation classes are $r = (r_1, r_2, r_3, r_4, r_5) = (0.15, 0.10, 0.12, 0.08, 0.06)$, and the customer reservation transferring matrix is

$$(\delta_{ij})^T = \begin{pmatrix} 0.90 & 0.85 & 0.77 & 0.70 & 0.95 \\ 0.75 & 0.80 & 0.82 & 0.75 & 0.90 \\ 0.88 & 0.86 & 0.90 & 0.78 & 0.92 \\ 0.92 & 0.78 & 0.75 & 0.65 & 0.85 \end{pmatrix}$$

Consequently, substituting these data into model (2), we can obtain the optimal RAP model in the following:

$$\max \Pi = \sum_{i=1}^{5} \sum_{j=1}^{4} d_{ij} \epsilon_{ij} - 1240$$

$$s.t. \begin{cases} \epsilon_{11} + \epsilon_{21} + \epsilon_{31} + \epsilon_{41} + \epsilon_{51} \leq 4500, \\ \epsilon_{12} + \epsilon_{22} + \epsilon_{32} + \epsilon_{42} + \epsilon_{52} \leq 5200, \\ \epsilon_{13} + \epsilon_{23} + \epsilon_{33} + \epsilon_{43} + \epsilon_{53} \leq 4200, \\ \epsilon_{14} + \epsilon_{24} + \epsilon_{34} + \epsilon_{44} + \epsilon_{54} \leq 6250, \\ \epsilon_{11} + \epsilon_{12} + \epsilon_{13} + \epsilon_{14} \geq 900, \\ \epsilon_{21} + \epsilon_{22} + \epsilon_{23} + \epsilon_{24} \geq 1500, \\ \epsilon_{31} + \epsilon_{32} + \epsilon_{33} + \epsilon_{34} \geq 2100, \\ \epsilon_{41} + \epsilon_{42} + \epsilon_{43} + \epsilon_{44} \geq 1680, \\ \epsilon_{51} + \epsilon_{52} + \epsilon_{53} + \epsilon_{54} \geq 1194, \\ \epsilon_{ij} \geq 0, \end{cases} \tag{9}$$

where

$$D = (d_{ij})^T = \begin{pmatrix} 1.20 & 1.06 & 1.02 & 0.85 & 1.15 \\ 1.05 & 1.01 & 1.07 & 0.90 & 1.10 \\ 1.18 & 1.07 & 1.15 & 0.93 & 1.12 \\ 1.22 & 0.99 & 1.00 & 0.80 & 1.03 \end{pmatrix}$$

According to Section (2) and Section (3), model (9) can be transformed into a simple linear programming problem and a neural network system can be constructed. The solution of RAP decision matrix is

$$(\epsilon_{ij})^T = \begin{pmatrix} 4500 & 0 & 0 & 0 & 0 \\ 0 & 1500 & 830 & 1680 & 1190 \\ 2930 & 0 & 1270 & 0 & 0 \\ 6250 & 0 & 0 & 0 & 0 \end{pmatrix}$$

and under the decision the firm can obtain the most revenue $\Pi = 23200$.

6 Conclusions

In the paper, we construct the model of RAP according to two-period revenue acquisition definition and transform this model into simple linear programming by the relationship between storage transferring and revenue acquisition. With the neural network method, we are not only able to devise the optimal RAP, but also give the maximum of revenue under the given storage capacity.

References

1. Tank, D. W., Hopfield, J. J.: Simple ’neural’ Optimization Network: An A/D Converter, Signal Decision Circuit and a Linear Programming Circuit, IEEE Trans. Circuits and Systems, **7** (1986) 533-541
2. Kennedy, M. P., Chua, L. O.: Neural Networks for Nonlinear Programming, IEEE Trans. Circuits and Systems, **7** (1988) 554-562
3. Baldcci, R., Maniezzo, V., Mingozzi, A.: An Exact Method for the Car Pooling Problem Based on Lagrangean Column Generation, Operations research, **52** (2004) 422-439
4. Xia, Y., Wang, J.: Global Exponential Stability of Recurrent Neural Networks for Solving Optimization and Related Problems, IEEE Trans. Neural Networks, **11** (2000) 1017–1022
5. Xia, Y., Wang, J.: A Recurrent Neural Networks for Nonlinear Convex Optimization Subject to Nonlinear Inequality Constraints.IEEE Trans. Circuits and Systems, **7** (2004) 1385-1394
6. Xia, Y., Wang, J.: A General Projection Neural Networks for Solving Monotone Variational Inequalities and Related Optimization Problems, IEEE Trans. Neural Networks, **7** (2004) 318-328
7. Powell, W.B. et al: Implementating Real-time Optimization Models: A Case Application from the Motor Carrier Industry, Operations Research, **50** (2002) 571-581
8. Leung, Y., Chen, K. Z., Gao, X. B., Leung, K. S.: A New Gradient-based Neural Network for Solving Linear and Quadratic Programming Problems, IEEE Trans. Neural Networks, **7** (2001) 1074-1083
9. Cichocki, A., Unbehauen, R.: Neural Networks for Optimization and Signal Processing, London, U.K: Wiley, (1993)

Wastewater BOD Forecasting Model for Optimal Operation Using Robust Time-Delay Neural Network*

Lijie Zhao[1,2] and Tianyou Chai[1]

[1] Automation Research Center, Northeastern University, Shenyang 110004, China
tychai@mail.neu.edu.cn
[2] Information Engineering School, Shenyang Institute of Chemical Technology 110142, China
zlj_lunlun@163.com

Abstract. Due to the lack of reliable on-line sensors to measure water quality parameters, it is difficult to control and operational optimization in the wastewater treatment plants (WWTPs). A hybrid time-delay neural network (TDNN) modeling method with data pretreatment is applied for BOD forecasting model in wastewater treatment. PCA is combined with robust expectation-maximization (EM), which reduces the influence of noise, outliers and missing data. The principal components are used as inputs to time-delay neural networks to predict the effluent BOD value. The simulation results using real process data show an enhancement in speed and accuracy, compared with a back propagation neural networks.

1 Introduction

Due to biological characteristics of the activated sludge process, it is difficult to measure parameters using on-line sensors. For example, the biochemical oxygen demand (BOD_5 or simply BOD) cannot be determined by laboratory tests until 5 days after the sample. Therefore, developing an effective water quality model to predict the treatment results has become a very important task in the optimization of the activated sludge process operation.

In recent years, many industrial applications of process modeling by neural networks in wastewater treatment have been reported [4], [1], [2]. The prediction capacity of neural network model strongly depends on the quality of the training data. It yields the very unreliable result when training data contains some outlier and missing data. In this paper, a robust time-delay neural network modeling method is proposed for predicting the treatment results. As the first step, data analysis performed on a robust EMPCA model is used to eliminate gross error, estimate missing data and reduce the input dimensions of the prediction model. Then a MLP time-delay neural networks model is further used to predict the performance of the treatment results. Subsequently, an on-line prediction and model-updating strategy is proposed and implemented.

* This paper is supported by the Chinese National Hi-Tech Development Program (2004AA412030), and the National Natural Science Foundation of China under Grant 60074019.

2 A WWTP Case Study

Fig. 1 shows a schematic diagram of an activated sludge process at Shenyang, China. The plant provides primary and secondary treatment. The following variables were measured at the points from (a) to (c) every 4 hours (see Table 1). The average hydraulic retention times for the primary settling tank, the aeration tank, and the final settling tank were calculated once 4 hours. Effluent BOD measured at point c depends on the other parameters at point (a) and point (b) measured several hours early.

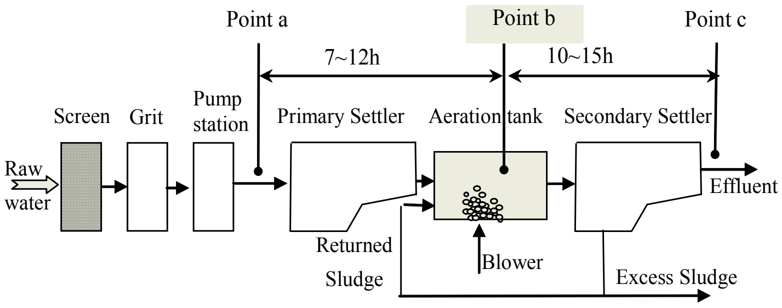


Fig. 1. Schematic of the municipal WWTPs.

Table 1. Selected variables characterizing the behaviors of the studied WWTP.

Sample points	Online data	Laboratory test data
Influent (a)	Flow rate (Q_{in})	COD, SS
Bioreactor (b)	Aeration air flow rate (Q_a) Returned sludge flow rate (Q_r)	
Effluent (c)	Water temperature (T) pH	COD,BOD_5

3 Robust BOD Model Using Hybrid Time-Delay Neural Networks

In practice, data from industrial WWTPs often contains some outliers and is inherently incomplete, whose points are partly missing. Poor-quality data has become a serous problem for the neural network model. Recent studies indicate that consideration of statistical principles in the ANN model building process may improve modeling performance (Maier and Dandy, 2000). Expectation-maximization and principal component analysis are combined as a preprocessing stage. The purpose of this project is to combining an improved PCA with outliers' rejection, the missing estimation in high dimensional data set with time-delay artificial neural networks to improve prediction accuracy and speed of the effluent BOD, as shown in Fig. 2.

3.1 Robust EMPCA for Missing Data with Outliers

EM algorithm is an iterative computational method to get a maximum-likelihood estimate when the data can be conveniently viewed as incomplete. EM algorithm consists of two steps: expectation step (E step) and Maximization (M step). The E-step which involves filling in missing value is based on the expected value of the data. The M-step is maximized log-likelihood function to give revised parameter estimates based on sufficient statistics calculated in E-step.

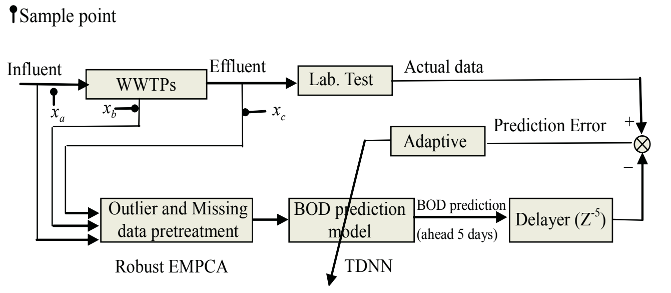


Fig. 2. Structure of the effluent BOD prediction system.

PCA is well-established technique for dimensionality reduction in data analyses, whose attractive features are dimensionality reduction, noise removal and orthogonal transformation. Traditional PCA is based on the assumption that data have not incomplete data set. To solve the drawbacks of traditional PCA, Sam Roweis presented an expectation-maximization (EM) algorithm for principal component analysis (EMPCA). EMPCA algorithm is summarized by:

$$\text{e-step: } X = (C^T C)^{-1} C^T Y \tag{1}$$

$$\text{m-step: } C = YX^T (XX^T)^{-1} \tag{2}$$

For the missing points for EM, the generalized e-step minimizes the squared distance ($\left\| Cx^* - y^* \right\|$) between the points and its reconstruction. The traditional PCA constructs the rank k subspace approximation to zero-mean training data that is optimal in a least-squares sense. The main disadvantage of least squares is its sensitivity to outliers. Outliers have a large influence on the regression because squaring the residuals magnifies the effects of these extreme data points. To minimize the influence of outliers, robust least squares was defined as:

$$\min_{x^*, y^*} \left\| W(Cx^* - y^*) \right\| \tag{3}$$

Robust least square minimizes a weighted sum of squares, where the weight W given to each data point depends on how far the point is from the fitted line. Points near the line get full weight. Points farther from the line get reduced weight. Points that are farther from the line than would be expected by random chance get zero weight. Refer to [7] for a detailed description of weight W. Robust EMPCA is presented by :

$$\text{e-step: } X = (C^T WC)^{-1} C^T WY \tag{4}$$

$$\text{m-step: } C = (YWX^T)(YWX^T)^{-1} \tag{5}$$

3.2 Time-Delay Neural Network with Data-Processing

A time-delay neural network (TDNN) model for predicting the effluent BOD in the activated sludge process is proposed and implemented. Fig. 3 gives the structure of the TDNN model. The inputs of the model are divided into three groups according to the inputs measurement locations. Let $x_a(k)$, $x_b(k)$ and $x_c(k)$ be the input vectors at point a, point b and point c, and $y_c(k)$ be the output scalar at point (c).

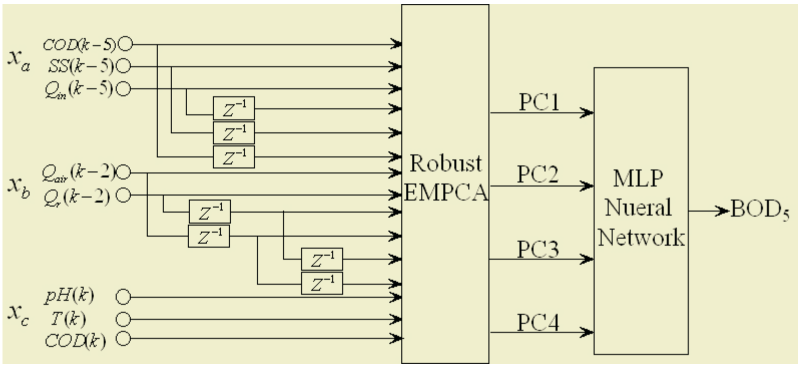


Fig. 3. Architecture of the TDNN forecasting the effluent BOD.

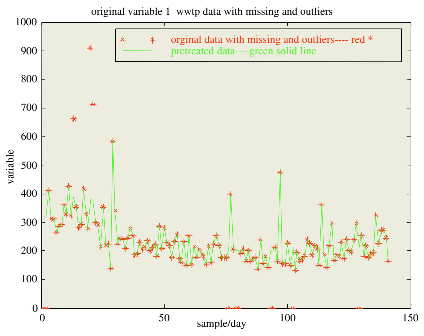
For the activated sludge process, the retention time between points a, b and c changes with the flow rate and other factors, which affect the static input-output mapping relationship. One of the most popular techniques for dealing with the modeling of dynamic systems is time-delay neural network (TDNN). It is a multilayer feed forward networks whose hidden neurons and output neurons are replicated across time. TDNN topology is similar to that of a standard MLP network.

All input variables are converted to a few principal components by robust EM and PCA. The PCs are used as inputs to MLP neural networks to predict the effluent BOD value. Considering a 7- to 12 hour process delay from point (a) to point (b), and 10- to 15-hour delay from point (b) to point (c), the TDNN model is expressed as:

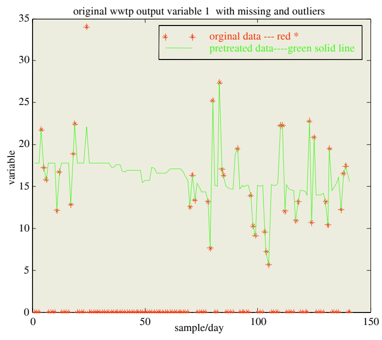
$$y_c(k) = f(x_c(k), x_b(k-2), x_b(k-3), x_b(k-4), x_a(k-5), x_a(k-6)) \qquad (6)$$

4 Result and Discussion

Estimation of Effluent BOD was carried out using the 141 samples. The training data pretreatment was firstly done using an expectation-maximization principal component analysis (EMPCA) using an iterative recursive robust least square (IRRLS) algorithms to remove the outlier and estimate the missing data. Comparisons between some of the original and rectified data were shown in fig. 4. Red * marks are original training data set with the serious missing data and outliers, and the green line is pretreated data curve. Results of data rectification show robust EMPCA can reject outliers and estimate the missing data.

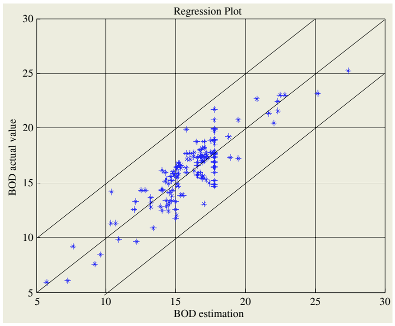


(a)

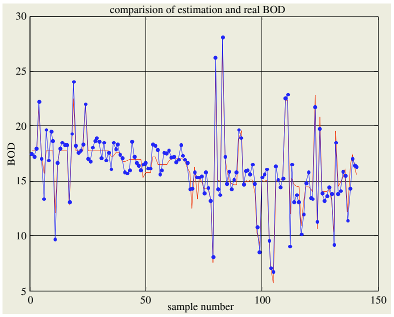


(b)

Fig. 4. Comparison between the actual and rectified training data (a) input variable COD (b) output variable BOD.



(a)



(b)

Fig. 5. Comparisons between estimated and real BOD (a) Relation between estimated vs. real BOD using TDNN without PCA;(b) Estimated BOD curve using TDNN with EMPCA.

To improve the BOD prediction, a model reduction technique based on PCA was used. The model dimension has been reduced from the original 15 variable to 4 principal components, and about 85% information has been described. Fig. 5 (a) shows the prediction results without EMPCA , and (b) the prediction results using EMPCA. The training and the testing of the TDNN with PCA took significantly less time. The Linear correlation between the real BOD and estimation without PCA is $y = 0.9644x + 0.6341$. The prediction results using TDNN with EMPCA were improved, as shown in Fig. 5 (b), whose linear correlation $y = 0.9809x + 0.3490$. Blue line is actual value, and red line BOD estimation.

5 Conclusions

This paper deals with the development of effluent BOD software sensor techniques. We focus our attention on the preprocessing of noise, outliers and missing data inherent in measured values. To minimize the influence of outliers, robust EMPCA is used. A hybrid time-delay neural network (TDNN) used expectation-maximization principal component analysis as a preprocessing stage. Simulation results show that EMPCA can reduce the dimension of input variables, estimate the missing data and remove the outliers. The hybrid TDNN can enhance prediction capability and reduces the overffitting problem of neural networks.

References

1. Zhu, J., Zurcher, J., Rao, M., Meng, M.Q.H.: An Online Wastewater Quality Predication System Based on a Time-delay Neural Network. Engineering Applications of Artificial Intelligence, **11** (1998) 747-758
2. Belanche, L.A., Valdé, J.J., Comas, J., Roda, R.I., Poch, M.: Towards a Model of Input-output Behaviour of Wastewater Treatment Plants Using Soft Computing Techniques. Environmental modeling & software, **14** (1999) 409-419
3. Lee, D., Jeon, C.O., Park, J.M., Chang, K.S.: Hybrid Neural Network Modeling of a Full-scale Industrial Wastewater Treatment Process. Biotechnology and Bioengineering, **78** (2002) 670-682
4. Tomida, S., Hanai, T., Honda, H., Kobayashi, T.: Construction of COD Simulation Model for Activated Sludge Process by Recursive Fuzzy Neural Network. Journal of Chemical Engineering of Japan, **34** (2001) 369-375
5. Doymaz, F., Bakhtazad, A., Romagnoli, J.A., Palazoglu, A.: Wavelet-based Robust Filtering of Process Data. Computers and Chemical Engineering, **25** (2001) 1549-1559
6. Waibel, A., Hanazawam, T., Hinton, G., Shikano, K., Lang, K.J.: Phoneme Recognition Using Time-delay Neural Networks. IEEE Trans. Neural Network, **4** (1989) 86-95
7. DuMouchel, W., O'Brien, F.: Integrating a Robust Option into a Multiple Regression Computing Environment. Computing Science and Statistics. Proceedings of the 21st Symposium on the Interface, American Statistical Association, Alexandria (1989) 297-301

A Split-Step PSO Algorithm in Prediction of Water Quality Pollution

Kwokwing Chau

Department of Civil and Structural Engineering, Hong Kong Polytechnic University, Hunghom, Kowloon, Hong Kong, China
cekwchau@polyu.edu.hk

Abstract. In order to allow the key stakeholders to have more float time to take appropriate precautionary and preventive measures, an accurate prediction of water quality pollution is very significant. Since a variety of existing water quality models involve exogenous input and different assumptions, artificial neural networks have the potential to be a cost-effective solution. This paper presents the application of a split-step particle swarm optimization (PSO) model for training perceptrons to forecast real-time algal bloom dynamics in Tolo Harbour of Hong Kong. The advantages of global search capability of PSO algorithm in the first step and local fast convergence of Levenberg-Marquardt algorithm in the second step are combined together. The results demonstrate that, when compared with the benchmark backward propagation algorithm and the usual PSO algorithm, it attains a higher accuracy in a much shorter time.

1 Introduction

Over the past two decades, frequent algal blooms with occasional massive fish kills have been recorded in Tolo Harbour. It may largely be attributed to its intrinsic semi-enclosed nature and the extremely low tidal flushing rate. Moreover, most of the freshwater runoff in the catchment area is routed to reservoirs so that the river discharges to the harbour are much reduced. The condition is further deteriorated by the nutrient enrichment through municipal and livestock waste discharges in the harbour through the rapid economic development recently. Precise prediction of algal booms is beneficial to fisheries and environmental management since it allows the fish farmers to have more float time to take appropriate precautionary measures. However, the extremely complex dynamics of algal blooms are related to various pertinent physical and biochemical factors and are not well-comprehended.

Process-based mathematical models, such as finite element or finite difference methods, are conventionally used to forecast flow and water quality parameters in a water body. In general, they require exogenous input and embrace different assumptions. In numerical modeling, the physical problem is represented by a highly coupled, non-linear, partial differential equation set. The involving processes are highly complex and uncertain which may consume enormous computing cost and time. In this sense, mechanistic models are not totally satisfactory in representing the highly complex inter-relationships. Recently, soft computing (SC) techniques have been gradually becoming a trend to complement or replace the process-based models. The characteristics of these data-driven approaches include built-in dynamism, data-error tolerance, no need to have exogenous input and so on. Amongst others, artificial neural networks (ANN), in particular the feed forward back-propagation (BP) perceptrons, have been widely applied in water resources engineering [1]. However, slow training convergence speed and easy entrapment in a local minimum are inherent drawbacks of the commonly used BP algorithm [2]. Levenberg-Marquardt (LM) optimization technique [3] is a commonly used ANN that has attained certain improvements such as convergence rates over the BP algorithm. Swarm intelligence is another recent SC technique that is developing quickly [4]. This technique has been applied in hydrological problems and accomplished satisfactory results [5],[6].

In this paper, a split-step PSO algorithm is employed to train multi-layer perceptrons for algal bloom prediction in Tolo Harbour of Hong Kong with different lead times and input variables. It is believed that, by combining the two algorithms, the advantages of global search capability of PSO algorithm in the first step and local fast convergence of LM algorithm in the second step can be fully utilized to furnish promising results.

2 Characteristics of PSO Algorithm

When PSO algorithm is initially proposed, it is considered a tool for modeling social behavior and for optimization of difficult numerical solutions [4], [7]. This computational intelligence technique is intimately related to evolutionary algorithms and is an optimization paradigm that mimics the ability of human societies to process knowledge [8]. Its principle is based on the assumption that potential solutions will be flown through hyperspace with acceleration towards more optimum solutions. PSO is a populated search method for optimization of continuous nonlinear functions resembling the biological movement in a fish school or bird flock. Each particle adjusts its flying according to the flying experiences of both itself and its companions. During the process, the coordinates in hyperspace associated with its previous best fitness solution and the overall best value attained so far by other particles within the group are kept track and recorded in the memory.

Among other advantages, the more significant one is its relatively simple coding and hence low computational cost. One of the similarities between PSO and a genetic algorithm is the fitness concept and the random population initialization. However, the evolution of generations of a population of these individuals in such a system is by cooperation and competition among the individuals themselves. The population is responding to the quality factors of the previous best individual values and the previous best group values. The allocation of responses between the individual and group values ensures a diversity of response. The principle of stability is adhered to since the population changes its state if and only if the best group value changes. It is adaptive corresponding to the change of the best group value. The capability of stochastic PSO algorithm, in determining the global optimum with high probability and fast convergence rate, has been demonstrated in other cases [7], [8]. PSO can be readily adopted to train the multi-layer perceptrons as an optimization technique.

3 Training of Perceptrons by PSO

Without loss of generality, a three-layered preceptron is considered in the following. $W^{[1]}$ and $W^{[2]}$ represent the connection weight matrix between the input layer and the hidden layer, and that between the hidden layer and the output layer, respectively. During training of the preceptron, the i-th particle is denoted by $W_i = \{W^{[1]}, W^{[2]}\}$ whilst the velocity of particle i is denoted by V_i. The position representing the previous best fitness value of any particle is denoted by P_i whilst the best matrix among all the particles in the population is recorded as P_b. Let m and n represent the index of matrix row and column, respectively, the following equation represents the computation of the new velocity of the particle based on its previous velocity and the distances of its current position from the best experiences both in its own and as a group.

$$\begin{aligned} V_i^{[j]}(m,n) &= V_i^{[j]}(m,n) + r\alpha[P_i^{[j]}(m,n) - W_i^{[j]}(m,n)] \\ &\quad + s\beta[P_b^{[j]}(m,n) - W_i^{[j]}(m,n)] \end{aligned} \tag{1}$$

where j = 1, 2; m = 1, …, M_j; n= 1, …, N_j; M_j and N_j are the row and column sizes of the matrices W, P, and V; r and s are positive constants; α and β are random numbers in the range from 0 to 1. In the context of social behavior, the cognition part $r\alpha[P_i^{[j]}(m,n) - W_i^{[j]}(m,n)]$ denotes the private thinking of the particle itself whilst the social part $s\beta[P_b^{[j]}(m,n) - W_i^{[j]}(m,n)]$ represents the collaboration among the particles as a group. The new position is then determined based on the new velocity as follows:

$$W_i^{[j]} = W_i^{[j]} + V_i^{[j]} \tag{2}$$

The fitness of the i-th particle is determined in term of an output mean squared error of the neural networks as follows:

$$f(W_i) = \frac{1}{S}\sum_{k=1}^{S}\left[\sum_{l=1}^{O}\{t_{kl} - p_{kl}(W_i)\}^2\right] \tag{3}$$

where f is the fitness value, t_{kl} is the target output; p_{kl} is the predicted output based on W_i; S is the number of training set samples; and, O is the number of output neurons.

4 Split-Step PSO Algorithm

The combination of two different SC techniques could enhance the performance through fully utilization of the strengths of each technique. In this algorithm, the training process is divided into two stages. Initially the perceptron is trained with the PSO algorithm for a predetermined generation number to exploit the global search ability for near-optimal weight matrix. Then, after this stage, the perceptron is trained with the LM algorithm to fine tune the fast local search. This might be able to avoid the drawback of either entrapment in local minima in LM algorithm or longer time consumption in global search of PSO algorithm.

5 Algal Bloom Prediction in Tolo Harbour

In order to test the capability of the model to mimic a particular case study with accurate depiction of real phenomena, it has been employed to predict the algal bloom

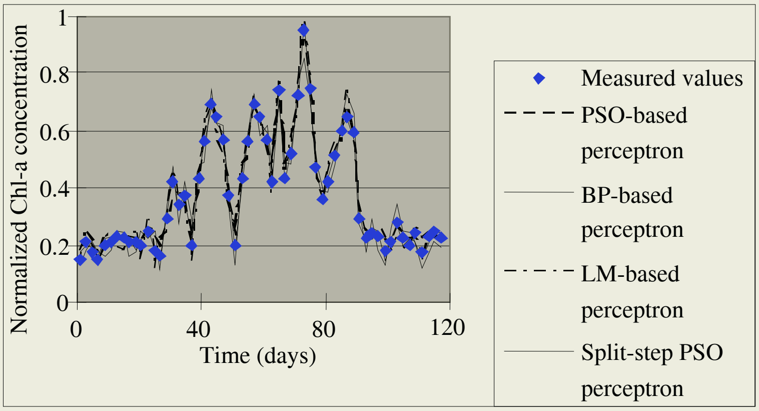


Fig. 1. 1 week lead time chlorophyll-a prediction for scenario 2 in the validation process.

dynamics in Tolo Harbour of Hong Kong [9], [10]. Observation data indicate the life cycle of algal blooms to be in the order of 1 to 2 weeks. As such, algal biomass, represented as chlorophyll-a, is forecasted with a lead time of 1 and 2 weeks based on a complete set of biweekly water quality data at Tolo Harbour from year 1982 to year 2002. The data of 1982-1995 and those of 1996-2002 are used for training and testing/validation, respectively. The division of data is tailored so as to include extreme frequency and intensity in both sets of data. Throughout the analysis, depth-averaged values from the surface, mean, and bottom of the water column are employed.

In this case, ten input variables, including the time-lagged chlorophyll-a, secchi disc depth, nitrogen, phosphorus, dissolved oxygen, rainfall, water temperature, solar radiation, wind speed and tidal range, are considered to be significant to the algal dynamics of Tolo Harbour [9]. Various perceptrons, having an input layer with one to ten neurons, a hidden layer with three to five neurons, and an output layer with one neuron, are tested. The single output node represents chlorophyll-a. Three scenarios are attempted with 10, 5 and 1 input variables for scenario 1, 2, and 3, respectively. Other major PSO parameters adopted are as follows: number of population is 30; the maximum and minimum velocity values are 0.3 and -0.3, respectively. All source data are normalized into the range between 0 and 1, by using the maximum and minimum values of the variable over the whole data sets.

6 Analysis and Discussions

The performance of the split-step multi-layer ANN is evaluated in comparison with the benchmarking standard BP-based network, a PSO-based network and a LM network. In order to provide a fair and common initial ground for comparison purpose, the training process of the BP-based perceptron or LM network commences from the best initial population of the corresponding PSO-based perceptron or split-step network. Figure 1 shows the 1 week lead time normalized chlorophyll-a prediction for scenario 3 by all perceptrons in the validation process. Table 1 shows comparison of the results for chlorophyll-a forecasting with both 1 week and 2 weeks lead times for

Table 1. Results for chlorophyll-a forecasting based on scenarios 1 to 3.

Input data	Algorithm	Coefficient of correlation			
		Training		Validation	
		1 week ahead	2 weeks ahead	1 week ahead	2 weeks ahead
Scenario 1	BP-based	0.984	0.953	0.968	0.942
	PSO-based	0.989	0.980	0.973	0.962
	LM	0.986	0.958	0.970	0.958
	Split-step	0.991	0.983	0.975	0.969
Scenario 2	BP-based	0.974	0.934	0.956	0.934
	PSO-based	0.984	0.976	0.959	0.954
	LM	0.977	0.942	0.957	0.945
	Split-step	0.988	0.981	0.968	0.963
Scenario 3	BP-based	0.964	0.935	0.946	0.923
	PSO-based	0.983	0.973	0.965	0.952
	LM	0.969	0.938	0.951	0.928
	Split-step	0.985	0.978	0.969	0.958

Table 2. Steady-state fitness evaluation times during training for various algorithms.

Algorithm	Steady-state fitness valuation time
BP-based	21,000
PSO-based	9,000
LM	5,000
Split-step	6,000

scenarios 1 to 3. It should be noted that the results do not exhibit a significant advantage of using more environmental variables as the network inputs and that 1 week lead time is better than its counterparts of 2 weeks. It can be observed that the split-step algorithm performs the best in terms of prediction accuracy. Table 2 shows the steady-state fitness evaluation times during training for various perceptrons. It can be observed that the split-step perceptron, with rate comparable to that of LM algorithm, exhibits much faster convergence than those by the BP-based perceptron and the PSO-based network.

7 Conclusions

In this paper, a perceptron based on a split-step PSO algorithm is employed for real-time prediction of algal blooms at Tolo Harbour in Hong Kong with different lead times and input variables. The results do not exhibit any advantage of using more environmental variables as the network inputs. The chlorophyll-a output from the 1 week time-lagged chlorophyll-a input is shown to be a robust forewarning and decision-support tool. The results also show that the split-step PSO-based perceptron outperforms the other commonly used optimization techniques in algal bloom prediction, in terms of both convergence and accuracy.

Acknowledgements

This research was supported by the Research Grants Council of Hong Kong (PolyU5132/04E).

References

1. Chau, K.W., Cheng, C.T.: Real-time Prediction of Water Stage with Artificial Neural Network Approach. Lecture Notes in Artificial Intelligence, **2557** (2002) 715-715
2. Rumelhart, D.E., Widrow, B., Lehr, M.A.: The Basic Ideas in Neural Networks. Communications of the ACM , **37** (1994) 87-92
3. Hagan, M.T., Menhaj, M.B.: Training Feedforward Networks with the Marquardt Algorithm. IEEE Transactions on Neural Networks, **5** (1994) 989-993
4. Kennedy, J., Eberhart, R.: Particle Swarm Optimization. Proceedings of the 1995 IEEE International Conference on Neural Networks. Perth (1995) 1942-1948
5. Chau, K.W.: River Stage Forecasting with Particle Swarm Optimization. Lecture Notes in Computer Science, **3029** (2004) 1166-1173
6. Chau, K.W.: Rainfall-Runoff Correlation with Particle Swarm Optimization Algorithm. Lecture Notes in Computer Science, **3174** (2004) 970-975
7. Kennedy, J.: The Particle Swarm: Social Adaptation of Knowledge. Proceedings of the 1997 International Conference on Evolutionary Computation. Indianapolis (1997) 303-308
8. Clerc, M., Kennedy, J.: The Particle Swarm – Explosion, Stability, and Convergence in a Multidimensional Complex Space. IEEE Transactions on Evolutionary Computation, **6** (2002) 58-73
9. Chau, K.W., Jin, H.S.: Eutrophication Model for a Coastal Bay in Hong Kong. Journal of Environmental Engineering ASCE, **124** (1998) 628-638
10. Chau, K.W., Jin, H.S., Sin, Y.S.: A Finite Difference Model of Two-Dimensional Tidal Flow in Tolo Harbor, Hong Kong. Applied Mathematical Modelling, **20** (1996) 321-328

Long-Term Prediction of Discharges in Manwan Reservoir Using Artificial Neural Network Models

Chuntian Cheng[1], Kwokwing Chau[2], Yingguang Sun[1], and Jianyi Lin[1]

[1] Institute of Hydroinformatics, Department of Civil Engineering,
Dalian University of Technology, Dalian, Liaoning 116024, China
ctcheng@dlut.edu.cn

[2] Department of Civil and Structural Engineering, Hong Kong Polytechnic University,
Hunghom, Kowloon, Hong Kong, China

Abstract. Several artificial neural network (ANN) models with a feed-forward, back-propagation network structure and various training algorithms, are developed to forecast daily and monthly river flow discharges in Manwan Reservoir. In order to test the applicability of these models, they are compared with a conventional time series flow prediction model. Results indicate that the ANN models provide better accuracy in forecasting river flow than does the autoregression time series model. In particular, the scaled conjugate gradient algorithm furnishes the highest correlation coefficient and the smallest root mean square error. This ANN model is finally employed in the advanced water resource project of Yunnan Power Group.

1 Introduction

Mathematical models are often employed to forecast the future trend of flow discharges in reservoirs. The long-term prediction results can be widely used in areas such as environmental protection, flood prevention, drought protection, reservoir control, and water resource distribution. This may have significant economic value in decision control of reservoirs and hydropower stations. Conventionally, factor analysis and hydrological analysis methods such as historical evolution method, time series analysis, multiple linear regression method and so forth, are used to forecast the long-term discharges. Nowadays, time series analysis and multiple linear regression method are the two most commonly used methods. The time series analysis is based on the decomposition of various factors into trend and cycle. After 1970s, autoregressive moving-average models proposed by Box et al. [1] are also widely used. Since 1990s, artificial neural network (ANN), based on the understanding of the brain and nervous systems, is gradually used in hydrological prediction. A comprehensive review of the application of ANN to hydrology can be found in ASCE Task Committee [2],[3]. In this paper, the current development on the application of ANN in long-term prediction of flow is presented. Its prediction effectiveness is evaluated for the prototype case study in Manwan hydropower station.

2 Three Layer Feed-Forward Back-Propagation Network Model

Figure 1 shows a typical three layer feed-forward back-propagation ANN. Among others, although the steepest descent method is the simplest, it has the drawbacks of slowest convergence and lack of effectiveness. In real ANN applications, the steepest descent method is seldom used. Haykin [4] discussed several data-driven optimization training algorithms such as Levenberg-Marquardt (LM) algorithm and scaled conjugate gradient (SCG) algorithm. In this paper, the SCG algorithm [5],[6] is employed and the procedure is as follows [7]:

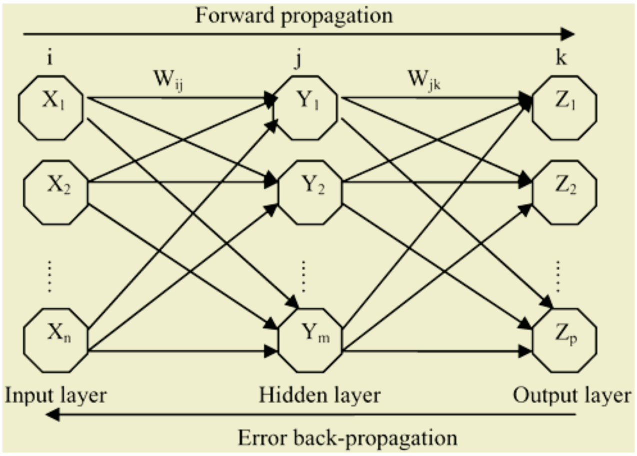


Fig. 1. Three layer feed-forward back-propagation neural network.

1. The weight matrix w is initialized, with range from -0.5 to 0.5.

$$\vec{d}_0 = -\vec{g}_0 \quad . \tag{1}$$

   where g_0 is the gradient of error function and d_0 is an initialized searching direction.

2. At the start of the generation number k, the learning rate α_k is determined from linear search via function $f(\vec{w}+\alpha_k\vec{d}_k)$ where $\vec{d}_k$ is the searching direction at the generation k. The weight matrix is adjusted through the following equation:

$$w_{k+1} = w_k + \alpha_k \vec{d}_k \quad . \tag{2}$$

3. If the error is smaller than the threshold value or the preset training generation is reached at the generation number k+1, the training process is terminated.

4. Otherwise, the new searching direction $\vec{d}_{k+1}$ is computed. If (k+1) is an integer multiple of dimension number of weight matrix w, then

$$\vec{d}_{k+1} = -\vec{g}_{k+1} \quad . \tag{3}$$

   Otherwise,

$$\vec{d}_{k+1} = -\vec{g}_{k+1} + \beta_k \vec{d}_k \quad . \tag{4}$$

$$\beta_k = \frac{(g_k g_k^T)}{(g_0 g_0^T)} \quad . \tag{5}$$

5. Go back to step 2.

3 Application of ANN to Flow Prediction in Manwan

A three layer feed-forward back-propagation ANN model is employed for flow prediction in Manwan. It has four input neurons (Q_t, Q_{t-1}, Q_{t-2} and Q_{t-3}), four hidden neurons, and one output neuron (Q_{t+1}). Huang and Foo [8] found that, among different training algorithms, the SCG algorithm converges faster and attains a higher accuracy. The real daily flow data from 2001 to 2003 and monthly flow data from 1953 to 2003 are studied. All data are normalized to range between -1 and 1 first.

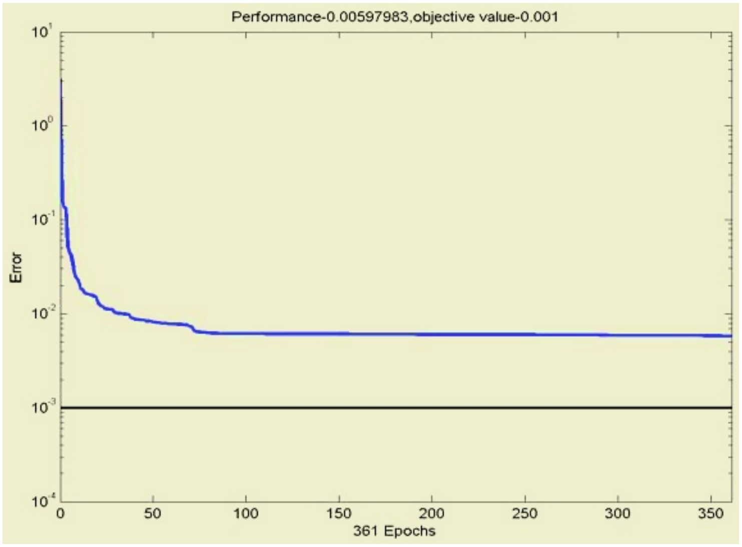


Fig. 2. Training results of daily flow data by ANN.

3.1 Prediction of Daily Flow and Monthly Flow

Daily data for the entire year 2001 and those from 2002 to 2003 are used to train and verify the ANN model, respectively. Figure 2 shows that, after training for 361 sample points, the minimum error is only 0.00598. Figure 3 shows the verification results which give a correlation coefficient of 0.97 between the predicted and actual values and a root mean square error (RMSE) of 0.0087. It is demonstrated that the prediction of daily flow results is highly satisfactory. Similarly, monthly data from 1953 to 1993 and those from 1994 to 2003 are used to train and verify the ANN model, respectively. The prediction of monthly flow results is also satisfactory.

3.2 Sensitivity Analysis of Various Training Algorithms

Although many algorithms are available in training ANN, each one has its own advantages and limitations. Here, the gradient descent, LM and SCG algorithms are compared. All three algorithms undergo the training of ANN under the same condi-

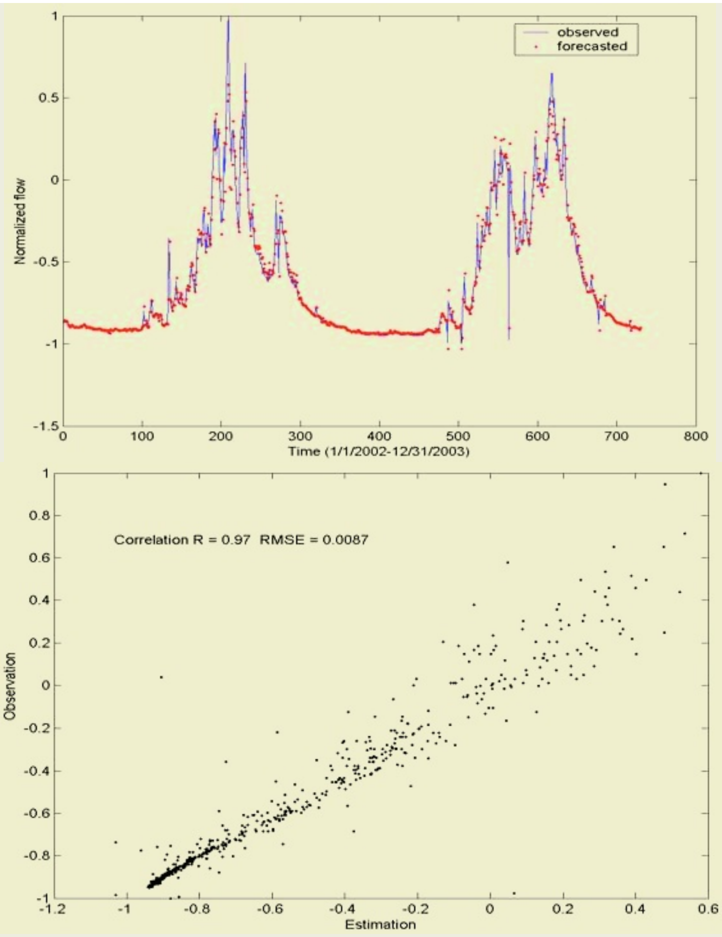


Fig. 3. Verification results of daily flow data by ANN.

tions. Table 1 shows the simulation results, which show that the gradient descent algorithm has the slowest convergence, smallest correlation coefficient and the largest RMSE. On the other hand, the SCG algorithm has the highest accuracy and the LM algorithm converges most quickly.

4 Result Comparison with Time Series Model

Auto-regression time series model is a conventional time series prediction model. Owing to the simplicity of both the model and its data requirements, it has been widely applied in flow prediction [9]. Hence, it is used as the yardstick to gauge the performance of the ANN model in this case. In order to have the same basis of comparison, the same training and verification sets are used for both models. It is demonstrated that, when employed for flow prediction in Manwan, ANN exhibits distinct advantages over conventional time series model. Table 2 shows the performance comparison between ANN model and time series model for prediction of monthly flow. The correlation coefficient of ANN model is 0.89, which is larger than its counterparts of time series model (0.84). Moreover, the RMSE of ANN model is 0.03, which is much smaller than that of time series model (0.108).

Table 1. Sensitivity analysis of various algorithms for monthly flow prediction in Manwan.

Training algorithm	Correlation coefficient	Normalized RMSE	Number of sampling points
Gradient descent	0.799	0.057	1000
LM	0.878	0.036	71
SCG	0.890	0.03	415

Table 2. Performance comarison between ANN model and time series model for prediction of monthly flow.

Correlation coefficient		Normalized RMSE	
ANN model	Time series model	ANN model	Time series model
0.89	0.84	0.03	0.108

5 Conclusions

In this paper, an ANN model is used to predict long-term flow discharges in Manwan based on historical records. Data from 2001 to 2003 and from 1953 to 2003 are used for daily and monthly flow predictions, respectively. The results indicate the ANN model can give good prediction performance. The correlation coefficients between the prediction values and the observational values are 0.97 and 0.89 for daily and monthly flow analysis, respectively. The sensitivity analysis of the training algorithms show that the SCG algorithm can enhance the accuracy of model prediction results effectively. It is found, through result comparison with a conventional time series model, that the ANN model is able to give more accurate prediction. This demonstrates its distinct capability and advantages in identifying hydrological time series comprising non-linear characteristics.

Acknowledgements

This research was supported by the National Natural Science Foundation of China (No. 50479055) and the Internal Competitive Research Grant of Hong Kong Polytechnic University (G-T592).

References

1. Box, G.E.P., Jenkins, G.M.: Time Series Analysis Forecasting and Control. Holden-Day, San Francisco (1976)
2. ASCE Task Committee. Artificial Neural Networks in Hydrology-I: Preliminary Concepts. Journal of Hydrologic Engineering, ASCE **5** (2000) 115-123

3. ASCE Task Committee: Artificial Neural Networks in Hydrology-II: Hydrological Applications. Journal of Hydrologic Engineering, ASCE **5** (2000) 124-137
4. Haykin, S.: Neural Networks, a Comprehensive Foundation. Prentice Hall, Upper Saddle River (1999)
5. Hagan, M.T., Demuth, H.B., Beale, M.: Neural Network Design. PWS Pub, Boston London (1996)
6. Fitch, J.P., Lehman, S.K., Dowla, F.U., Lu, S.K., Johansson, E.M., Goodman, D.M.: Ship Wake Detection Procedure Using Conjugate Gradient Trained Artificial Neural Network. IEEE Transactions on Geosciences and Remote Sensing, **9** (1991) 718-725
7. Moller, M.F.: A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning. Neural Networks, **6** (1993) 523-533
8. Huang, W., Foo, S.: Neural Network Modeling of Salinity in Apalachicola River. Water Resources Research, **31** (2002) 2517-2530
9. Wang, B.D.: Fuzzy Mathematical Methods for Long-Term Hydrological Prediction. Dalian University of Technology Press, Dalian, China

Application of Artificial Neural Networks to Predicate Shale Content

Kesheng Wang, Resko Barna, Yi Wang, Maxim Boldin, and Ove R. Hjelmervik

Knowledge Discovery Laboratory
Department of Production and Quality Engineering
N-7491 NTNU, Trondheim, Norway

Abstract. This paper describes an Artificial Neural Network approach to the predication problem of shale content in the reservoir. An interval of seismic data representing the zone of interest is extracted from a three-dimensional data volume. Seismic data and well log data are used as input and target to Regularity Back-propagation (RBP) neural network. A series of ANNs is trained and results are presented.

1 Introduction

Development and exploitation of a hydrocarbon reservoir typically makes use of maps describing spatial distributions of relevant parameters. These maps are generated from well information that may or may not adequately sample the reservoir. In addition, reservoir often exhibits a high degree of heterogeneity that introduces high levels of uncertainties into interpolated parameters. A recent approach has used complex trace attributes derived for 3-D seismic data volumes to track the desired parameters through heterogeneous zones. This is usually a time consuming process requiring a skilled interpreter.

Oil or gas reservoir, in large part, is characterized by lithological parameters, such as porosity, permeability and shale content. Accurate estimates of these parameters are important for calculating oil or gas reserves and developing exploitation strategies. The motivation for this study was to be able to produce accurate shale content predications.

Current, 3-D seismic surveying is one of the primary tools for characterizing a hydrocarbon reservoir. Attributes extracted from seismic data are used to qualitatively describe variations in lithology and associated physical parameters across zones of interest. By correlating seismic data with well log data, it is, in principle, possible to produce qualitative predications at each common midpoint (CMP) location.

Conventional techniques correlate seismic attributes to lothologic parameters using determinist and stochastic methods. The use of an ANN approach enables seismic data to be related to shale content without explicitly defining the relationships for the various parameters – i.e. without previous knowledge in order to establish a mathematical model. More important, ANNs naturally utilize intervals of seismic data or combinations of attributes and other information rather than single values. This ability increases the amount of available information from which to make predications.

This report describes a set of procedures that were used to produce shale content predication for an oil field in the Harmmerfest basin located in the Norwegian Sea of the coast of Norway. The primary focus is how shale content predications depend on attributes from seismic trace and well log. One well log attribute related to shale content, VSH was selected as the target of the ANN and combinations of several slops of seismic amplitude samples and time index was selected as input to the ANN.

Since the important issue is to define shale heterogeneities in the sand-dominated reservoir, the VSH response is used as the indicator for reservoir characteristics. Consequently, the idea is to predict the VSH response from seismic attributes. This will enable generation of VSH from seismic trace. Predicting the shale content is to reduce the risk of drilling by differentiating between sand layers and shale layers. The well log VSH varies between 1 and 0 and indicates the shale content. A VSH value of 1 indicates pure shale, while 0 indicates pure sand.

Well data is a key to establishing the relationship between shale content and seismic attributes. Before such a relationship can be established, the VSH log which was recorded into function of time has to be up-scaled and transformed into a function of time. The up-scaling was conducted with an averaging process.

2 Architecture of the Applied ANN

2.1 Back-Propagation ANN

The Artificial Neural Networks technique employed was a Back-propagation net of Multilayer Perceptron (MLP) type.

The representative data set consisted of seismic traces extracted at and round the wells. (In this testing, we simply chose a seismic trace at the well.) The VSH traces are recorded in this specified well. In particular, the nearest nine traces should be selected around each well. Consequently, at each well, nine traces were presented to the ANN to relate to the desired output (VSH log).

The variations available for constructing an ANN are considerable. The number of input layers as well as the number of hidden layers can be modified to improve the performance of the ANN. Furthermore, the number of nodes in each hidden layer can be varied and some parameters in ANN, such as learning rate and momentum, and the activation functions can be changed.

2.2 Regularized Back-Propagation Network

In the testing, a regularized Back-propagation network proposed by Saggaf et al. [2003] was selected. A traditional BP ANN is constructed by solving a system of equations such that the network weights minimize the misfit error between the training data and the network output. The Objective Function (*OF*) of the optimization is thus the mean square error of the misfit:

$$OF = \frac{1}{N}\sum_{i=1}^{N} e_i^2$$

where e_i is the misfit error for the ith training observation and N is the total number of observations in the training data set. The non-monotonous generalization behavior of the network can be alleviated by modifying the objective function such that not only is the misfit error reduced, but also the network weight as well:

$$OFR = (1-\lambda)\frac{1}{N}\sum_{i=1}^{N} e_i^2 + \lambda\frac{1}{M}\sum_{i=1}^{M} w_i^2$$

where w_i is the ith network weight, M is the total number of weights in the ANN, and λ is the regularization parameter.

This modified objective function results in a network that not only fits the data, but also has small weights that get rise to smaller variations in the output and thus yield a smoother result in output space that is less likely to over-fit the data.

The regularization parameter λ determine the relative emphasis of smoothness versus degree of fit to the training data. A small value of λ results in an un-regularized network. On the other hand, if λ is too large, then the network would not adequately fit data.

3 Input and Output Slops

The input of the ANN was three subsequent seismic amplitudes and their sliding window index. (Note that the sliding window index is not the same as depth, since the time intervals for each well were adjusted to the actual depth of the horizon.) The output of the ANN was the VSH well log value that corresponded to the seismic amplitude in the middle of the 3 inputs. The organization of input and output slops is shown in Figure 1.

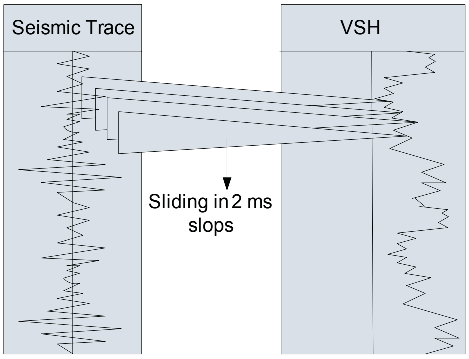


Fig. 1. Principle of sliding-window technique. The window relates several seismicsamples simultaneously to a certain sand property (e.g. VSH).

4 Data Description and Preprocessing

4.1 Seismic Data

The raw seismic data amplitudes were the final processed data samples of the reservoir interval. The instantaneous amplitude attribute was calculated from the raw amplitudes.

4.2 Well Log Data

The logs were converted from depth to time and sampled to the same sampling interval of the seismic data.

In the tests three well logs were used, two of which served to provide the training data and one was used to provide measured well log data with which the prediction was compared.

A short time period was chosen to avoid the uniqueness problem. A three-sample-wide window was sliding within the black rectangle shown on the Figure 2.

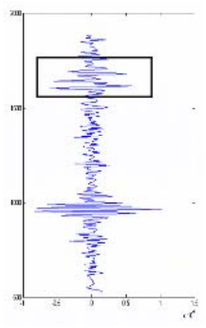


Fig. 2. Seismic trace and selected short time period.

In the case of the three wells used in the experiment, the depth of the horizon to which the black rectangle window was attached is different. The difference can be seen on the Figure 3.

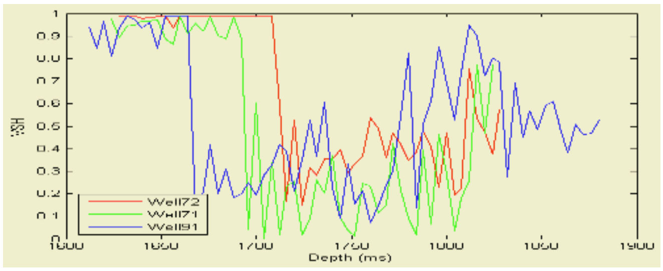


Fig. 3. The different horizon of well 72, 71 and 91.

5 Results

Training network with such data yielded acceptable results, as shown in the Figure 4. In the below experiments the time interval spanned 160ms. In another experiment we tried to expand the interval to see if the results would become worse, due to the probable occurrence of the uniqueness problem. The results are shown below.

It is visible on a lower right image that the prediction has become much less precise, though still follows the tendency of the real shale content.

6 Conclusions

This paper shows that a neural networks approach can be a valuable technique in order to predicate shale content in a short interval in the zone of interest. Compared to conventional methods, artificial neural networks have several advantages while many of their limitations are the same as that of other mathematical methods. The inclusion of time index (depth index) in the input data is shown to be essential for the ANN to produce realistic predication. Testing results show that misfit errors are acceptable.

In order to use ANN with success in seismic analysis, it is crucial to go through these steps:

- Exact definition of the problem to solve
- Understand the dependencies between parameters
- Design the training data (source, guess, etc.)

In this study, we only use a small part of data for training and validation (3 well logs and 3 seismic trace). For further research, we need to extend to the whole field. It will help to make more accurate predication. In principle, the method used in this study has a generalization to all attributes, especially to porosity and permeability.

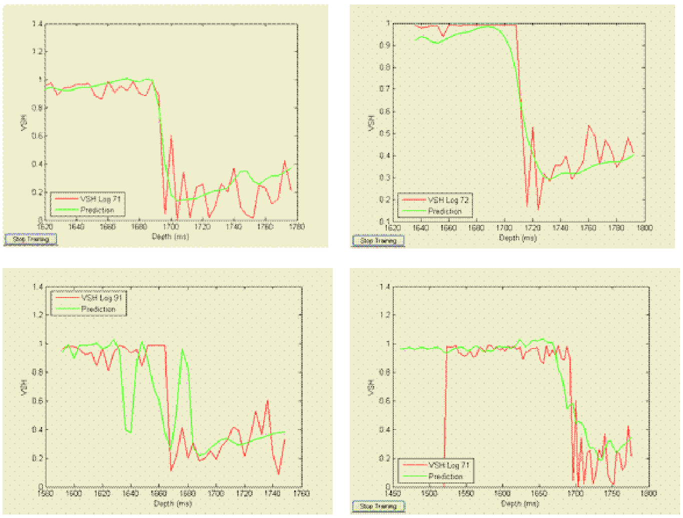


Fig. 4. Predication result for well 71, well 72 and well 91.

Acknowledgements

The authors would like to acknowledge Arild Buland and Britta Paasch form Statoil AS for valuable support in this research project and thanks for the financial support from the Research Council of Norway, Statoil AS and the National Natural Science Foundation of China under grant No.50375090.

References

1. Lindsay, R., Bocanegra, D.: Sand Thickness Prediction from Band-Limited Seismic Attributes Using Neural Networks: Oriente Basin, Ecuador, 72nd Ann. Internat Mtg: Soc. of Expl. Geophys (2002) 2451-2454

2. Liu, Z., Castagna, J., Zhang, X., Li, Y.: Rock Parameter Modeling Using Well and Prestack Depth Migration Data By Neural Networks. 72nd Ann. Internat. Mtg: Soc. of Expl. Geophys (2002) 967-970.
3. McCormack, M. D.: Neural Computing in Geophysics. The Leading Edge, **10** (1991) 11-15.
4. Nikravesh, M. Aminzadeh, F., and Zadeh, L. A.: Soft Computing and Intelligent Data Analysis. Elsevier (2003)
5. Poulton, M.: Neural Networks as an Intelligence Amplification Tool: A review of Applications: GEOPHYSICS, Soc. of Expl. Geophys., **67** (2002) 979-993
6. Russell, B., Hampson, D. and Lines, L.: Application of the Radial Basis Function Neural Network to the Prediction of Log Properties from Seismic Attributes - A Channel Sand Case Study. 73rd Ann. Internat. Mtg.: Soc. of Expl. Geophys (2003) 454-457
7. Russell, B., Ross, C. P. and Lines, L.: Neural Networks and AVO: The Leading Edge, **21** (2002) 268-277
8. Saggaf, M. M., Toksoz, M. N., and Marhoon, M. I.: Estimation of Reservoir Properties from Seismic Data by Smooth Neural Networks. Geophysics, **68** (2003) 1969-1983
9. Saggaf, M. M., Toksoz, M. N., and Marhoon, M. I.: Seismic Facies Classification and Identification by Competitive Neural Networks. **68** (2003) 1984-1999
10. Sandheim, W., Legget, L., and Aminzadeh, F.: Applications of Artificial Neural Networks and Fuzzy Logic. Kluwer Academic Publisher (2003)
11. Tonn, R.: Neural Network Seismic Reservoir Characterization in a Heavy Oil Reservoir: The Leading Edge. **21** (2002) 309-312
12. Walls, J.D., Taner, M. T., Taylor, G., et al: Seismic Reservoir Characterization of a U.S. Midcontinent Fluvial System Using Rock Physics, Poststack Seismic Attributes, and Neural Networks: The Leading Edge, **21** (2002) 428-436.
13. Wong, P. M., Aminzadeh, F., and Nikravesh, M., 2002, Soft computing for reservoir characterization and modeling, studies in Fuzziness and soft computing, ed. Physca-Verlag, Springer-Verlag.

Optimization of Forecasting Supply Chain Management Sustainable Collaboration Using Hybrid Artificial Neural Network

Sehun Lim[1] and Juhee Hahn[2]

[1] Department of Information System, Chung-Ang University
40-1, Naeri, Deaduck-Myun, Ansung City, Kyunggi-Do 456-050, South Korea
slimit@hanmail.net

[2] Department of Business, Chung-Ang University
40-1, Naeri, Deaduck-Myun, Ansung City, Kyunggi-Do 456-050, South Korea
jhan02@hanmail.net

Abstract. Artificial Neural Network (ANN) is widely used in business to optimize forecasting. Various techniques have been developed to improve outcomes such as adding more diverse algorithms, feature selection and feature weighting in input variables, and modification of input case using instance selection. In this research, ANN is applied to solve problems in forecasting a Supply Chain Management (SCM) sustainable collaboration. This research compares the performance of forecasting SCM sustainable collaboration with four types of ANN models: COANN (COnventional ANN), FWANN (ANN with Feature Weighting), FSANN (ANN with Feature Selection), and HYANN (HYbrid ANN with Feature Weighting and Feature Selection). Using HYANN to forecast an SCM sustainable collaboration gave the best results.

1 Introduction

Artificial Neural Network (ANN) is an excellent method for forecasting in business management. For example, ANN is widely used in the bank industry to forecast bankruptcy or insolvent operation. It is also used for forecasting of withdrawing customers and customer service management in credit card companies.

Recently, in order to forecast problems more accurately in cooperation management, many varieties of technique have been developed to improve outcomes such as adding more diverse algorithms, feature selection and feature weighting in input variables, and modification of input case using instance selection.

In this research, ANN is applied to solve problems in forecasting a Supply Chain Management (SCM) sustainable collaboration. A sustainable SCM is achieved until an SCM result is shown in the current cooperation or a substantial result has been achieved hereafter. Therefore, the sustainable collaboration of SCM implies its success. However, there are no guidelines to determine the need for SCM sustainable collaboration, nor are there any models for forecasting an SCM.

Therefore, this research has developed a model for forecasting SCM sustainable collaboration by adjusting the balance of measurements in the framework which was formed by Brewer, Speh [1], based on the Balanced ScoreCard (BSC) presented by Kaplan and Norton [10]. In developing the most accurate forecasting model for a sustainable collaboration of SCM, four sample models of forecasting performances were compared.

The result of this research demonstrates a more accurate forecasting model for the management of a cooperation. In addition, guidelines are developed that will assist in the use of a sustainable collaboration SCM.

2 Research Background

2.1 ANN as a Optimization Tool for Forecasting

ANN lacks any systematical method to determine the target output with input values or vector figures [3], [8], [9]. Despite this minor limitation, ANN retains superior performance over logit analysis or multivariable discriminant analysis (MDA). Therefore, it is used in forecasting stock price, bankruptcy, and customers churning, pattern recognition, and classification [7], [12], [13], [14].

There is a long history of research using ANN. Fletcher and Coss [2], who used ANN in forecasting a bankruptcy of a cooperation, proved that ANN has better performance in forecasting than logit analysis does. Tam and Kiang [12] also verified the superior forecasting of ANN in a bank bankruptcy. Recently, many researches are presently using ANN to improve the performance of forecasting with fuzzy membership function, genetic algorithm, and case-based reasoning. For example, Yang et al. [14] has proven that PNN (probabilistic neural networks) using neural network in forecasting a bankruptcy is superior to existing simple back propagation neural networks or MDA.

2.2 Feature Weighting and Feature Selection

Feature weighting and feature selection techniques are available to improve the performance of forecasting in data mining. Feature weighting puts more weight on feature variables, while feature selection is a technique to determine the most sufficient feature variable in data mining. Both methods have been proven to be effective in forecasting performance.

There are various methods applicable in feature weighting, including distance matrix by mechanical drills, weighting using genetic algorithm, and weighting of existing data values based on calculated relative importance used by AHP (Analytical Hierarchy Process) [11]. Feature selection also has many methods including selecting important feature variables such as genetic algorithm, selecting feature using entropy [5], and using statistical analysis methods such as t-test, chi-square test, logit, and MDA etc. Kai et al. [6] has demonstrated remarkable improvements in forecasting the performance of a recommended system based on collaboration with feature weighting and instance selection methods. Ahn et al. [5] also verified the innovative progress in predicting stock price using a genetic algorithm and feature selection with case selection through case based reasoning.

3 Methodology

3.1 Research Variable

In this research, the independent variables were established as learning perspective, internal process perspective, customer perspective, and financial perspective based on

BSC [1]. Survey questionnaires to measure the independent variable were modified appropriately to fit in distributing and manufacturing companies. Each question was given values over a 7-point Likert scale. For operating purposes, the dependent variable was measured as '1' in the case of sustainable collaboration, and otherwise as '0'.

Table 1. BSC Description Statistics.

	Feature Name	Min	Max	Mean	SD
Learning Perspective	Product and process innovation	1	6	4.361111	1.179722
	partnership management	2	6	4.333333	1.059007
	Information flows	2	6	4.231481	1.115787
	Threats and substitutes	1	7	4.425926	1.161724
Internal Process Perspective	Waste reduction	2	6	4.277778	1.092312
	Time compression	2	6	4.314815	1.116136
	Flexible response	2	7	4.305556	1.147597
	Unit cost reduction	2	7	4.342593	1.153313
Customer Perspective	View of product/service	2	7	4.314815	1.132759
	View of timeliness	2	7	4.259259	1.113808
	View of flexibility	2	7	4.351852	1.087867
	View of customer value	2	6	4.166667	1.045685
Financial Perspective	Profit margins	2	6	4.416667	1.086235
	Cash flow	3	6	4.268519	1.046637
	Revenue growth	2	6	4.240741	1.012726
	Return on assets	2	6	4.203704	1.039044

This survey was administered from June through September, 2003, to SCM specialists of distributing and manufacturing companies that are carrying out SCM. Out of 300 surveys that were distributed, 120 were collected and 108 were used for analysis after discarding surveys with incomplete answers. The statistical values of the variables are shown in <Table 1>.

3.2 Research Method

This research compared the performance of forecasting SCM sustainable collaboration through four types of ANN models: a traditional ANN model called COANN (COnventional ANN), FWANN (ANN with Feature Weighting) based on feature weighting, FSANN (ANN with Feature Selection) which reflects feature selection, and HYANN (HYbrid ANN with Feature Weighting and Feature Selection) which approaches successively based on obtaining information theory by feature selection after carrying out feature weighting

When COANN was carried out, feature selection or feature weighting was not used. Generally, the factors which affect the performance of forecasting in ANN are hidden layer value, learning rate, momentum etc. According to Hornik [4], simply controlling the hidden layer values can obtain a satisfactory outcome in classification problems.

However, as it is only the early stage of SCM use in the domestic market, it was impossible to gather a sufficient amount of data. Therefore, various Hidden Layer consideration analyses could not be performed in this analysis of ANN. Thus, COANN, FWANN, FSANN, HYANN were given the fixed value of 3 for the hidden layer in the tests. The other conditions were set as given by the option provided in Clementine 8.1.

In feature weighting, there is a method using AHP to find relative importance and reflecting on ANN. In feature selection, t-test is used as a method of selecting the variables created by the difference between independent variables and dependent variables. Additionally, in reflecting the feature weighting values for FWANN, an in-depth interview was conducted with three SCM specialists. After summarizing the in-depth interview, AHP, developed by Professor Saaty, was used to calculate the global relative importance for factors [11]. In observing each value, product and process innovation was 0.002, partnership management 0.1, information flows 0.004, threats and substitutes 0.031, waste reduction 0.17, time compression 0.076, flexible response 0.033, unit cost reduction 0.019, view of product/service 0.004, view of timeliness 0.024, view of flexibility 0.014, view of customer value 0.045, profit margins 0.166, cash flow 0.026, revenue growth 0.029, and return on assets 0.087. The consistency index demonstrated under 0.1 successfully [11].

The feature selection of FSANN drew the differences existing in 4 variables of profit margins, cash flow, revenue growth, and return on assets through t-test between 16 independent variables and dependent variables. Feature weighting and feature selection were used successively in HYANN.

The ratio for test data set and holdout data set was 80:20 for the test. These results consisted of 86 test data sets and 22 holdout data sets. The learning rate value for ANN was based on that recommended by SPSS Clementine 8.1. The remaining default values were used as quick algorithm, alpha 0.9, initial eta 0.3, eta decay 30, high eta 0.1, and low eta 0.01.

4 Results

The prediction performances of the four models are compared in this section. Table 2 describes the average prediction accuracy of each model. As table 2 shows, HYANN achieved higher prediction accuracy than COANN, FWANN, and FSANN by 1.5%, 12.03%, and 2.38% for the holdout data, and by 1.62%, 4.19%, and 7.23% for the training data, respectively.

Table 2. Average prediction accuracy of COANN, FWANN, FSANN, and HYANN.

Model	COANN	FWANN	FSANN	HYANN
Training Data	60.000	57.971	54.930	62.162
Holdout Data	84.210	73.684	83.330	85.714

According to this research outcome, HYANN gave the best forecast for SCM sustainable collaboration. The order of improving forecasting performance was as follows: FWANN < FSANN < COANN < HYANN. Reflecting on the opinions of specialists is very important for realizing Feature Weighting using AHP. However, it lacks the performance of forecasting SCM sustainable collaboration through the process of reflecting the relative importance as calculated by AHP. Two reasons for facing these problems may be the use of an insufficient number of specialist opinions and of samples used in ANN.

Likewise, operating a network with selected factors, for which the statistical difference between independent variables and dependent variables has been verified using Feature Weighting through t-test, also showed lower performance accuracy than COANN. This is due to inaccurate measurements of Likert scale when measuring the BSC performance data of those who are carrying out SCM sustainable collaboration.

5 Conclusions

This research has optimized the forecasting of a sustainable collaboration performance using HYANN. This research result is very significant in confirming a more accurate decision-making model for SCM management.

The research featured the following limitations. First, the evolution of SCM is in the early stage in the Korean business market. Due to an insufficient number of samples, ANN tests were not diverse enough. Therefore, for future research it is necessary to collect more samples of cooperation using SCM to increase the diversity of feature weighting and feature selection in the comparison of actual analysis. Second, the variables for measuring SCM performance were used with the partially altered variables supplied by Brewer and Speh [1]. The feature of each field of cooperation where SCM is used should be reflected in future research. ANN analysis should be more elaborate and quantitative analysis should be used in measuring the index of SCM performance. Third, ANN has its own limitations. For example, a supplement is required to avoid falling in local optimum when using the hill climbing method of ANN. In order to prevent these problems, genetic algorithms and global search algorithms need to be used. This also requires simultaneous optimization, rather than sequential optimization, to obtain an improved outcome for optimization. However, this approach may require a great deal of time to achieve a better optimization. Hence, to achieve more accurate forecasting of SCM sustainable collaboration, diverse machine learning algorithms such as GA (Genetic Algorithm) and SVM (Support Vectors Machines) must be applied. These improvements will assist in introducing the general application of SCM sustainable collaboration in upcoming years.

Acknowledgements

This research was supported partially by the Chung-Ang University Research Grants in 2005. Correspondence concerning this article should be addressed to Juhee Hahn, Department of Business, Chung-Ang University.

References

1. Brewer, C. Peter and T. W. Speh.: Using the Balance Scorecard to Measure Supply Chain Performance. Journal of Business Logistics, **21** (2000) 75-93
2. Flecher, D., and Goss, E.: A Comparative Analysis of Artificial Neural Network using Financial Distress Prediction. Information and Management, **24** (1993) 159-167

3. Hebb, D. O.: The Organization of Behavior : A Neuropsychological Theory, New York : Wiley (1949)
4. Hornik, K.: Approximation Capability of Multilayer Feedforward Networks. Neural Networks, **4** (1991) 251-257
5. Hyunchul Ahn, Kyoung-jae Kim, and Ingoo Han.: Hybrid Genetic Algorothms and Case-Based Reasoning Systems. Lecture Note in Computer Science (2004)
6. Kai et al.: Feature Weighting and Instance Selection for Collaborative Filtering : an Information Theoretic Approach. knowledge and information system **5** (2003) 201-224
7. K. Tam, M. Kiang.: Managerial Application of Neural Networks : The Case of Bank Failure Predictions. Management Science, **38** (1992) 926-947
8. McCulloch, W. S. and Pitts, W.: A Logical Calculus of The Ideas Immnanent in Nervous Activity. Bulletin of Mathematical Biophysics, **5** (1943) 115-133
9. Rosenblatt, F.: The Perceptron: A Probabilistic Model For Information Storage And Organization In The Brain. Psychological Review, **65** (1958) 386-408
10. Rovert S. Kaplan and David P. Norton.: The Balanced Scorecard Measures that Drive Performance. Harvard Business Review, **70** (1992) 71-79.
11. Saaty, Tomas L.: The Analysis Hierarchy Process. McGraw-Hill Book Company (1980)
12. Tam, K., and Kiang, M.: Managerial Application of Neural Networks : the Case of Bank Failure Prediction. Management Science, **38** (1992) 926-947
13. Wilson, R, and Sharda, R.: Bankruptcy Prediction using Neural Networks. Decision Support Systems, **11** (1994) 545-557
14. Yang, Z. R., M. B. Platt and H. D. Platt.: Probalilistic Neural Networks in Bankruptcy Prediction. Journal of Business Research, **44** (1999) 667-74

Multiple Criteria Inventory Classification Based on Principal Components Analysis and Neural Network

Quansheng Lei, Jian Chen, and Qing Zhou

School of Economics and Management, Tsinghua University, Beijing 100084, China
leiqsh@em.tsinghua.edu.cn

Abstract. The paper presents two methods for ABC classification of stock keeping units (SKUs), The first method is to apply principal components analysis (PCA) to classify inventory. The second method combines PCA with artificial neural networks (ANNs) with BP algorithm. The reliability of the models is tested by comparing their classification ability with a data set. The results show that the hybrid method could not only overcome the shortcomings of input limitation in ANNs, but also further improve the prediction accuracy.

1 Introduction

A company often has thousands of different type of items in inventory, it is useful to classify products by importance, especially, when resource is limited. As a widely used inventory manage method, ABC classification is original from Villerfredo Pareto, who thought that 20% of the people controlled 80% of the wealth.

Inventory items are categorized as A, B, C according to dollar value per unit multiplied by annual usage rate, common known as dollar usage [1]. That is, item have been ordered in descending order of their annual usage values. The top 5%-10% of items usually account for 60%-70% of dollar usage and constitute class A, the next 20%-30% of items represent approximately 20% of dollar usage and belong to class B, and class C present 50%-70% of items and 15% of dollar usage. Class A contains the items with the highest priority and receives the most attention. The medium Class B receives less attention, and Class C, which contains the items with the lowest priority, is controlled routinely.

The wide use of ABC classification is its simple, that is to use the annual dollar usage, but it has a serious drawback, many items may be have other criteria which is very important to manager, such as the certain of supply, the rate of obsolescence, the criticality of the item and the impact of stock-out of items, and some of these may even weigh more heavily than dollar usage for inventory management.

Some researchers have proposed many methods to receive it, Flores and Whyback [2],[3] consider multiple criteria for ABC classification, and use the joint criteria matrix for two criteria, but there is no obvious way with their methods to extend the procedure to more than two criteria, and the weighs of different criteria are assumed to be equal.

Analytical Hierarchy Process (AHP) developed by Saaty [10] has been successfully applied to multicriteria inventory classification by many authors [4],[7],[8]. The advantage of AHP is that it can incorporate many relevant qualitative and quantitative criteria, but the important drawbacks of the method is that a significant amount of subjectivity if involved in pair wise comparisons of criteria, rating levels and assigning a rating level and associated weights.

In recently, some researchers study inventory classification with artificial intelligence, Guvenir et al. [5] using genetic algorithm to optimize the parameters that represent the weights of criteria, Partovi and Anandarajan [9] presents artificial networks for ABC classification. Although ANNs have several advantages, but the number of variables that can be into the models are limited.

The paper presents two methods to classify inventory. The first method is to apply principal PCA to classify the inventory. The second method is to extract the principal components of different attribute of items by PCA, and then input them to neural networks to determine the inventory classification of items.

The rest of the paper is organized as follows. Section 2 reviews the methods of PCA and ANNs, we analysis the results in section 3 and conclude with section 4.

2 Methods

Principal components analysis (PCA) is the most common multivariate-statistical technique for reducing the dimensionality of multivariate data [6]. Its aim is to exhibit the underlying dimensions in a data set and explore the relationships among individuals. As it is well known, PCA consists in determining a small number of principal components that recover as much variability in the data as possible. These components are linear combinations of the original variables. The interpretation of the principal components is generally based on the respective magnitudes of the loadings assigned to the variables.

Consider an m-dimensional data set $X=[x_1,x_2,...,x_m]$ whose variance-covariance matrix has eigenvalue -eigenvector pairs

(λ_1 , p_1), (λ_2 , p_2) , ... , (λ_m , p_m) ,where $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_m \geq 0$.

The principal component decomposition of X can be defined as:

$$X = TP^T + E = \sum_{i=1}^{l} t_i p_i^T + E \quad (l < m) \cdot \tag{1}$$

where $T=[t_1 ,t_2 , ... ,t_l]$ is the matrix of principal component scores, $P=[p_1 , p_2 , ... , p_l]$ is the matrix of principal component loadings, and E is the residual matrix in the sense of minimum Euclidean norm. If the process variables are collinear or correlated, the dimensionality of the problem can be reduced through the application of PCA and a smaller number of principal components than original variables can be used to explain the major underlying sources of variability in the data. In this way the first l principal components, where $l < m$, are usually sufficient to describe the major sources of variability in the data.

Artificial neural networks (ANNs) is a computer based system derived from the simplified concept of the brain in which a number of nodes, called processing elements or neurons, are interconnected in a netlike structure. Three components constitute an ANNs: the processing elements, the topology of connection between the nodes, and the learning rules. The ANNs have two advantages: First, ANNs are capable of detecting and extracting nonlinear relationships and interactions among predictor variables. Second, the inferred patterns and associated estimates of the precision of the ANNs do not depend on the various assumptions about the distribution of variables.

A multi-layer perceptron (MLP) is a kind of widely used ANNs. Between the input layer and output layer of this kind of network are several hidden layers. Theoretically, a MLP with a hidden layer can achieve an estimation of the function very well through an appropriate training.

The training of a MLP usually adopts a back-propagation (BP) algorithm, BP is the best known of the neural network training algorithms in which the gradient vector of the error surface is calculated. This vector points along the direction of the steepest descent. Moving along the vector a short distance will decrease the error. Repeating this process and moving along the vector in shorter distances will eventually find a minimum.

In this paper, the PCA is done with the statistical software SPSS and BP neural network is used with software MATLAB.

3 Results and Discussion

3.1 Material

The data of this paper is come from Partovi and Anandarajan [9], which is a real-world data obtained from a large pharmaceutical company located in northeastern United States. Each data set represented a spare part and contained four types of information: unit price, ordering cost, demand range and lead time.

3.2 PCA Analysis

Firstly, we standardized the input sample and compute its correlation matrix. From table 1 we can see that there are some correlation among unit price, ordering cost, demand range and lead time, therefore, these four criteria exist more overlap about information with inventory classification.

Table 1. The correlation matrix of input data.

1.000	.990	-.311	.702
.990	1.000	-.302	.670
-.311	-.302	1.000	-.445
.702	.670	-.445	1.000

Secondly, we can compute eigenvalues, eigenvectors and cumulative variance with the statistical software SPSS, and the results see in Table 2 and Table 3.

Table 2. Total Variance Explained.

Eigenvalues				Eigenvectors			
0.0090	0	0	0	0.7214	0.2656	-0.2962	0.5669
0	0.3579	0	0	-0.6910	0.3361	-0.3099	0.5599
0	0	0.8451	0	-0.0049	-0.3032	-0.8973	-0.3206
0	0	0	2.7880	-0.0460	-0.8512	0.1049	0.5122

Table 3. Total Variance Explained.

Component	Initial Eigenvalues		
	Total	% of Variance	Cumulative %
1	2.788	69.688	69.688
2	.845	21.133	90.821
3	.358	8.952	99.773
4	9.077E-03	.227	100.000

In table 3, we can see that the total variance of the first three principal components account to 99.773%, and they may be seen as significant and be selected to classify inventory.

Thirdly, for explaining the economical implication of the principal components, we compute rotated component matrix as following:

Table 4. Rotated Component Matrix.

	Component		
	1	2	3
UPRICE	.935	-.134	.322
OCOST	.949	-.130	.279
DEMAND	-.131	.975	-.181
LTIME	.414	-.238	.879

In table 4, we can see that the magnitudes of unit price and ordering cost in the principal component 1 are 0.935 and 0.949 respectively, so the principal component 1 reveals the item attribute of unit price and ordering cost. Similarly, the principal component 2 show the item attribute of demand, and the item attribute of lead time is illustrated in the principal component 3. Therefore, these three principal components completely show the attributes of all items, which are unit price, ordering cost, demand range and lead time.

Fourthly, we need give the integrative score and its order of all the items in order to classify inventory, so we select first three factors and model synthesis appraising function as following:

$$F=69.688*Factor1+21.133*Factor2+8.952*Factor3. \quad (2)$$

The experimental sample is spilt into two distinct groups, namely, a training group (50 items) and a testing group (45 item). From formula (1), we can use the training group to obtain the classify criteria, and the results of inventory classification of testing group see Table 5.

3.3 Artificial Neural Networks

The selected PCs are processed by ANNs, which is trained with the back-propagation (BP) algorithm. Firstly, we use the first three PCs as input variables to the ANNs in order to determine the optimum classification of the inventory. The ANNs consisted of 3 input neurons, 6 hidden neurons and 3 output neurons (namely, inventory items A, B or C), as illustrated in Fig.1.

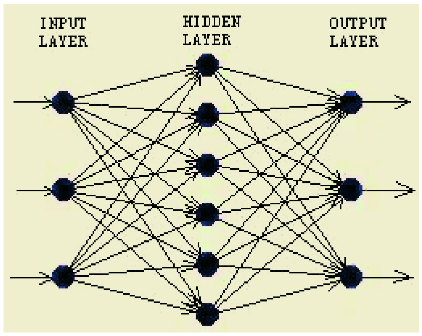


Fig. 1. Structure of the ANNs.

Secondly, Using the training group (50 items), the network models were trained, and the trained ANNs was used to test the selection accuracy of the network for the 45 testing data set, and the results see Table 5.

3.4 Results

In order to study the effectiveness of hybrid method, the classification results of hybrid method and ANNs were compared with PCA. In Table 5, it has been shown that the predictive accuracy with hybrid method is more than ANNs and PCA.

Table 5. Prediction accuracy among hybrid method , ANNs and PCA.

	Hybrid method(%)	ANNs(%)	PCA(%)
Training	82.0	80.00	72.00
Testing			
Over classification	77.78	71.11	60.00
Item A	71.43	57.14	85.71
Item B	66.67	60.00	60.00
Item C	86.96	82.61	52.17

4 Conclusions

This paper presents two new approaches for ABC classification of various SKUs. PCA is a very powerful dimensionality reduction technique, whilst retaining as much as the variability of the original data as possible. Firstly, we use PCA to give the integrative score and its order of all items and then classify inventory. Secondly, we use the extracting PCs as input variables to the ANNs with BP algorithm. The ANNs were a valuable tool for the classification of the inventory due to its simplicity and excellent predictive ability. The results show that the hybrid method has two advantages: (1) It can overcome the shortcomings of input limitation in ANNs, (2) It improves the prediction accuracy. Although two proposed methods are promising analytical tools in the multiple criteria inventory classification, the hybrid method has a higher predictive accuracy than PCA.

Acknowledgment

This research is supported by National Natural Science Foundation of China (No.70401021), and by the Chinese Postdoctoral Science Foundation (No.200 4036266).

References

1. Cohen, M.A., Ernst, R.: Multi-item Classification and Generic Inventory Stock Control Policies. Production and Inventory Management Journal, **29** (1988) 6-8
2. Flores, B.E., Whybark, D.C.: Mutiple Criteria ABC Analysis. Journal of Operations Management, **6** (1986) 38-45
3. Flores, B.E., Whybark, D.C.: Implementing Multiple Criteria ABC Analysis. Journal of Operations Management, **7** (1987) 79-84
4. Gajpal, P.P., Ganesh, L.S., Rajendram, C.: Criticality Analysis of Spare Parts Using the Analytic Hierarchy Process. International J. of Production Economics, **35** (1994) 293-297
5. Guvenir, H.A., Erel, E.: Multicriteria Inventory Classification Using A Genetic Algorithm. European Journal of Operational Research, **105** (1998) 29-37
6. Joliffe, I.T.: Principal Component Analysis. Springer-Verlag, New York (2002)
7. Partovi,F.Y., Burton,J.: Using the Analytic Hierarchy Process for ABC Analysis. International Journal of Production and Operations Management, **13** (1993) 29-44
8. Partovi, F.Y., Hopton,W.E.: The Analytic Hierarchy Process As Applied to Two Types of Inventory Problems. Production and Inventory Management Journal, **35** (1994) 13-19
9. Partovi, F.Y., Anandarajan,M.: Classifying Inventory Using An Artificial Neural Network Approach.. Computers and Industrial Engineering, **41** (2002) 389-404
10. Saaty, T.L.: The Analytic Hierarchy Process. New York, NY: McGraw-Hill (1980)

Author Index

Abraham, Ajith III-439
Acuña, Gonzalo II-448, III-833
Ahn, Taechon III-107
Alam, Mohammad S. II-772
Alvestad, Per III-851
Amari, Shun-ichi I-1
Asari, Vijayan K. I-485

Banerjee, Satarupa II-556
Bao, Fumin II-747
Bao, Zheng I-913, II-369
Barna, Resko III-1046
Bashyal, Shishir III-796
Becerikli, Yasar II-411
Beham, Michael III-865
Ben, Kerong III-567
Bian, Zhengzhong II-690
Bo, Liefeng I-851
Bolden, Cortney III-92
Boldin, Maxim III-1046

Cai, Huiying III-971
Cai, Jiamei II-637
Cai, Jun III-680
Cai, Nian II-673
Cai, Qutang II-526
Cai, Wei II-741
Cai, Yuanli II-436
Cai, Yunze I-843, II-442, III-501
Cai, Zhiping I-805
Cai, Zixing III-265, III-285
Cao, Binggang III-247
Cao, Feilong I-39, I-63
Cao, Jinde I-102, I-162, I-288, I-706
Cao, Wenming II-87, II-345
Carvajal, Karina III-833
Celebi, M. Emre II-292
Chacón, Max III-833
Chai, Tianyou III-167, III-871, III-1028
Chai, Yi III-470, III-654, III-1004
Chai, Yueting II-430
Chang, Lungyi III-628
Chang, Pei-Chann III-947
Chang, Ray-I III-767
Chang, Thoonkhin III-86
Chang, Yaw-Jen III-538
Chau, Kwokwing III-1034, III-1040
Chau, Rowena II-238
Chen, Bin III-992
Chen, Bo II-637
Chen, Boshan I-282, I-294
Chen, Chinghan II-698
Chen, Dingguo III-42
Chen, Dongyue II-14
Chen, Enhong II-232
Chen, Gang III-992
Chen, Guangjun III-591
Chen, Guanrong II-630
Chen, Haixia I-526
Chen, Hongda III-494
Chen, Hungcheng III-628
Chen, Jeanne III-409
Chen, Jiajian III-476
Chen, Jian III-1058
Chen, Jin III-531, III-603
Chen, Kai II-690
Chen, Luonan III-640
Chen, Ning II-222
Chen, Poyueh III-320
Chen, Quanshi I-615
Chen, Rongchang II-916, III-409
Chen, Shengyong I-762, III-463
Chen, Shiming I-805
Chen, Shuangping II-45
Chen, Songcan II-659
Chen, Teyu III-409
Chen, Tianping I-215, I-308, II-466
Chen, Tieming II-637
Chen, Tungshou II-916, III-409
Chen, Weimin III-890
Chen, Weisheng III-30
Chen, Wenbin I-546
Chen, Xiaolin III-609
Chen, Xin III-228
Chen, Xuedong I-132
Chen, Xuhui I-770
Chen, Yang III-185, III-397
Chen, Yenwei I-949, II-262
Chen, Yong I-455
Chen, Yuehui III-439

Chen, Yunping III-654
Chen, Zehua III-674
Chen, Zhaojian II-131
Chen, Zhenzhou I-817
Cheng, Chuntian III-1040
Cheng, Hong II-226
Cheng, Hongmei II-45, II-232
Cheng, Jian I-949
Cheng, Qiansheng I-955
Cheng, Shiduan III-391
Cheng, Wengang II-819
Cheng, Xiefeng II-508
Cheng, Yonghong III-609
Cheung, Yiuming I-572, II-472
Chi, Zheru II-274
Chicharo, Joe F. II-520
Chien, Chengchih I-380
Chien, Yuer II-916
Cho, Jaehyun II-209
Choi, Nakjin I-558, I-578
Chu, Heng II-568
Chuang, Chenglong I-380
Chung, Duckjin I-660, II-665
Chung, FuLai II-801
Ci, Song III-343
Cichocki, Andrzej I-973, II-502
Çivicioğlu, Pınar II-679
Cong, Xiangping II-550, II-795
Cui, Pingyuan III-21
Cui, Suyuan III-445

Dai, Qionghai III-403
Dai, Wenzhou III-470
Dang, Chuangyin I-712
de Almeida, Aníbal T. III-173
Deng, Beixing III-977
Deng, Feiqi I-174
Deng, Shaojiang I-991, II-868
Diao, Jinhui III-561
Ding, Fenghua II-765
Ding, Gang I-473
Ding, Yongsheng III-119
Dittenbach, Michael II-75
Dong, GM III-603
Dong, Guangbo III-748
Dong, Guangming III-531
Dong, Hao I-931
Dong, Liju III-939
Dou, Quansheng III-903
Du, Jixiang II-281
Du, Lan II-369
Duan, Haixin III-476
Duan, Shukai I-363, I-491
Duan, Zhongxing III-482
Duan, Zhuohua III-265, III-285

Englund, Cristofer I-461
Er, Meng Joo I-434, III-216
Evans, David J. I-265

Fan, Jian I-634
Fan, Jinghong III-992
Fan, Jinsheng I-875
Fan, Mingming II-618
Fan, Shu III-640
Fan, Youping III-654
Fan, Zhun I-652
Fang, Minglun III-851
Fei, Shumin III-36, III-54
Feng, Bo II-399
Feng, Boqin II-222, III-349
Feng, Chunbo I-69, I-271, I-694, III-550
Feng, Dalong III-1004
Feng, Dazheng I-1004
Feng, Jianfeng I-831
Feng, Jun I-875
Feng, Shan I-652
Feng, Xin I-776
Feng, Yaning II-848
Feng, Yong III-428
Feng, Zhiquan II-508
Ferreira, Fernando J.T.E. III-173
Freeman, Walter J. I-21
Fu, Chaojin I-276
Fu, Jianguo III-297
Fu, Peng I-931, III-482
Fu, Xianghua III-349
Fu, Xiuju I-682
Fu, Yan II-63
Fu, Yonggang II-789
Fu, Yuli II-484, II-532

Gan, Liangzhi I-837
Gan, Woonseng III-161
Gao, Dengpan I-955
Gao, Hongli III-883
Gao, Jinwu I-811
Gao, Xinbo II-57, II-363, II-375
Gao, Xinying III-303

Gao, Yang I-621
Gao, Ying III-198
Gao, Zhiwen II-81
Ge, Baoming III-173
Ge, Fei III-703
Ge, Zhirong II-759
Geng, Guohua II-292
Gheorghiu, M.D. III-192
Gillingwater, David III-998
Givigi, Sidney Nascimento I-398
Gómez, Angel Manuel I-943
Górriz, Juan Manuel I-943
Gong, Dengcai III-634
Gong, Maoguo I-369
Gong, Weiguo II-118
Gou, Shuiping II-286
Gu, Bo III-309
Gu, Hongying II-81
Gu, Shenshen I-750
Gu, Weikang I-203
Gu, Wenjin III-150
Gu, Xiao II-281
Gu, Xiaodong II-836
Gu, Yingkui III-927
Guan, Qiu I-762, III-463
Guan, Xinping I-314, III-445
Guan, Zhiguang II-765
Guo, Chen III-113
Guo, Chuang I-369
Guo, Jun II-351
Guo, Lei I-69, III-550
Guo, Maoyun III-470, III-1004
Guo, Mingen II-854
Guo, Ping I-925, II-310
Guo, Qianjin III-521
Guo, Wei III-391
Guo, Wenqiang III-737
Guo, Ying III-332

Habibipour Roudsari, Farzad III-373, III-662
Hahn, Juhee III-1052
Han, Bing II-363
Han, Fei I-572
Han, Gunhee II-665
Han, Gyusik II-885
Han, Min II-580, II-618
Han, Xiaoxia III-591
Hao, Zhifeng I-869
He, Fang II-643
He, Hangen III-185
He, Hanlin I-199, I-276
He, Jingrui III-397, III-476
He, Lianghua II-137
He, Lin I-247, III-567
He, Liqiang III-955
He, Xing III-501
He, Yi I-672
He, Zhaoshui II-532
He, Zheng III-983
He, Zhengjia I-937
He, Zhenya II-496
Heng, Xingchen I-520
Hinde, Chris III-998
Hjelmervik, Ove R. III-1046
Ho, Tubao II-605
Hong, Keongho II-321, II-842
Hong, Yi I-770
Hou, Jianhua II-854
Hou, Zengguang III-277
Hsiao, Mingchang III-379
Hsiao, Yingtung I-380
Hsieh, Chunhung III-409
Hsieh, Shenghsien II-698
Hu, Baogang I-824
Hu, Baosheng III-143
Hu, Changhua III-591
Hu, Chunhua III-60
Hu, Dewen I-961
Hu, Die II-137
Hu, Dongcheng II-189
Hu, Hanhui III-1022
Hu, Huibo II-508, II-777
Hu, Jianhui II-87
Hu, Kuanghu II-673
Hu, Lingyun I-566
Hu, Qiao I-937
Hu, Tingliang III-60
Hu, Tongfu III-622
Hu, Yafeng II-216
Hu, Yaohua III-813
Hu, Zhenjiang II-118
Hu, Zhonghui III-501
Huang, Chen III-457
Huang, Degen II-246
Huang, Deshuang I-572, II-281, II-478, II-807
Huang, Dexian I-391, III-15, III-131
Huang, Gaoming II-496
Huang, Guangbin I-572
Huang, Hanmin II-624

Huang, Hongzhong I-725, I-800, III-813, III-927
Huang, Jian I-229
Huang, Kuohua III-628
Huang, Linlin II-99
Huang, Shitan II-848
Huang, Sunan III-897
Huang, Tao III-556
Huang, Tingwen I-168
Huang, Wei II-605, II-879
Huang, Wentao I-369
Huang, Xiaoling III-939
Huang, Xiyue II-624
Huang, Xuemei III-774
Huang, Xuming I-985
Huang, Yinlin II-910
Huang, Yu II-777
Huang, Yumei I-187
Hur, Yeon II-928
Hwang, Hyungsoo I-416
Hwang, Kaoshing III-379

Ichikawa, Michinori I-467
Ishitobi, Mitsuaki II-454

Jaipradidtham, Chamni III-668
Jalili-Kharaajoo, Mahdi III-137, III-373, III-662
Jamett, Marcela II-448
Jeong, Eunhwa II-321, II-842
Ji, Ce I-209
Ji, Hongbing II-363, II-375
Ji, Jian II-730
Jia, Peifa III-291, III-908
Jia, Ying I-1010, II-538
Jia, Yupeng III-303
Jian, Xiao III-179
Jiang, Dongxiang III-561
Jiang, Haijun I-102
Jiang, Jingqing I-863
Jiang, Joeair I-380
Jiang, Kai I-526
Jiang, Minghu III-971, III-977
Jiang, Minghui I-108, I-114, I-156, I-235
Jiang, Mingyan III-326
Jiang, Tao I-997
Jiang, Xingzhou I-997
Jiang, Yingtao I-497
Jiang, Yiwei II-819
Jiang, Yugang II-310
Jiao, Binquan III-877
Jiao, Licheng I-51, I-57, I-369, I-385, I-666, I-851, I-893, II-8, II-26, II-57, II-286, II-723
Jin, Fan I-455
Jin, Mingxi II-339
Jin, Wuyin I-302
Jin, Yihui I-391, III-15, III-131
Jin, Zhijian III-819
Jin, Zi II-643
Jing, Wenfeng II-1
Joo, Younghoon I-603
Ju, Yanwei II-730
Jung, Sungwon III-807

Kalra, Prem K. I-95
Kang, Chunyu II-387
Kang, Yong III-983
Kasabov, Nikola II-51
Kemp, Ray III-965
Khorasani, Khashayar II-713, III-574, III-584
Kim, Dong Seong III-415
Kim, Dongsun I-660, II-665
Kim, Dongwon III-780
Kim, Eunju III-355
Kim, Hongsik II-665
Kim, Hwan Y. III-987
Kim, Hyunki I-404
Kim, Hyunsik I-660, II-665
Kim, Jonghan III-807
Kim, Kwangbaek I-603, II-147, II-209
Kim, Myungwon III-355
Kim, Sooyoung II-904
Kim, Sung-III II-333
Kim, Sungshin I-603
Kim, Tae Seon III-845
Kim, Yongkab I-428, III-987
Kobatake, Hidefumi II-99
Kocsor, András II-357
Kuan, Yeander III-933
Kwan, Chiman II-32, III-494

Lai, K. Robert III-947
Lang, Congyan II-819
Lee, ChiYung II-611, III-320
Lee, Daewon I-857, II-885
Lee, Hyoseok II-904
Lee, Jaewook I-597, I-857, II-885, II-904
Lee, Kidong II-873

Lee, Sangjae II-873
Lee, SeonGu III-780
Lee, SeungGwan I-627
Lee, Sungyoung I-1016
Lee, Tong Heng III-897
Lee, Weijen III-494
Lei, Kaiyou I-672
Lei, Quansheng III-1058
Lei, Sheping I-253, I-259
Lei, Xuanyang III-531
Lei, Xusheng II-789
Leu, Ming C. II-157
Leung, Hofung I-776
Li, Aiguo II-747
Li, Bicheng I-979, II-105
Li, Bin III-247
Li, Chengqing II-630
Li, Chongrong III-397, III-476
Li, Chuandong I-120, I-126, I-346
Li, Chunguo I-81
Li, Chunxia II-298
Li, Fei III-338
Li, Fucai III-603
Li, Fuhai III-697
Li, Guangqin III-719
Li, Guolin I-479
Li, Guozheng I-532
Li, Hongyu I-546
Li, Hua I-340
Li, Hui III-737
Li, Jianhua III-297, III-361
Li, Jianing I-640
Li, Jianwei II-171
Li, Jie I-320, I-782, II-57, II-568
Li, Jing I-467
Li, Junmin III-30
Li, Junquan I-509
Li, Kai III-561
Li, Lee III-204
Li, Lei I-817
Li, Li III-574
Li, Lihua II-246
Li, Linfeng III-634
Li, Lishuang II-246
Li, Luoqing I-538
Li, Min II-741
Li, Minzan III-1017
Li, Ning III-790
Li, Qing I-893
Li, Qingdu I-357
Li, Qunxia II-351
Li, Sen III-332
Li, Shangfu III-470, III-1004
Li, Shaofa II-257
Li, Shaorong I-320, I-327
Li, Shaowen I-320, I-327
Li, Sheng I-831
Li, Shenghong III-361
Li, Shujun II-630
Li, Shutao II-268, II-753
Li, Weihong II-118
Li, Weihua III-556
Li, Wenwei III-367
Li, Wujun II-131
Li, Xiaobing III-742
Li, Xiaodong III-161
Li, Xiaohong III-877, III-983
Li, Xiaoli I-314, III-445
Li, Xiaolin I-162
Li, Xihai II-393
Li, Xing III-476
Li, Xiugai III-15
Li, Xunming III-634
Li, Yan I-314
Li, Yanda III-680
Li, Yangmin III-155, III-228, III-271
Li, Yanlai II-177
Li, Yao III-192
Li, Ye III-501
Li, Yi III-143
Li, Yonghua III-927
Li, Youfeng I-652
Li, Youmei I-39, I-63
Li, Yuan I-532
Li, Yuanqing I-973, II-502
Li, Zhongqi III-584
Liang, Hualou II-544
Liang, Shan II-454, II-910
Liang, Xinran III-361
Liang, Xun I-87, II-897
Liang, Yanchun I-334, I-863, I-869
Liang, Yixiong II-118
Liao, Changrong III-890
Liao, Guanglan III-556
Liao, Huichuan III-385
Liao, Xiao H. III-92, III-192
Liao, Xiaofeng I-120, I-229, I-346
Liao, Xiaoxin I-114, I-156, I-181, I-199, I-276
Liao, Yong III-877

Lien, Hsinchung III-933
Liew, Alan Wee Chung III-709
Lim, JunSeok I-558, I-578
Lim, Sehun II-928, III-1052
Lim, Sunja III-987
Lin, ChengJian II-611, III-320
Lin, Jianyi III-1040
Lin, Mingxing II-765
Lin, Qiuhua II-544
Lin, Wei II-730
Lin, Xiaokang I-1022, III-403
Lin, Yinhua II-910
Ling, Jinjiang III-790
Ling, Wei II-643
Liu, Bao III-119
Liu, Baowei III-760
Liu, Bing I-45
Liu, Bo I-391, I-869
Liu, Chengliang III-550
Liu, Chengming I-985
Liu, Chunsheng I-800
Liu, Cuiyan II-381
Liu, Daizhi II-393
Liu, Daxue III-185
Liu, Derong I-142, I-209
Liu, Dong III-654
Liu, Fang II-813
Liu, Gang II-351, III-1017
Liu, Guangyuan I-363, I-672, I-688
Liu, Guixia III-686
Liu, Hailin II-472
Liu, Hailong II-327
Liu, Hongtao III-992
Liu, Hongwei I-913, II-369
Liu, James N.K. II-399
Liu, Jian III-343
Liu, Jianfeng III-697
Liu, Jilin I-203
Liu, Jingang I-591
Liu, Ju II-508, II-777, III-309
Liu, Jun II-659
Liu, Kuang-Yu III-719
Liu, Long III-597
Liu, Meiqin I-108
Liu, Miao II-298
Liu, Mingyu III-731
Liu, Mingzhe III-965
Liu, Qiang II-466
Liu, Quan III-903
Liu, Sanyang I-718
Liu, Shirong II-418
Liu, Shubao III-240
Liu, Tianming I-203
Liu, Tie II-226
Liu, Wenbing I-193
Liu, Wenhong III-737
Liu, Wenlong II-586
Liu, Xianghui I-805
Liu, Xiaodong III-403
Liu, Xiaohua II-298
Liu, Xiaojing III-48
Liu, Xiaoying III-403
Liu, Yangyang I-34
Liu, Yankui I-811
Liu, Yi II-430, III-125
Liu, Yu I-520
Liu, Yue I-532
Liu, Yuehu I-906
Liu, Yugang III-271
Liu, Yuhua II-580
Liu, Zhiyong III-955
Liu, Zihong I-479
Liu, Zunxiong III-385
Lo, Ken II-274
Long, Fei III-54
Lu, Anwen I-120
Lu, Baoliang I-467, I-646, I-881, I-887, II-39
Lu, Congde II-124, II-424
Lu, Hongtao I-247, II-789, II-801
Lu, Jiwen II-183
Lu, Ke I-677
Lu, Wei II-801
Lu, Wenlian I-215
Lu, Xiaodan II-14, III-198
Lu, Xicheng I-805
Lu, Yiyu III-877
Luan, Zhiyan III-131
Luo, Bin II-131
Luo, Hongwen III-903
Luo, Yiping I-174
Luo, Yupin II-189, III-457
Luo, Zhiwei II-813
Luo, Zhonghui III-827
Lv, Hairong II-203
Lv, Jiancheng I-585
Lv, Qi II-538

Ma, Jian III-748
Ma, Jinwen I-955, III-697, III-703
Ma, Liying II-713, III-574, III-584

Ma, Runnian I-253, I-259
Ma, Shugen III-247
Ma, Siliang II-828
Ma, Tianyi I-497
Ma, Xiaohong II-550, II-586, II-795
Ma, Xiuli I-51, I-666, II-26
Ma, Zhaofeng I-931
Ma, Zhenying III-421
Magne, Luis III-833
Mao, Chengxiong III-640
Mei, Tiemin II-520
Meng, Deyu II-1
Meng, Guang III-597
Meng, Hongling I-446
Meng, Kaiwen II-69
Meng, Zhiqing I-712, II-251
Miao, Xianglin II-690
Miao, Yalin II-690
Mishra, Deepak I-95
Mitra, Vikramjit II-556
Mittal, Gauri S. III-259
Mu, Aiping I-1022
Mu, Xuewen I-718

Nakamori, Yoshiteru II-879
Nakao, Zensho II-262
Nguyen, Ha-Nam III-415
Nhat, Vo Dinh Minh I-1016
Ni, Lin III-434
Nie, Weike II-8
Nie, Yinling II-8
Ning, Zuogui III-531
Niu, Yanmin II-171

Oh, Am-Sok I-603
Oh, Amsuk II-209
Oh, Sungkwun I-404, I-410, I-416, I-422, I-428, III-107, III-987
Ohn, Syng-Yup III-415
Ou, Zongying II-93, II-854
Ouyang, Gaoxiang I-314, III-445
Oz, Cemil II-157, II-195
Ozawa, Seiichi II-51

Pan, Ancheng III-1022
Pan, Chen III-754
Pan, Guanyu III-903
Pan, Xiaoxia II-345
Pan, Yingjun II-118
Pang, Shaoning II-51
Pang, Yunfu III-567
Park, Byoungjun I-404, I-422
Park, Daehee I-410
Park, GwiTae III-780
Park, Hosung I-410
Park, Jong Sou III-415
Park, Junghee II-873
Park, Keonjun I-416
Park, Lae-Jeong II-922
Parthasarathi, Rishikesh I-434
Pavel, Misha II-598
Pei, Jinan I-187, I-241
Peng, Chunyi II-526
Peng, Dezhong I-1028
Peng, Hong III-179
Peng, Jing II-568
Peng, Jun II-860
Peng, Qicong II-735
Peng, Qinke III-143
Peng, Xianghe III-992
Pernía, Oscar I-943
Pölzlbauer, Georg II-75
Postalcioglu, Seda II-411
Prahlad, V. III-897
Pu, Xiaorong II-112
Puntonet, Carlos García I-943

Qian, Tao II-32, III-494
Qiao, Jianping III-309
Qiao, Qingli I-745
Qin, Jianzhao I-973, II-502
Qin, Zheng I-520, II-747
Qiu, Jianlong I-288
Qiu, Tianshuang II-490, III-332, III-737, III-742
Qiu, Yuhui I-672
Qu, Hong I-739
Qu, Liangsheng III-508
Quek, Chai II-165

Rauber, Andreas II-75
Ray, Sudipta I-95
Ren, Changrui II-430
Ren, Guanghui II-381
Ren, Lihong III-119
Ren, Zhubing III-494
Roh, Seokbeom I-428, III-107
Ryu, Joungwoo III-355

Sadri, Mohammadreza III-373
Sang, Jun II-772
Sang, Nong II-316
Scheer, Sergio I-731
Seow, Ming-Jung I-485
Shang, Li II-478, II-807
Shang, Yanlei III-391
Shao, Zhiqiong I-609
Sheeley, Joseph III-494
Shen, Haifeng II-351
Shen, I-Fan I-546
Shen, Ruimin II-789
Shen, Yi I-108, I-114, I-156, I-181, I-235
Shen, Zhipeng III-113
Sheng, Wanxing III-616
Shi, Chenjun III-113
Shi, Daming II-165
Shi, Tielin III-556
Shi, Weiren II-910
Shi, Yinshui II-375
Shi, Zhangming III-813
Shi, Zhenghao II-848
Shi, Zhiwei II-580
Shi, Zhongzhi I-591
Shimizu, Akinobu II-99
Shou, Yongyi I-794
Shu, Jiwu I-700
Sim, Eoksu III-807
Siqueira, Paulo Henrique I-731
Smith, Kate A. II-238
Song, HeeJun III-780
Song, Heping II-246
Song, Qiankun I-132
Song, Yong D. III-92, III-192
Steiner, Maria Teresinha Arns I-731
Su, Daizhong I-473
Su, Hongtao I-913
Su, Jianbo II-405, II-813, III-222
Su, Tieming II-93
Sun, Caixin III-774
Sun, Catitang II-298
Sun, Changyin I-271, I-694, III-634
Sun, Fuchun III-204, III-234
Sun, Haibo II-93
Sun, Huali I-756
Sun, Hui II-189
Sun, Jianbo III-113
Sun, Jiancheng II-424
Sun, Jiande II-777
Sun, Jitao I-222
Sun, Lingiang III-851
Sun, Qiang III-361
Sun, Shixin III-488
Sun, Shouliang III-326
Sun, Wansheng III-654
Sun, Wei II-274
Sun, Weitao I-700
Sun, Weixiang III-603
Sun, Xipeng I-320
Sun, Xusheng I-375, II-685
Sun, Yingguang III-1040
Sun, Yongmei III-737
Sun, Youxian I-837, III-1, III-7
Sun, Zengqi I-515, I-566, III-60, III-234, III-253, III-748
Sun, Zhanli II-478, II-807
Sun, Zhanquan III-801
Sun, Zonghai I-837, III-7
Sung, KoengMo I-558, I-578

Tan, Kok Kiong III-897
Tan, Min III-277
Tan, Shan I-51, I-57
Tan, Shunwen III-379
Tan, Songsong II-910
Tan, Ying I-440, II-574
Tan, Zheng II-741
Tang, Guangrong III-457
Tang, Haoyang I-126
Tang, Hong III-332
Tang, Jian I-515, III-253
Tang, Lixin I-776, I-788
Tang, Qiling II-316
Tang, Xusheng II-93
Tang, Yuangang III-204, III-234
Tang, Yuanyan I-265
Tang, Zhide III-774
Tang, Zhihong I-174
Temeltaş, Hakan III-210
Teng, Li I-546
Teng, Zhidong I-102
Tian, Jinsi II-405
Tian, Qiming II-189
Tian, Zheng II-643, II-730
Tian, Zhigang I-725
Tie, Ming III-167, III-871
Tong, Hanghang III-397, III-476
Tran, Quang-Anh III-476
Tsai, Hungming III-320
Tu, Fenghua I-346

Valenzuela, Tito III-833
van Alphen, Deborah II-262
Verikas, Antanas I-461

Wan, Chunru I-45
Wang, Bin II-651, II-759
Wang, Bo I-979
Wang, Chiajiu II-556
Wang, Chiang I-380
Wang, Chong II-550, II-795
Wang, Chongjun II-131
Wang, Dongli I-621
Wang, Dongsheng III-531
Wang, Dongxia II-592
Wang, Fangju III-259
Wang, Fenghua III-819
Wang, Gang I-961
Wang, Guizeng III-622
Wang, Guoping II-598
Wang, Guoyin I-455
Wang, Haixiao II-222
Wang, Hong III-550
Wang, Hongbing III-674
Wang, Hongbo II-339
Wang, Jiao III-914
Wang, Jin III-86
Wang, Jue II-327, III-731
Wang, Jun III-179, III-240
Wang, Junping I-615
Wang, Kaian I-887
Wang, Kesheng III-851, III-1046
Wang, Kongqiao I-949
Wang, Kuanquan II-177
Wang, Lei II-8, III-482, III-488
Wang, Lidan I-363, I-491
Wang, Ling I-391, I-497, I-851
Wang, Lipo I-45, I-682
Wang, Lunwen I-440
Wang, Manghui I-503
Wang, Min I-385, I-893, II-562, II-723, III-79
Wang, Pan I-652
Wang, Ping I-800
Wang, Qi III-921
Wang, Qingchao III-98
Wang, Qun I-181
Wang, Ruili III-965
Wang, Shoujue I-34, II-87, II-339, II-345
Wang, Shouyang II-879
Wang, Shuang I-57, II-26
Wang, Shuning I-375, II-685
Wang, Sunan III-616
Wang, Wanliang I-762, III-463
Wang, Wei III-760
Wang, Wenyuan II-203
Wang, Wilson II-460
Wang, Xiaofeng II-281
Wang, Xianhui I-520
Wang, Xiaodong II-707
Wang, Xiaohong III-315
Wang, Xiaohua II-765
Wang, Xiaozhen III-827
Wang, Xihuai III-648
Wang, Xinfei III-983
Wang, Xiong III-839, III-914
Wang, Xiongxiang III-531
Wang, Xiuhong I-745
Wang, Xizhao I-81
Wang, Xuchu II-171
Wang, Xuebing I-991
Wang, Xuexia II-381
Wang, Xufa II-45, II-232
Wang, Yaonan II-268, II-753
Wang, Yi III-851, III-1046
Wang, Yingcai III-297
Wang, Yong I-824
Wang, Yongfu III-167
Wang, Yongqiang III-7
Wang, Yu II-586
Wang, Yuehong II-685
Wang, Yuguo III-1010
Wang, Zengfu I-282
Wang, Zhanshan I-142
Wang, Zhengqu I-745
Wang, Zhihua I-479
Wang, Zhongsheng I-199
Wei, Na II-292, III-567
Wei, Pengcheng II-860
Wei, Xiaopeng I-193
Wen, Guangrui III-508
Wen, Junhao II-69
Wen, Xianbin II-304
Wen, Xiangjun I-843, II-442
Wen, Xiaotong III-725
Wen, Yimin I-881
Weng, Jiansheng III-890
Williams, Peter I-831
Wong, Stephen I-203, III-719
Wu, Baihai III-827
Wu, Chengshong III-379

Wu, Chunguo I-863
Wu, Danyang II-828
Wu, Deyin I-126
Wu, Fei II-81
Wu, Genfeng I-532
Wu, Gengfeng I-634
Wu, Hongyan II-69
Wu, Jiansheng III-965
Wu, Kaigui III-428
Wu, Kaiyang III-409
Wu, Lingyao I-69
Wu, Shuanhu III-709
Wu, Shunjun II-562
Wu, Si I-831
Wu, Sitao III-315
Wu, Tao III-185
Wu, Wei I-609
Wu, Weidong I-800
Wu, Xing III-603
Wu, Yadong II-735
Wu, Yan II-339
Wu, Ying I-302
Wu, Zhilu II-381
Wu, Zhongfu I-340, II-69, III-428

Xi, Guangcheng I-552, I-640, III-801
Xi, Jiangtao II-520
Xi, Jianhui II-580, II-618
Xia, Bin II-514
Xia, Feng III-1
Xia, Liangzhen I-271
Xia, Mengqin I-677
Xia, Xiaohua III-131
Xia, Zhijun II-387
Xiang, Lan I-308
Xiang, Li I-187
Xiang, Tao I-229
Xiang, XiaoLin I-739, I-1028
Xiao, Di II-868, III-434
Xiao, Gang II-87
Xiao, Jianmei III-648
Xiao, Ming II-484
Xiao, Li I-688
Xie, Gang III-674
Xie, Guihai III-748
Xie, Jianying I-756, I-782
Xie, Keming III-674
Xie, Li I-203
Xie, Shengli II-484, II-532
Xie, Xiaojun III-609
Xie, Yuan I-782
Xing, Fei I-925
Xing, Hongjie I-824
Xiong, Haitao II-673
Xiong, Qingyu I-126, II-910
Xiong, Zhihua III-125, III-839
Xiong, Zhongyang III-428
Xu, Aidong III-521
Xu, Bingji I-181
Xu, Daoyi I-187, I-241
Xu, De II-819
Xu, Dianjiang II-131
Xu, Hongji III-309
Xu, Hua III-291, III-908
Xu, Jianfeng II-257
Xu, Jianhua I-900
Xu, Jianwen II-20
Xu, Jianxue I-302
Xu, Jie I-538
Xu, Lei I-5
Xu, Mingheng III-883
Xu, Qingjiu III-150
Xu, Qingxin II-765
Xu, Roger II-32, III-494
Xu, Tingxue III-150
Xu, Xiaoming I-843, II-442, III-501
Xu, Xin I-961
Xu, Xinli I-762
Xu, Xu I-334
Xu, Yiqiong I-979
Xu, Yongmao III-839
Xu, Zongben I-39, I-63, II-1
Xue, Jiansheng III-939
Xue, Lihua III-813
Xue, Xiaoning III-827
Xue, Yanxue II-183
Xue, Yaofeng I-756, III-691

Yadav, Abhishek I-95
Yan, Gaowei III-674
Yan, Hong III-709
Yan, Lanfeng III-760
Yan, Shaoze III-155, III-271
Yan, Tianyun I-455
Yan, Xiangguo III-754
Yang, Guang II-124, II-165, II-424
Yang, Hongying III-622
Yang, Hu II-20
Yang, Huaqian II-860

Yang, Jiaben III-42
Yang, Jie I-609, II-673
Yang, Jinmin III-367
Yang, Ju III-439
Yang, Junyan III-857
Yang, Li III-1010
Yang, Luxi II-496
Yang, Pei I-621
Yang, Shangming II-891
Yang, Shuyuan I-385, I-893, II-562, II-723
Yang, Simon X. II-418, III-259
Yang, Tianqi III-451
Yang, Xianhui III-125
Yang, Xiaofan I-265
Yang, Xiaosong I-357
Yang, Xiaowei I-869
Yang, Xiaozong I-497
Yang, Xin II-624
Yang, Xu II-819
Yang, Xuehong III-1017
Yang, Xuhua III-7, III-463
Yang, Xusheng III-616
Yang, Yahong I-770
Yang, Yang I-646, I-875
Yang, Yingjie III-998
Yang, Yong III-488
Yang, Yongqing I-706, III-1022
Yang, Yujiu I-824
Yang, Yuru II-916
Yang, Zhi III-92
Yang, Zhichun I-187, I-241
Yao, Li III-725
Ye, Hao III-622
Ye, Mao I-1034, II-112
Ye, Meiying I-919
Yeh, Chunghsing II-238
Yen, Gary G. III-69
Yeşiloğlu, S. Murat III-210
Yi, Jianqiang I-552, I-640, III-48, III-801
Yi, Zhang I-585, I-739, I-967, I-1028, II-112, II-891
Yin, Chenbo III-514
Yin, Fuliang II-520, II-544, II-550, II-586, II-592, II-795
Yin, Hujun II-105
Yin, Jianping I-805
Yin, Xingbo III-297
Yin, Yixin I-509
Yin, Zhonghang III-204
Yoon, Younggui I-597, II-904
You, Yong III-616
Yu, Bo III-921
Yu, Chung III-987
Yu, Daoheng I-446
Yu, Ding-Li III-86, III-544, III-865
Yu, Dingwen III-544
Yu, Dongmei I-770
Yu, Gang II-690
Yu, Haibin III-521
Yu, Jianqiao I-149
Yu, Jinshou II-418
Yu, Jinxia III-265, III-285
Yu, Lun II-424
Yu, Miao III-890
Yu, Yibin III-827
Yu, Zhandong III-98
Yu, Zhenhua II-436
Yu, Zhezhou II-298
Yu, Zhiping I-479
Yuan, Dongfeng III-303, III-326, III-343
Yuan, Jingqi III-691
Yuan, Qingfeng III-851
Yuan, Senmiao I-526
Yuan, Zejian I-906
Yue, Bo III-609
Yue, Heng III-871

Zeng, Jiazhi I-677
Zeng, Xiangyan II-262
Zeng, Zhigang I-282
Zha, Daifeng III-742
Zha, Xu III-21
Zhai, Junyong III-36
Zhang, Anqing II-490
Zhang, Bin II-393
Zhang, Bo III-971
Zhang, Bofeng I-532
Zhang, Changjiang II-707
Zhang, Chunshu I-75
Zhang, Dafang III-367
Zhang, Dan II-630
Zhang, Daoqiang II-659
Zhang, David II-177
Zhang, Deyun I-931, III-385, III-482
Zhang, Erhu II-183
Zhang, Fan II-783
Zhang, Guicai III-603
Zhang, Guowei III-247
Zhang, Haixia III-303
Zhang, Hao II-222

Zhang, Haoran II-707
Zhang, Hemiao III-494
Zhang, Heng III-760
Zhang, Hongbin II-783
Zhang, Honghui III-890
Zhang, Hongying II-735
Zhang, Huaguang I-142, I-209
Zhang, Hui II-605, II-879
Zhang, Jie III-125, III-839
Zhang, Jun I-120
Zhang, Junying II-369
Zhang, Lejie III-277
Zhang, Li II-298
Zhang, Libiao II-298
Zhang, Liming I-985, II-14, II-651, II-759, II-836, III-198
Zhang, Ling I-440
Zhang, Linhua I-168, I-991, II-848, II-868
Zhang, Linke III-567
Zhang, Liping III-247
Zhang, Liqing II-514
Zhang, Qiang I-193
Zhang, Qizhi III-161
Zhang, Ran III-192
Zhang, Shengrui I-253, I-259
Zhang, Shuguang II-45, II-232
Zhang, Taiyi II-124
Zhang, Tengfei III-648
Zhang, Tianqi I-1022
Zhang, Tianxu II-316
Zhang, Wei I-149, II-124, II-860
Zhang, Wenrong III-332
Zhang, Xianda I-1010, II-216, II-526, II-538
Zhang, Xiangrong I-51, I-57
Zhang, Xiaohong I-351
Zhang, Xinhua II-387, II-490
Zhang, Xining III-508
Zhang, Xuegong III-291, III-908
Zhang, Xuxiu II-490
Zhang, Yaling I-718
Zhang, Yanyan I-788
Zhang, Yinping I-222
Zhang, Yong II-508
Zhang, Youyun III-857
Zhang, Yumin I-69, III-550
Zhang, Zheng III-247
Zhang, Zhenya II-45, II-232
Zhang, Zhifen III-470, III-1004
Zhang, Zhigang II-183
Zhang, Zhilin I-967
Zhang, Zhiyong I-591
Zhang, Zhizhou I-247
Zhang, Zhong I-126
Zhang, Zhongbo II-828
Zhang, Zhousuo I-937
Zhao, Dongbin I-640, III-48
Zhao, Feng I-869
Zhao, Fuqing I-770
Zhao, Gang III-561
Zhao, Hai I-887, II-39
Zhao, Hanqing III-185
Zhao, Hongchao III-150
Zhao, Jianye I-446
Zhao, Jidong I-677
Zhao, Li II-137
Zhao, Lijie III-167, III-1028
Zhao, Pengfei II-93
Zhao, Shengmei III-338
Zhao, Xiaojie III-725
Zhao, Yali I-552
Zhao, Yongrui II-765
Zhao, Zhenjiang I-132
Zheng, Baoyu III-338
Zheng, Chongxun II-327, III-731, III-754
Zheng, Chunhou II-478, II-807
Zheng, Gengsheng I-398
Zheng, Hongying III-434
Zheng, Kai III-155
Zheng, Nanning I-906, II-226
Zheng, Shiyou III-54
Zheng, Weimin I-700
Zheng, Weiyu I-398
Zheng, Zhenyu III-297
Zheng, Ziming II-112
Zhong, Shisheng I-473
Zhong, Yingji III-343
Zhou, Chunguang II-298, III-686, III-903
Zhou, Donghua III-79
Zhou, Gengui II-251
Zhou, Hao II-651
Zhou, Huajie III-790
Zhou, Jianzhong II-137
Zhou, Jin I-308
Zhou, Junlin II-63
Zhou, Qing III-1058
Zhou, Qingmin III-514
Zhou, Shangbo I-340, I-351
Zhou, Wei II-112
Zhou, Wengang III-686
Zhou, Xiaobo III-719

Zhou, Yali III-161
Zhou, Yi III-216
Zhou, Zhengzhong I-1022
Zhou, Zhihua III-790
Zhou, Zhijie III-591
Zhu, Anmin III-259
Zhu, Daqi I-706
Zhu, Feng II-216
Zhu, Hongcan II-251
Zhu, Jie III-921
Zhu, Jihong I-515, III-60, III-253
Zhu, Qin II-454
Zhu, Shijian III-567
Zhu, Wei I-241
Zhu, Weile II-568
Zhu, Xiaolong I-1010
Zhu, Yihua II-251
Zhu, Yuanxian III-686
Zhu, Zhelei III-609
Zhu, Zishu III-819
Zong, Chengqing III-977
Zong, Xiaojun I-235
Zou, Anmin III-277
Zou, Bin I-538
Zou, Cairong II-137
Zou, Xiaobing III-265, III-285
Zuo, Chun III-1010
Zuo, Ming J. I-800