Ivan Jordanov and Lakhmi C. Jain (Eds.)

Innovations in Intelligent Machines -3

Contemporary Achievements in Intelligent Systems

Springer

Editors
Dr. Ivan Jordanov
University of Portsmouth
UK

Prof. Lakhmi C. Jain
School of Electrical and Information Engineering
University of South Australia
Adelaide
South Australia
Australia

ISSN 1860-949X e-ISSN 1860-9503
ISBN 978-3-642-32176-4 e-ISBN 978-3-642-32177-1
DOI 10.1007/978-3-642-32177-1
Springer Heidelberg New York Dordrecht London

Library of Congress Control Number: 2012943844

Preface

This research volume is a continuation of our previous volumes on intelligent machines. We laid the foundation of intelligent machines in SCI Series Volume 70 by including the possible and successful applications of computational intelligence paradigms in machines for mimicking the human behaviour.

The present volume includes the recent advances in intelligent paradigms and innovative applications such as document processing, language translation, English academic writing, crawling system for web pages, web-page retrieval technique, aggregate k-Nearest Neighbour for answering queries, context-aware guide, recommendation system for museum, meta-learning environment, case-based reasoning approach for adaptive modelling in exploratory learning, discussion support system for understanding research papers, system for recommending e-Learning courses, community site for supporting multiple motor-skill development, community size estimation of internet forum, lightweight reprogramming for wireless sensor networks, adaptive traffic signal controller and virtual disaster simulation system.

Modern information technology relies on intelligent systems that can learn and reason over a range of knowledge hidden in available datasets. Achieving these objectives involve applying machine learning biology and nature inspired methods to inductively construct computational and mathematical models (that can use explicit or implicit human supervision) and gain insight in terms of patterns and relationships hidden into the datasets in hand. This 'insight' helps the intelligent systems to reason and learn and to use the extracted knowledge for prediction of trends and tendencies, for processes and products monitoring and control, for fault detection and diagnosing in a wide range of application areas.

The aim of this edited book is to promote current theoretical and application oriented Intelligent systems research (more specifically in the field of neural networks computing) and to present examples of experimental and real-world investigations that demonstrate contemporary achievements and advances in the area.

Leading researchers contribute articles presenting works from this multi-faceted and burgeoning area of research at both theoretical and application levels, covering a variety of topics related to intelligent systems.

This book is directed to engineers, scientists, researchers, professor and the undergraduate/postgraduate students who wish to explore the applications of intelligent paradigms further.

We are grateful to the authors and reviewers for their contribution and appreciate the assistance provided by the editorial team of Springer-Verlag.

Dr. Ivan Jordanov, University of Portsmouth, UK
Professor Lakhmi C. Jain, University of South Australia, Adelaide, Australia

Contents

Chapter 1
An Introduction to Contemporary Achievements in Intelligent Systems

Jeffrey W. Tweedale[1] and Ivan Jordanov[2]

[1] School of Electrical and Information Engineering
University of South Australia
Mawson Lakes Campus
South Australia SA 5095
Australia

[2] School of Computing
University of Portsmouth
Portland Building
Hampshire PO1 2UP
United Kingdom

1 Introduction

The term intelligent systems is used to describe the necessary level of performance required to achieve the system goals. Intelligence has been observed and scientifically categorized as a biological stimuli response mechanism that is provided to satisfy an intended activity.Intelligence considers cognitive aspects of human behaviour, such as perceiving, reasoning, planning, learning, communicating and innovation. As society evolved, innovative individuals invented tools to assist them in achieving better outcomes. Since the industrial revolution [1], science and mechanization have become central to many academic challenges, driving a paradigm shift from philosophy towards systems engineering techniques. This desire to improve mechanized systems created the need for improvements to automation processes. These achievements extend the pioneering efforts of others stimulating new research and developments [2]. Computational Intelligence (CI) has evolved over the past 60 years [3] with many new fields of study emerging to dissolve obstacles encountered. These attempts relate to efforts at personifying attributes of human behavior and knowledge processes within machines. The resulting *Machine Intelligence* [4, 5] efforts stimulated the study of Artificial Intelligence (AI) [6, 7] and led to the evolution of many contemporary techniques.

Agent technologies, and in particular agent teaming, are increasingly being used to aid in the design of *intelligent systems* [8]. Software engineer have made significant progress in fields, such as knowledge representation, inference, machine learning, vision and robotics [9]. Minsky suggested that AI is the science relating to making machines do things that would be done by a human [10]. AI researchers originally studied science relating to human and animal intelligence. These concepts were initially conceived by Newell and Simon using production systems [11]; however, the study quickly divided into two streams with John McCarthy and Nil Nillson considered the Neats (using formal logic as a central tool to achieving AI, while Marvin Minsky and Roger Schanks where considered the scrufs (using a psychological approach to AI). Russel and Norvig entered the argument by describing an environment as something that provides input and receives output, using sensors as inputs to a program, producing outputs as a result of acting on something within that program. The AI community now uses this notion as the basis of definition of an agent [12]. AI technology is becoming the paradigm of choice for the development of complex distributed systems and as the natural progression from pure object oriented programming. Learning has an important role to play in both cooperative and autonomous systems. Agents with predefined behaviors based on a priori knowledge of the system that is modified using feedback from experience will continue to mature. Rather than having purely agent-based applications, we then have cooperative applications involving teams of agents and humans. AI will retain their architectural foundations but the availability of more appropriate reasoning models and better design methodologies will see them being increasingly used in mainstream software development. Furthermore, better support for human-agent teams will see the development of a new class of intelligent decision support applications. For more information, see the evolution of intelligent agents within the world wide web [13] and the ability to embedded automation into modern intelligent systems in a *Human-Agent* environment [14].

This book examines the basic concepts relating to contemporary in *Intelligent Systems* and provides a number of case studies to examine specific examples. Prior to examining those topics, the basic terms are defined to provide clarity and additional references where the reader chooses to seek more information. The following section progressively build the concepts associated with *Intelligent Systems* and some of the techniques used to create them. Hence the discussion review the definition of a system, intelligence, AI and associated intelligent paradigms.

1.1 What Is a System?

A system is defined as "a set of things working together as parts of a mechanism or an interconnecting network" [15]. Other dictionaries qualify the definition based on categories or functionality. For instance, the *American Heritage* dictionary defines a *system* as [16]:

– A group of interacting, interrelated, or interdependent elements forming a complex whole
– A functionally related group of elements, especially:
  - The human body regarded as a functional physiological unit.
  - An organism as a whole, especially with regard to its vital processes or functions.
  - A group of physiologically or anatomically complementary organs or parts. Examples include the nervous system and the skeletal system.
  - A group of interacting mechanical or electrical components.

  - A network of structures and channels, as for communication, travel, or distribution.
  - A network of related computer software, hardware, and data transmission devices.
- An organized set of interrelated ideas or principles.
- A social, economic, or political organizational form.
- A naturally occurring group of objects or phenomena. For example, the solar system.
- A set of objects or phenomena grouped together for classification or analysis.
- A condition of harmonious, orderly interaction.
- An organized and coordinated method; a procedure.
- The prevailing social order; the establishment.

This book discusses the concept of a *system* in relation to intelligent systems using contemporary achievements. The focus is on using software to enhance machine intelligence to control robotic behavior. Examples provided are associated with AI techniques aimed at harnessing knowledge from artifacts collected within the environment or learning algorithms using neural networks.

1.2 What Is Intelligence?

By definition, intelligence is "the ability to acquire and apply knowledge and skills" [15]. The level of *intelligence* grades the ability to think, postulate or even compose a thesis to a solution. This measure examines many skills, compliance to lots of rules and a significant level of expert knowledge in the problem Domain. We measure human intelligence using a variety of Intelligence Quotient (IQ) tests. This measures a wide range of skills to generate a rating using a common standard. Psychologist can administer *professionally engineered* tests, such as the Wechsler Adult Intelligence Scale (WAIS)[1] and the Ravens Progressive Matrices[2]. What is measured and its relevance is the major issue clouding the terminology. Computers appear to achieve intelligent feats, but they are not intelligent, they are merely doing what they are programmed to do!

Intelligence has been defined in several ways [17]:

- the ability to learn or understand from experience,
- ability to acquire and retain knowledge,
- mental ability,
- the ability to respond quickly and successfully to a new situation, and
- use of the faculty of reason in solving problems or directing conduct effectively.

In this book, the concept of intelligence is associated with techniques used to enhance a machines ability to achieve tasks that are currently not supported by existing systems. Research into intelligent systems is evolving, however no scale exists to measure or

[1] See wilderdom.com for more information.

[2] See www.pearsonassessments.com

compare machine intelligence because of the complexity required to normalize the results. Ongoing research into automation may develop an machine intelligence quotient[3], until then, the focus remains on improving the techniques associated with collecting information and generating knowledge.

1.3 What Is an Intelligent System?

An intelligent system is therefore a collective set of techniques that create a mechanism to achieve temporary objectives within a limited space and time using information sensed within its environment. The system uses concept provided or divined in response to its current state and perceived situation based on the constraints of its programming. There is no clear definition, although society is forming a belief that an intelligent system should be a machine that is capable of simulating the human decision making process (exhibit a basic IQ). In reality intelligent machines remain formal or informal systems that gather or manage data, to obtain and process information into knowledge to enable reasoned judgments by human decision makers. The term is not limited to intelligence organizations or services but includes any system, in all its parts, that accomplishes the listed tasks [18].

The concept of embedding intelligence within machines has existed since the first AI conference held at Dartmouth in 1956 [19]. Software has traditionally been used to monitor systems without providing significant direction or control. Machines and production lines are still controlled by operators that require specified skills to achieve predetermined goals. When the required stimuli is missing or delayed, that machine or process become disrupted and may fail. The efficiency of attaining a goal, should not be confused with the intelligence of the machine or operator. Automation is the incarnation of a known sequence of series of processes that contribute to a predefined task. This concept should not be interpreted as intelligence, regardless of the level of technology or efficiency it provides [14]. Applications are increasingly being developed using AI techniques to automate functions traditionally conducted by the human element within the overall system. At present, many of these systems retian one or more humans in the loop, however they are being displaced are more advanced AI techniques evolve.

1.4 What Is AI?

The term AI was born out of a conference held at Dartmouth in 1956 [19] and generally attributed to John McCarthy [3]. He was latter acknowledged as the father of AI and has since reported he should have used the term CI [20]. The concept of AI at that time was documented by Newell and Simon [21, 22] who highlighted their production systems [11] in those examples. The field soon divided into two streams with John McCarthy and Nils Nilsson considered the *Neats*[4], while Marvin Minsky and Roger Schanks where considered the *scrufs*[5]. Minsky has since told reporters in 1982 that,

[3] It would be appropriate to use the term Machine Quotient (MQ) in lieu of machine intelligence quotient because it would reflect the personified association with humans.

[4] Who started using science and formal logic as a central tool to achieving weak AI.

[5] Who retained a cognitive or psychological approach to strong AI.

"AI has one foot in science and one in engineering" [23]. Where as Roszak stated "AIs record of barefaced public deception is unparalleled in annals of academic study" [24]. The definition has become a heated debate, so now we focus on the terms *strong* AI and *weak* AI.

The goal of *strong* AI is to build a machine that is capable of thought, consciousness and emotion (the mind), where *weak* AI merely develops models to test theories about understanding humans and animals, usually in robotic form. These models provides useful tools that help us understand the mind. We know that AI bounds a number of disciplines that include: psychology, philosophy, linguistics and neuroscience.

Psychology: The *Pavlov's dog* demonstrated how to observe behavior, through the study of stimulus and response. Fredholm determined there was a valid set of concepts that explained memory, learning and reasoning [25]. The study of intelligence in AI exhibits this interdisciplinary approach to cognitive psychology.

Philosophy: Simon introduced the theory on computational intelligence. In 1957 he suggested that "within ten years, that most psychological theories would take the form of computer programs" [26]. Descartes introduced the concept of *dualism* where he argued there was a fundamental difference between the mental realm and the physical realm [27]. There is a parallel about the computer program requiring a computer to manifest itself and the mind requiring a brain to exist. This attempts to embody knowledge into machines makes several assumptions regarding ontology and hermeneutics, such as *the sorts of things machines need to know*. There are varying degrees of intelligence in humans, in machines this relies on the computation engine. Walter Grey constructed several robots with a number of sensors for collision detection. *Elsie* wandered autonomously until her battery level fell[6] and provided the illusion of mimicking what appeared to be complex behavior [28]. Similarly, Wilhelm von Osten claimed to train a horse to do maths, albeit gesture driven behavior. Both examples highlight the problems associated with determining an agents capacity solely on behavior.

Linguistics: Chomsky had a predisposition for language and believed the human competency for speech was only shaped by their environment (noting we are born with some knowledge of language).

Neuroscience: This concept forms the foundation of most high-level cognitive processes used in AI.

1.5 Putting AI to Work

In 1966 a robot called *shaky* was built at Stanford [29]. He combined a number of AI techniques to assist him organising blocks within a simple constrained environment.

[6] At which point, she was attracted to a light positioned inside a charging tunnel. When fully charged, she repeatedly continued wandering.

Wheel slippage introduced errors into the *sense, model, plan* and *act* cycle which disrupted perception and positioning within the environment. Unfortunately as the environment became more complex the techniques in this model eventually failed [30]. As AI matured into the modern era, McClelland introduced the connectionist theory which quickly theory gained popularity [31].

McCulloch and Pitts presented the concept of a Neural Networks (NN) by making many parallels to the brain on a highly abstracted level. One obvious difference is the brain only processes information at a rate of approximately 1000 signals per second. It does however conduct millions, possibly billions, of parallel processes simultaneously[7]. The brain has approximately 100 billion neurons with at least 100 trillion synapses interconnected by axons using multiple microtubules. Each neuron behaves like a single processor, which itself runs an application one instruction at a time (serially). It could take billions of instructions to recognise one shape, pattern or image. We are also aware that the brain organises symbols in a hierarchical order based on the frequency of memories or association. This means that *machine learning* must encompass techniques from both the *symbolic* and *connectionist* branches of AI. In symbolic representation the information used for comparison is generally stored within a central or locatable package. We know in the real world, much of that information is distributed, therefore connections are required to enable information to be *evoked*. Literate discussing innovative examples using AI techniques within multi-agent systems is available [32].

1.6 New AI

It was Gregory Bateson who ushered in a new set of principles when he said: "What thinks is a brain in a human being who is part of a system that includes an environment" [33]. Some use the term, *new AI*. If we follow the disembodied theory, we should concentrate on autonomous agent behavior in the everyday world (free of situation) working from the bottom up. The higher-level functions, like knowledge and reasoning should still be approached top-down. Brooks showed this concept using a robot called *genghis* that had fifty-one parallel programs (sub-systems) [34]. Again Luc Steels demonstrated the concept of a shared *lexicon* in agency theory via his *talking heads* experiments [35].

1.7 Intelligent Paradigms

A number of *Intelligent Paradigm* techniques are reported in literature. These include; decision-trees, rule-based, Bayesian, rough sets, dependency networks, reinforcement learning, Support Vector Machines (SVM), NNs, genetic algorithms, evolutionary algorithms and swarm intelligence. Many of these topics are covered in this book. An example of intelligence is to use AI search algorithms to create automated macros or templates [36]. Again Generic Algorithm (GA) can be employed to induce rules using rough sets or numerical data. A simple search on data mining will reveal numerous paradigms, many of which are intelligent. The scale of search escalates with the volume

[7] In comparison the Intel i7-990X Extreme *Gulftown* has six independent cores all running at 3.73 GHz peak (or 12 hyper-threads).

of data, hence the reason to model data. As data becomes ubiquitous, there is increasing pressure to provide an on-line presence to enable access to public information repositories and warehouses. Industry is also increasingly providing access to certain types of information using kiosks or paid web services. Data warehousing commonly uses the following steps to model information:

- data extraction,
- data cleansing,
- modeling data,
- applying data mining algorithm,
- pattern discovery, and
- data visualization.

Any number of paradigms are used to mine data and visualize queries. For instance, the popular *six-sigma* approach (define, measure, analyse, improve and control) is used to eliminate defects, waste and quality issues. An alternative is the SEMMA (sample, explore, modify, model and Assess). Other intelligent techniques are also commonly employed. Although we don't provide a definitive list of such techniques, this book focuses on many of the most recent paradigms being developed, such as Bayesian analysis, SVMs and learning techniques.

1.8 Knowledge

Information, knowledge and wisdom are labels commonly applied to the way humans aggregate practical experience into an organized collection of facts. Knowledge is considered a collection of facts, truths, or principles resulting from a study or investigation. The concept of knowledge is a collection of facts, principles, and related concepts. Knowledge representation is the key to any communication language and a fundamental issue in AI. The way knowledge is represented and expressed has to be meaningful so that the communicating entities can grasp the concept of the knowledge transmitted among them. This requires a good technique to represent knowledge. In computers symbols (numbers and characters) are used to store and manipulate the knowledge. There are different approaches for storing the knowledge because there are different kinds of knowledge such as facts, rules, relationships, and so on. Some popular approaches for storing knowledge in computers include procedural, relational, and hierarchical representations. Other forms of knowledge representation used include *Predicate Logic, Frames, Semantic Nets, If-Then rules and Knowledge Inter-change Format*. The type of knowledge representation to be used depends on the AI application and the domain that Intelligent Agents (IAs) are required to function. [37]. Knowledge should be separated from the procedural algorithms in order to simplify knowledge modification and processing. For an IA to be capable of solving problems at different levels of abstraction, knowledge should be presented in the form of frames or semantic nets that can show the *is-a* relationship of objects and concepts. If an IA is required to find the solution from the existing data, Predicate logic using IF-THEN rules, Bayesian or any number of techniques can be used to cluster information [14].

1.9 Other Effects

Cognitive Science is a field of research attracting significant effort. It was preceded by the process management evolution with many prominent achievements, such as, Taylors introduction to Scientific Management and the Hawthorn Experiments conducted by the National Research Council (NRC). Formalizing organizational systems and behavioral science provides the tools required to decompose human oriented task. Any real-world system takes inputs as sensors will only react appropriately when it is able to modify the outputs. Simulation models rely on the same approach. Agents can be used to monitor sensors and stimulate the decision making required to modify one or more outputs.

Apollo 13 experienced a quintuple fault which required an army of ground crew the challenge of remotely assessing the status/health of the spacecraft prior to rapidly redesigning a new mission plan with revised procedures. The ultimate decision compromised the original mission goal of landing on the moon which was quickly revised to a successful return to earth. The ground crew were required to search for a new unintended reconfiguration of the space crafts subsystems with the required procedures required to effect those changes manually.

Agent technologies, and in particular agent teaming, are increasingly being used to aid in the design of "intelligent systems" [38]. In the majority of the agent-based software currently being produced, the structure of agent teams have been reliant on structures defined by the programmer or software engineer. The development of a model that extends the communications architecture of an agent framework that is adaptable when contacting a series of Multi-Agent System (MAS) or teams. The ideal properties of agents, includes: deliberative agents, reactive agents, interface agents (HCI) and mobile agents [26]. Different systems may be instantiated with a variety of hierarchies, with each level performing predetermined tasks in a subordinate or supervisory role. An Agent Architecture is considered to include at least one agent that is independent or a reactive/proactive entity and conceptually contains functions required for perception, reasoning and decision. The architecture specifies how the various parts of an agent can be assembled to accomplish a specific set of actions to achieve the systems goals. Wooldridge believes that it is essential for an agent to have "the ability to interact with other agents via some communication language" [39].

Research on agents requires the formation of teams of agents in order to dynamically configure the team with the ability to solve the decomposed task of the goal presented. Traditionally all tasks must be completed successfully or the team fails the goal [8]. A dynamic architecture would substitute agents within the team with alternative capabilities in order to succeed. It may even compromise and offer a partial solution and offer it to another system to complete. A good communications framework is required to pass messages between separate agent and other systems. An IA frameworks has recently been extended within the Knowledge-Based Intelligent Information and Engineering Systems (KES) Centre to enable individual students to successfully fast track the development of their research concepts. A Plug n Play concept based on a multi-agent blackboard architecture forms the basis of this research. The authors believe that a core architecture is required for MAS developers to achieve flexibility. Current research focuses on how agents can be teamed to provide the ability to adapt and dynamically organize the required functionality to automate in a team environment. The model is conceptual and is proposed initially as a blackboard model, where each element represents a block of functionality required to automate a process in order to complete a specific task. Discussion is limited to the formative work within the foundation layers of that framework.

1.10 Why Agents?

As stated previously, agents are increasing being used to solve progressively more complex problems. As we approach applications that solve real-world problems, the skills required have risen dramatically. Practical examples of where agents are currently used, include: spell checking, spam filters, travel and event booking systems. The code required to create an agent factory will focus on the needs of the programmer. This code could be linked at compile-time, but more preferably, instantiated and attached during run-time. In order to abstract the inherent complexity of this task, a factory is required. It needs functionality that dynamically wraps and packages a given capability. Intelligent systems can be developed to modify existing code, even when in memory, without loss of state or downtime. Agents are typically used in this process.

2 Chapters Included in the Book

This book includes nine chapters. The following section contains a summary of each topic.

Chapter 2: Market Power Assessment Using Hybrid Fuzzy Neural Network

This research discusses market power assessment as an important aspect of electric market analysis and operation. It proposes a multi output fuzzy neural network (FNN) for market power assessment and for finding on line market power ranking status of the generating company in a competitive power system, using a fuzzy composite market index (FCMI). This index is formulated by combining a Lerner index, a Relative market power and a Nodal Cost indices. In the proposed FNN, a trained multi-output neural network is used as a fuzzy inference engine. The input of the FNN consists of real loads and a bipolar code to represent a trading interval, while the output consists of the fuzzy values of the FCMI. A number of training patterns covering the full operating range of the power system are generated using the system data (such as offer prices and operating constraints) in order to train the FNN. The determined optimal power flow results are used to compute the above three market power indices and the corresponding FCMI. The performance of the proposed method is tested on an IEEE 14 bus system for 20 testing trading periods. The obtained results can be directly uploaded to an open access on-line information system (or to a dedicated web site), so that an independent system operator or customers can make use of this valuable information.

Chapter 3: Coaching Robots: Online Behavior Learning from Human Subjective Feedback

Efforts to coach robots resulted in an a novel methodology to create an agent learning behavior that incorporates both interactive and iterative approaches. The method is called Coaching and allows human trainer to give a subjective evaluation of the robotic agent in real time and the agent can simultaneously update the reward function. The research demonstrates agents capability of learning the desired behavior by receiving simple and subjective instructions (positive and negative) by implementing Coaching framework of typical reinforcement learning. The proposed approach is also effective when it is difficult to determine in advance a suitable reward function for the learning situation. The validation and verification of the investigated method advantages are done through conducting several experiments involving simulated and a real robot arm systems.

Chapter 4: Persian Vowel Recognition Using the Combination of Haar Wavelet and Neural Network

This chapter studies Lip image localization and segmentation as part of lips movement analysis which has significant role in speech recognition. Even after detecting the lips there are still major problems that any vowel (especially Persian) recognition method is faced with, such as: low chromatic in lip region; low contrast luminance; overlap between the lip and facial skin color; and similarity between lips movement in some vowels. A new, automatic and fast approach for the lip extraction based on using the Haar wavelet is proposed here and its output is used as a input feature vector for a hybrid neural network. The proposed algorithm uses the CIE L*U*V* and CIE L*a*b* color spaces in order to improve the contrast between the lip and the other face regions. Subsequently, the lips are modeled and a feature vector with longitudinal and angular parameters is extracted and used as an input for a feedforward backpropagation hybrid neural network. The proposed method is tested on 2200 images and the obtained accuracy shows about 15% improvement when compared with similar methods.

Chapter 5: The Reproduction of the Physiological Behaviour of the Axon of Nervous Cells by Means of Finite Element Models

This research investigates 3D Finite Element modeling solutions for a segment of a nervous cell axon, which take into account the non linear and time varying dynamics of the membrane surrounding it, in order to reproduce its physiological behavior in terms of Action Potentials elicitation and its temperature dependence. A combination of the so called Hodgkin-Huxley equations modeling the dynamics of the membrane voltage-controlled ionic channels, together with the Maxwell equations in Electro Quasi-Static approximation describing the electromagnetic behavior of each medium, is tackled in a numerical procedure implemented in a commercial Finite Elements multi-physical environment. Two different models are investigated: the first one exploits typical thin layer approximation for the axon membrane, proving to be useful when the field solution inside the membrane domain is not of interest; and in the second model the axon membrane is considered a non-linear active medium (exploiting its equivalent electric conductivity), allowing also reproduction of the electric potential inside the membrane, which is more realistic representation of the studied system. Although theoretical, this chapter presents models that open a wide range of applications and extensions in order to understand the true behavior of a complete neuron.

Chapter 6: A Study of a Single Multiplicative Neuron (SMN) Model for Software Reliability Prediction

This is an application oriented investigation that studies the use of a single multiplicative neuron model for prediction of cumulative faults of software. Standard back propagation (BP) and real coded genetic algorithm (GA) with mean squared error as a fitness function are employed for the model parameters optimisation. The performance of the proposed model is tested and evaluated on two real data sets and the obtained results are compared with existing parametric software reliability models, showing the superiority of the investigated SMN model (for both BP and GA training). The advantages of the model are based on its easy applicability on wide range of software failure data and its computational efficiency resulting from simplified NN architecture (no hidden layers).

Chapter 7: Numerical Treatment for Painlev Equation I Using Neural Networks and Stochastic Solvers

This study investigates theoretically and proposes a new stochastic numerical method for solving Painlev I equation. The mathematical model of the equation is formulated with feed-forward artificial neural networks. Linear combination of the networks defines the unsupervised error for the equation. For the networks training genetic algorithm, simulating annealing and pattern search algorithms hybridized with interior point algorithm for rapid local search are implemented and compared. The reliability and the effectiveness of the discussed approach are validated through statistical analysis. Comparison with standard approximate analytic solvers of the equation shows that the proposed stochastic solvers not only produce reliable and effective solutions, but are also superior for larger inputs.

Chapter 8: An Investigation into the Adaptive Capacity of Recurrent Neural Networks

Typical characteristic of any intelligent system is its ability to appropriately adjust its behaviour or modify its structure in response to environmental change. Feed-forward neural networks have commonly been used to model such behaviours. However, the weights of feed-forward neural networks remain static once trained and so can hardly be categorised as adaptive. On the contrary, recurrent networks (RNN) have the capability to exhibit dynamic behaviour having, in general, feedback connections after the applied non-linear activation function. In this work, network architectures with different feedback connections made before and after the non-linear activation function are studied in order to investigate their adaptive capabilities. Backpropagation training algorithms are applied to these networks with a minimum number of recurrent neurons at which adaptive behaviour is attainable. Three benchmark problems are investigated to analyse the performance and the learning ability of the proposed RNN architectures, demonstrating better performances for architectures with feedback connections before the nonlinear activation function when compared with feedback applied after the nonlinear function. These results can be very useful in designing RNN applications for a variety of problems.

Chapter 9: An Extended Approach of a Two-Stage Evolutionary Algorithm in Artificial Neural Networks for Multiclassification Tasks

This chapter studies a modified algorithm which operates with evolutionary artificial neural networks (EANN) to add broader diversity at the beginning of the evolutionary process and extends it to EANN with sigmoidal units. A simultaneous evolution of the investigated architectures and weights is performed with a two-stage evolutionary algorithm. The proposed methodology operates with two initial populations, each one containing individuals with different topologies which are evolved for a small number of generations. At this point half of the best individuals from each population are selected and combined to constitute a single population and the whole evolutionary cycle is applied to this new population. This idea was previously proposed by the authors for product unit neural networks and here it is extended to sigmoidal neural networks. The simulation, testing and validation of the proposed approach is carried out on twelve data sets from the UCI repository and two complex real-world problems that differ in their number of instances, features and classes. The results are analysed using nonparametric statistical tests to show significantly improved accuracy of the proposed models when compared with a standard methodology based on a single population. Moreover, the new methodology proves to be much more efficient than the previously developed by the authors a two-stage algorithm in evolutionary product unit neural networks.

3 Conclusion

An introduction to Contemporary Achievements in Intelligent Systems is provided to orient the reader and define the terminology used in this book. The following collection of case studies highlights the research contributions of many leading subject matter experts in the field of intelligent systems. This book is intended for students, professionals and academics from all disciplines to enable them the opportunity to engage in the state of art developments in:

- **Fuzzy Neural Network:** for commercial Power Assessment;
- **Behavior Learning:** Coaching Robots using On-line Behavior Learning from Human Subjective Feedback;
- **Haar Wavelet and Neural Network:** using Persian Vowel Recognition;
- **Software Reliability Prediction:** A Study of a Single Multiplicative Neuron (SMN) Model;

- **Finite Element Models:** to Reproduce the Physiological Behavior of The Axon of Nervous Cells;
- **Neural Networks and Stochastic Solvers:** Numerical Treatment for Painlev Equation I;
- **evolutionary algorithm in NN:** An extended approach using artificial neural networks for multi-classification tasks; and
- **Recurrent Neural Networks:** An Investigation into their Adaptive Capacity.

Readers are invited to contact individual authors to engage with further discussion or dialog on each topic.

References

1. Hudson, P.: The Industrial Revolution. Oxford University Press, Carey (1992)
2. Leedy, D.P., Ormrod, J.E.: Practical Research: Planning and Design, 8th edn. Person Press, New Jersey (2001)
3. Andresen, S.L.: John McCarthy: Father of AI. IEEE Intelligent Systems 17, 84–85 (2002)
4. Friedberg, R.M.: A learning machine: part I. IBM J. Res. Dev. 2, 2–13 (1958)
5. Friedberg, R.M., Dunham, B., North, J.H.: A learning machine: part II. IBM J. Res. Dev. 3, 282–287 (1959)
6. Minsky, M.: Heuristic aspects of the artificial intelligence problem. Lincoln Laboratory Report, Federal Scientific and Technical Information, Dept. of Commerce, Washington, pp. 34–55 (1956)
7. Russel, S., Norvig, P. (eds.): Artificial Intelligence: A Modern Approach, 2nd edn. Prentice Hall Series in Artificial Intelligence. Prentice Hall (2003)
8. Wooldridge, M., Jennings, N.R.: The cooperative problem-solving process. Journal of Logic and Computation 9, 563–592 (1999)
9. Grevier, D.: AI–The Tumultuous History of the Search for Artificial Intelligence. Basic Books, New York (1993)
10. Minsky, M.: Society of Mind. Simon and Schuster, Pymble (1985)
11. Thagard, P.R.: Computational Philiosphy of Science. MIT Press (1993)
12. Franklin, S., Graesser, A.: Is it an agent, or just a program?: A taxonomy for autonomous agents. In: Proceedings of the Third International Workshop on Agent Theories, Architectures and Languages, Budapest, Hungary, pp. 193–206 (1996)
13. Tweedale, J., Jain, L.: The evolution of intelligent agents within the world wide web. In: Nguyen, N., Jain, L. (eds.) Intelligent Agents in the Evolution of Web and Applications. SCI, vol. 167, pp. 1–9. Springer, Heidelberg (2009)
14. Tweedale, J., Jain, L.C.: Embedded Automation in Human-Agent Environment. Adaptation, Learning, and Optimization, vol. 10. Springer, Heidelberg (2011)
15. Soanes, C., Stevenson, A.: Concise Oxford English dictionary. Oxford University Press, New York (2004)
16. Harcourt, A., Brace, D. (eds.): The American Heritage Dictionary of the English Language, 5th edn. Houghton Mifflin, Boston (2011)
17. Krishnakumar, K.: Intelligent systems for aerospace engineering - an overview. Technical Report ADA484100, NASA AMES Research Center, Mountain View, CA (2003)
18. J7, J.D.D. (ed.): DoD Dictionary of Military and Associated Terms. Number JP 102, US Joint Staff, Washington DC (June 2003)
19. McCarthy, J.: Programs with common sense. In: Symposium on Mechanization of Thought Processes, Teddington, England. National Physical Laboratory (1958)

20. McCorduck, P.: Machines who think, pp. 1–375. Freeman, San Francisco (1979)
21. Newell, A., Simon, H.A.: Human Problem Solving. Prentice-Hall, Englewood Cliffs (1972)
22. Newell, A.: Production systems: Models of control structure. In: Chase, W.G. (ed.) Visual and Information Processing, pp. 463–526. Academic Press, San Diego (1973)
23. French, R.M.: The chinese room: Just say "no!". In: Gleitman, L.R., Joshi, A.K. (eds.) 22nd Annual Cognitive Science Society Conference. Institute of Research and Cognitive Science, pp. 657–662. Lawrence Erlbaum Assoc., NJ (2000)
24. Vaux, J., Dale, R.: Review of "mind over machine". In: AI & Society, vol. 1(1), pp. 72–76. Springer, New York (1987)
25. Fredholm, L.: Pavlov's Dog. Nobel Media, Stockholm (2001)
26. Ericsson, K.A.: The Cambridge handbook of expertise and expert performance. Cambridge University Press, New York (2006)
27. Descartes, R.: Meditation vi. In: Cottingham, J. (ed.) Meditations on the First Philosophy (Translated 1986). Cambridge University Press (1641)
28. Grey, W.W.: The Living Brain. Duckworth (1953)
29. Rosen, C.A., NiIsson, N.J., Adams, M.B.: A research and development program in applications of intelligent automata to reconnaissance. Proposal for Research ESU 65-1, Stanford Research Institute, Menlo Park, California (1965)
30. Raphael, B.: Robot research at stanordresearch institute. Technical Note 64, Stanford Research Institute, Menlo Park, California (1972)
31. Hayward, M.: A connectionist model of poetic meter. In: Dowd, T., Janssen, S. (eds.) Poetics, vol. 20(4), pp. 303–317. Elsevier Press, New York (1991)
32. Tweedale, J., Ichalkaranje, N., Sioutis, C., Jarvis, B., Consoli, A., Phillips-Wren, G.: Innovations in multi-agent systems. Journal of Network Computing Applications 30(3), 1089–1115 (2007)
33. Bateson, G.: Steps to an Ecology of Mind. University of Chicago Press, Chicago (1972)
34. Brooks, R.A.: A robot that walks; emergent behaviors from a carefully evolved network. Technical Report 1091, MIT Artificial Intelligence Laboratory (1989)
35. Steels, L., Kaplan, F.: Bootstrapping grounded word semantics. In: Briscoe, T. (ed.) Linguistic Evolution Through Language Acquisition: Formal and Computational Models, pp. 53–73. Cambdrige University Press, Cambridge (2002)
36. Lin, T., Xie, Y., Wasilewska, A., Liau, C.J. (eds.): Data Mining: Foundations and Practice. SCI, vol. 118. Springer, New York (2008)
37. Bigus, J.P., Bigus, J.: Constructing Intelligent Agents Using Java. Professional Developer's Guide Series. John Wiley & Sons, Inc., New York (2001)
38. Urlings, P.: Teaming Human and Machine: A conceptual framework for automation from an aeronautical perspective. PhD thesis, University of South Australia, School of Electrical and Information Engineering (2004)
39. Wooldridge, M., Jennings, N.R.: Theories, Architectures, and Languages: A Survey. In: Wooldridge, M.J., Jennings, N.R. (eds.) ECAI 1994 and ATAL 1994. LNCS (LNAI), vol. 890, pp. 1–39. Springer, Heidelberg (1995)

Chapter 2
Market Power Assessment Using Hybrid Fuzzy Neural Network

Kirti Pal[1], Manjaree Pandit[2], and Laxmi Srivastava[2]

[1] Department of Electrical Engineering, RGGI, Meerut
kirtiglory@yahoo.co.in
[2] Department of Electrical Engineering, MITS, Gwalior, India
{manjaree_p,srivastaval}@hotmail.com

Abstract. Market power assessment is an important aspect of electric market analysis and operation. Market power problems are more complicated in an electric market than those in other markets due to the specific properties of electricity. A comprehensive and dynamic market power assessment has been proposed in this paper to protect and improve the open electricity market. This paper proposes a multi output fuzzy neural network (FNN) for market power assessment and for finding the on line market power ranking status of GENCOS in a competitive power system using a fuzzy composite market index (FCMI). This index is formulated by combining (i) Lerner Index, (ii) Relative market power and (iii) Nodal Cost. In the proposed FNN a trained multi-output neural network is being used as a fuzzy inference engine. The input of FNN consists of real loads and a bipolar code to represent a trading interval while the output consists of the fuzzy values of FCMI. To train the FNN a number of training patterns, covering the full operating range of the power system, are generated using the system data such as offer prices and operating constraints. OPF results are used to compute the above three market power indices and the corresponding FCMI. Once the network is trained it is capable of predicting the FCMI values in five fuzzy classes (GENCO ranking) for any given operating scenario, on line, instantaneously, without bothering about the computational burden of OPF. The computational effort is required only for training the network which is an off line process. Since the training of ANN is extremely fast and test results are accurate, they can be directly floated to OASIS (open access same time information system) and any other web site. An Independent system operator(ISO) and customers can access this information instantly. The performance of the proposed method has been tested on an IEEE 14 bus system.

Index Terms: Generator Market Share (GMS), Lerner Index (LI), market power, Must Run Ratio (MRR), open electricity market, Relative Market Power (RMP), transmission congestion.

1 Nomenclature

FNN	Fuzzy neural network
ISO	Independent system operator

GENCO	Generating Company
LI	Lerner index
MRR	Must run ratio
MRS	Must run share
RMP	Relative market power
NC	Nodal cost
FCMI	Fuzzy composite market index
OPF	Optimal power flow
GMS	Generator market share
μ	Membership value of the class to which GENCO belongs
W	Weighting factor
x_i	Input variable
ϕ_{ik}	Input variable fuzzy membership function for x_i corresponding to the data point *k*.
OASIS	Open access same time information system
ANN	Artificial neural network

2 Introduction

Market power is defined as the ability to alter profitably prices away from competitive level. Market power can be exercised either by withholding the quantity of commodity or by raising the asking price above the competitive price level without affecting the demand of the commodity. In power systems, transmission network provides the infrastructure to support a competitive electricity market, but congestion occurs frequently in weakly connected networks. In a competitive electricity market, the oligopoly structure of the market and the network constraints may produce results far from the perfect competition.

One of the main objectives in the market monitoring process is the analysis of market power issues. The path toward liberalization has been under taken under the belief that the competition would strive for market efficiency [1] and price reduction resembling to the microeconomic model of perfect competition in which the social welfare would be the highest possible and the price will be the lowest. Unfortunately, different reasons may lead the market far from this desirable result. Some papers focus on the congestion impacts also in presence of the demand elasticity representation and the reactive load modeling [2], [3], [4], and provide methods to alleviate congestion impacts. In [5], the transmission congestion cost and locational marginal prices are considered, while in [6], thermal voltage and stability limits are considered to represent the feasibility region for the system. Strategic bidding has been extensively considered according to different approaches such as statically approaches [7], [8], parametric dynamic programming [9], Lagrange relaxation [10], genetic algorithm [11], stochastic procedure [12], fuzzy set theory [13], and game theory [14], [15]. In [16], the oligopolistic competition is examined in the submarkets that are isolated by constrained transmission lines.

The primary objective of this paper is to explore the potential for using an engineering approach to measure the existence of market power in the real time operations of a power grid.

An ISO requires the bid prices of GENCOS to run the OPF. For measuring market power an ISO solves an Optimal Power Flow (OPF) to determine the least cost pattern of dispatch based on the available offers in a uniform price auction. The OPF is determined subject to physical constraints on the power grid, such as thermal limits on transmission lines, and operating constraints, such as maintaining voltage levels.

This chapterproposes a multi output fuzzy neural network (FNN) for market power assessment and for finding the on line market power ranking status of GENCOS in a competitive power system using a fuzzy composite market index (FCMI). This index is formulated by combining (i) Lerner Index, (ii) Relative market power and (iii) Nodal Cost. In the proposed FNN a trained multi-output neural network is being used as a fuzzy inference engine. The input of FNN consists of real loads and a bipolar code to represent a trading interval while the output consists of the fuzzy values of FCMI. To train the FNN a number of training patterns, covering the full operating range of the power system, are generated using the system data such as offer prices and operating constraints. OPF results are used to compute the above three market power indices and the corresponding FCMI. Once the network is trained it is capable of predicting the FCMI values in five fuzzy classes (GENCO ranking) for any given operating scenario, on line, instantaneously, without bothering about the computational burden of OPF. The computational effort is required only for training the network which is an off line process. Since the training of ANN is extremely fast and test results are accurate, they can be directly floated to OASIS (open access same time information system) and any other web site. The ISO and customers can access this information instantly.

The main advantage of this approach is that it requires only the current load information for computing the FCMI and corresponding GENCO ranking without having to run the full OPF for every load variation. The FCMI will be used to analyze the GENCOS behavior in power market for any particular trading interval for any given loading conditions.

The membership values of loads to linguistic classes of low, medium, high, etc. constitute the input vector while the output vector presents the operator with the probability of a GENCO belonging to different market power class. Therefore, the proposed method can accept and analyze data in linguistic as well as in quantitative form. The fuzzy load modeling enables the handling of the uncertainty associated with power system loads and a whole set of scenarios is analyzed at one time.

This chapter is organized as follows. Study on market power is presented in section 3. FNN approach for open electricity market is produced in section 4. Power market assessment based on hybrid FNN is done in section 5. Training and testing detail of proposed FNN used for ranking of GENCOS for power market assessment is described in section 6. Section 7 is the conclusion.

3 Market Power

There are two main reasons why the potential of market power is brought to the electricity market. First there is market dominance and then there are transmission constraints [9]. Market power due to market dominance is a scenario that applies for every imperfect market and not only for the electricity market. On the electricity market, a supplier that is large enough to affect price can exploit market power by either economical withholding or physical withholding. When dealing with economical withholding a seller keeps bidding above the marginal cost of production and thereby driving up the price. Physical withholding simply means that a seller withholds some of its available capacity.

Market power due to transmission constraints makes it necessary to get a full understanding of the topology of the transmission system before starting any plan of detecting the potential for market power [10].

If a supplier is placed within a so called load pocket, this participant will have a local market power. A supplier in this case can find himself in a position of monopoly by intentionally create congestion and limit access of competitors. This means that by getting dispatched at strategic points in the network, a supplier in a load pocket can gain profit even by increasing its generation rather than by withholding its generationcapacity [11]. Conclusively, transmission constraints in the electricity market make it possible even for a small supplier to exploit market power.

In a network loads cannot be accurately forecasted and energy cannot be stored economically. Demand and supply must be balance all the time in order to maintain the system frequency, voltage, stabilization standards; Kirchhoff's laws and impedance of the whole network which determine the power flows in the system [12]. In the congested area generation capacity will be relative scarcity, so congestion results in locational market power and causes invalidation of the optimization of generating resources in the whole network.

Zonal market power has been recognized and analyzed in [17]. The Must-run ratio has been proposed to consider the transmission constraints. The MRR for Group A in a transmission zone is defined [18] as follows:

$$MRR = (Pd - Pl(\sum_{k=1}^{Ng} P_{gk,\max} - \sum_{k=1}^{NgA} P_{gk,\max})) / \sum_{k=1}^{NgA} P_{gk,\max} \tag{1}$$

Where Pl is the import limit of the zone, $P_{gk,\max}$ is the output limit of Generator k in the zone, N_g is the number of Generators in the zone, and N_{gA} is the number of Generators owned by Group A in the zone and P_d is the total load of the zone.

The MRR represents the capacity that must be provided by a generation company (GENCO) to supply a given load in a congestion zone as the percentage of the maximum available capacity of the GENCO. Theoretically, if the MRR of a seller is large than zero the seller is said to have market power. The MRR can provide useful market power signals in a congestion zone, which refers to a simple configuration in which one transmission line (or a set of lines in a "corridor") can be filled to its limit by exporting generation from a low-cost region to a high-cost region. However, the MRR does not clearly indicate the controllability of a GENCO over market price which usually depends on the market share owned by a GENCO to supply a given load in a congestion zone. This can be explained using a specific GENCO with 300 MW installed capacity in the following two different congestion zones. *Congestion zone 1*: the total load is 3000MW, the maximum import from other part of the system is 1000MW, the available generation capacity from other GENCOs in the zone is 1700 MW, and the capacity must be supplied by the specific GENCO is 300 MW.

In this case, the GENCO holds 300/3000=10% Generator market share (GMS) and the MRR=300/300=100 %. *Congestion zone 2*: the total load is 1000MW, the maximum import from other part of the system is 200 MW, the available generation capacity from other GENCOs in the zone is 500MW, and the capacity must be supplied by the specific GENCO is 300 MW. In this case, the GENCO holds 300/1000=30 % GMS and the MRR=300/300=100 %. Obviously, the specific GENCO in both cases has the same MRR but different market power due to different market shares.

Market participants may exercise their market power under certain system operating conditions through financial withholding and quantity withholding. Exercising market power by a supplier can expose customers to the risk of paying high price. Market power may appear in a deregulated power system under contingency states caused by random failures. For example, a random failure in a transmission line may results in network congestion and a generating unit failure may cause inadequate system generation capacity. Network congestion and generation inadequacy may result in local and system market powers. Although the probability of a contingency state is small and the state duration is short (usually from a few minutes to a few hours), the market power possessed by suppliers due to random failure may be quite larger than that in the normal state. If market participants exercise their market power under contingency conditions, the price can be extremely high (price spike). Customers usually use the hedging tools such as long term bilateral contracts, futures and options as risk management instruments to reduce the risk of their paying high prices. A customer has to know the possible risk of paying high price before making the decision to select a suitable hedging tool. It is therefore necessary to evaluate the risk of a customer being exposed to price spikes caused by exercising market power. Market powers caused by random failures and the associated probabilities are rarely considered currently in power market analysis.

4 FNN Approach for Open Electricity Market

A fuzzy neural network is employed for monitoring the power market and ranking the GENCOS. Load uncertainty is dealt with by representing loads as fuzzy variables in different linguistic categories. A fuzzy composite market index is proposed to screen market power and rank the GENCOS on line. This index is fuzzified in different severity classes to get a more informative ranking compared to conventional crisp approaches. The excellent non linear mapping characteristics of an efficient high performance neural network are utilized to map inputs with the expected outputs. Fuzziness incorporated at the input as well as at the output level provides flexibility and insight into the ranking process and a whole set of load scenarios are analyzed at one time. The application of an efficient neural network as a fuzzy inference engine eliminates the complicated process of fuzzy if then rule extraction. Once the fuzzy neural network is properly trained, GENCOS are ranked on the basis of the class membership values of FCMI. It is assumed that the index belongs to the severity class having highest value of membership. Due to the fuzzy approach, its probability of belonging to other classes is also available in the form of membership to other classes.

a. Fuzzy composite market index (FCMI)

A new fuzzy composite index is proposed by [31] for contingency ranking. In this paper same approach is used for screening of market power used by GENCOS in power system. The index is based on combining (i) Lerner Index, (ii) Relative market power (iii) Nodal cost. By including the effect of all three indicators it is ensured that the screening achieved will be more realistic and accurate.

(i) Lerner index

The Lerner index is used to measure the proportional deviation of price at the firm's profit-maximizing output from the firms marginal cost at that output. It is defined as the following:

$$LI_i = \frac{P_i - mc_i}{P_i} = \frac{1}{\varepsilon_i^d} \quad (2)$$

Where LI_i is the Lerner index for firm i, p_i and mc_i are price and marginal cost at the firms profit-maximizing output, respectively, and ε_i^d is the elasticity of demand seen by the firm. The Lerner index takes into the consideration of the effect of demand elasticity on market power. The Lerner index includes the effect of other fringe firm's elasticity of supply in the form of the market clearing price. Theoretically, if the LI of a company in a power system is large than zero it possesses the market power.

(ii) Relative Market Power

In general, one would expect the degree of substitutability between two Generators to be inversely related to how far apart they are on the network. Some Generators have market power in the Actual Experiments. These are the cases that an ISO would observe. Hence, the next question is whether or not Generators are using their market power effectively to raise prices. Seeing prices for Generators substantially higher than the prices paid to other Generators may raise suspicions, but, this situation is neither sufficient nor necessary for exploiting market power. Combining the results for the observed OPF with the high offer and the low offer by the Generators, respectively, it is possible to calculate the following measure of relative market power (RMP):

RMP=100[Competitive price-Low offer price/High offer price-Low offer price] (3)

High values of RMP close to the maximum of 100 indicate that market power has been exploited successfully. Although the RMP works quite well for our examples, it is still not an ideal measure. Developing better measures of the exploitation of market power is one of the ongoing objectives of our current research. It should be noted, however, that the main limitation of the RMP is the inability to discover the true costs. This is a deficiency on the supply side. From the perspective of customers, the prices paid are more important than measuring profits. Hence, the RMP, or, as an alternative, the ratio [Competitive price/Low offer price], provides a reasonably good measure of how well the power system is working for customers in a load pocket.

(iii) Nodal cost

Nodal prices are the price of power, which the supplier are paid and the price which consumers are billed. In order to compute these prices the Pool Operator receives bid plots from market participants for both supply and demand. Fig.1 shows bid plots for both demand and supply. G_m and $H_m$are market clearing price and market clearing volume of electric power respectively in \$/MWh and in MW. The prices shown on y-axis are in \$/MWh.The supply bid plot shows the minimum price at which a generator is willing to produce a certain amount of power, while demand bid plot shows the maximum price, which is accepted by customers to buy a certain amount of power. For, the sake of simplicity it is assumed here that supply and demand bid is a single price not complete plot.

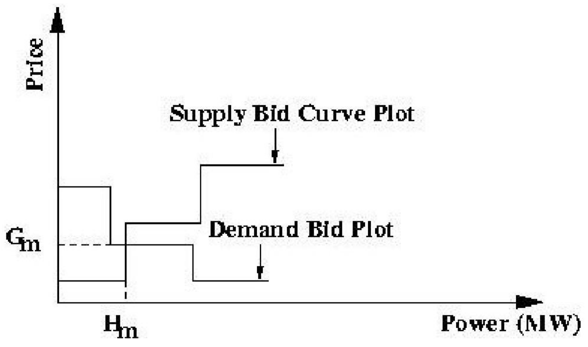


Fig. 1. Supply and Demand Bid Plot

In power market security pricing field, OPF-based approach is basically a non-linear constrained optimization problem. One crucial outcome of this optimization procedure can be nodal congestion prices. This outcome in pool-market operation is achieved through objective-function as Maximization of social welfare i.e. maximizing the generator's income for their power production and simultaneously ensuring that consumers pay cheapest price for their power purchase.

To combine the effect of all three, a composite index is proposed in (31) for contingency ranking the same approach is used here for GENCO assessment. The normalized values of LI, RMP and NC are fuzzified in different classes. Then the proposed index is computed as

$$FCMI = (\mu_{LI} \times W_{LI}) + (\mu_{LI}^{'} \times W_{LI}^{'}) + (\mu_{RMP} \times W_{RMP}) + (\mu_{RMP}^{'} \times W_{RMP}^{'}) + (\mu_{NC} \times W_{NC}) + (\mu_{NC}^{'} \times W_{NC}^{'}) \quad (4)$$

where μ_{LI} , μ_{RMP} and μ_{NC} are the memberships (highest value) of the class to which the market power of GENCOS belongs on the basis of FCMI value, i.e. on the basis of the combined effect of LI,RMP and NC. The memberships of the adjoining severity class (next highest value) are $\mu_{LI}^{'}$, $\mu_{RMP}^{'}$ and $\mu_{NC}^{'}$ and W_{LI} ,$W_{LI}^{'}$,W_{RMP} ,$W_{RMP}^{'}$,W_{NC} and $W_{NC}^{'}$ are the weighing factors.

b. Fuzzy modeling of power system loads

Load uncertainty is modeled by representing it as a fuzzy variable in the range (0–1) with memberships in different linguistic categories, such as, very small (VS), small (S), medium (M), large (L) and very large (VL). The membership value of ith linguistic category (μ_i) is calculated as [19]:

$$\mu_i = \frac{1}{1 + \left[\frac{x - a_i}{b_i}\right]^4} \tag{5}$$

where μ_i is the membership value in ith linguistic category, X is the crisp value to be fuzified, a_i and b_i are parameters corresponding to linguistic category i such as a_i determines the center value of the corresponding category, where the membership value is equal to 1.0 and b_i controls the width of the corresponding category. These parameters can be determined by carrying out simulations off-line under various operating conditions covering the possible range of variation. Past experience or operator judgment can also prove effective in setting these values. Non-linear membership functions are found to be most suitable to fuzzify power system variables (loads and FCMI) as they represent a more practical transition of loads from one category to the other compared to the common triangular or trapezoidal functions [19].

For each input variable x_i, the m data points in the x_i-y space are available. For every point in the x_i-y space, a fuzzy membership function ϕ_{ik} can be found, defined by [20]

$$\phi_{ik}(x_i) = \exp\left(-\left(x_{ik} - x_i / b\right)^2\right), \text{(k=1, 2, m)} \tag{6}$$

c. Data normalization

During training of a neural network, the higher valued input variables may tend to suppress the influence of smaller ones. Also the network does not produce outputs close to 1 or 0, as the neural network output governed by the activation or threshold function practically never realizes these values. To overcome this problem the input/output variables (x) are scaled in the range of 0.1–0.9. The normalized value x_n presented to the neural network as the input or target output is calculated using the equation:

$$x_n = \frac{(x - x_{\min})}{(x_{\max} - x_{\min})} 0.8 + 0.1 \tag{7}$$

Where x , $x_{\max}$ and $x_{\min}$ are the actual, maximum and minimum values of the variable which is to be normalized.

5 Hybrid Fuzzy Neural Network Based Power Market Assessment

The steps followed for power market assessment are:

(i) A large number of load patterns are generated randomly by perturbing the real loads at all the buses to cover the complete operating range of the power system under study.

(ii) For each pattern the values of LI, RMP and NC are calculated for each GENCO for each trading period using OPF solution and the offer prices of each GENCO.

(iii) The obtained indices are normalized between 0.1 and 0.9 for each load pattern using expression (7) and then fuzzified for computing the fuzzy composite market index (FCMI) using eq. (4).

(iv) The normalized loads at all buses are fuzzified into different linguistic categories and along with line codes (bi-polar digits used to represent the trading period) are fed to the fuzzy-neural network as training inputs. The first trading period is represented as (0 0 01) and so on.

(v) Computed FCMI in step three is then normalized and fuzzified into different linguistic categories. The membership values of FCMI of each GENCO form the desired output vector.

(vi) A one hidden layer neural network is trained with Levenberg– Marquardt back-propagation algorithm for input–output mapping. Once the network is properly trained, it is subjected to unseen patterns, for testing its performance.

(vii) During testing, a GENCO is assigned to the market power class for which it has highest value of membership.

a. Description of the test system

The hybrid fuzzy neural network was tested for measuring the existence of market power in the real-time operations of a power grid. An IEEE 14 bus system is used for testing and training the hybrid fuzzy neural network. The indices LI, RMP and NC are calculated from OPF solution obtained by MATPOWER [32] which simulate the full AC network, by using equ.(2) and (3). The weighing factors for computing FCMI in equ. (4) were taken equal to 1,2,3,4 and 5 for severity classes I, II, II, IV and V respectively for LI, RMP and NC. The weights were selected in this manner to assign highest weight to the most severe class (i.e. class V) and least weight to the least severe class (class I). Full AC, OPF solution were run for all load scenarios to obtain LI,RMP and NC for each trading period of an IEEE 14 bus system. The normalized LI, RMP and NC values were fuzzified using data given in Tables 1, 2 and 3 respectively. The graphical representation is given in figs 2, 3 and 4 respectively.

The value of FCMI of individual generator for each trading period was computed using membership values of the indices LI, RMP and NC. Table 4 presents the computation of FCMI of individual generator for 10 trading period each period has different load condition. The overall rank (last column) is found using fuzzy values of computed FCMI .Out of the 220 patterns generated 200 (20x10) were used for

Table 1. Fuzzy representation of LI

Linguistic Category for LI	Class V	Class IV	Class III	Class II	Class I
A	0.2	0.35	0.55	0.75	0.9
B	0.2	0.10	0.15	0.10	0.15

Table 2. Fuzzy representation of RMP

Linguistic Category for RMP	Class V	Class IV	Class III	Class II	Class I
A	0.2	0.45	0.65	0.8	0.9
B	0.3	0.15	0.2	0.10	0.2

Table 3. Fuzzy representation of NC

Linguistic Category for NC	Class V	Class IV	Class III	Class II	Class I
A	0.15	0.35	0.58	0.70	0.9
B	0.10	0.15	0.10	0.15	0.2

training the neural network while remaining 20 (2x10) unseen patterns were used to test its performance. Utility derived load compositions may also be employed to train the fuzzy-neural network instead of theoretically generated data. The obtained value of FCMI is normalized in the range of 0.1–0.9. Pattern wise normalization of FCMI ensures accurate ranking under peak as well as off-peak times of the day, because the generators are ranked for the current load based on their relative severity. Table 5 data was used to fuzzify normalized FCMI values into five fuzzy classes. The graphical representation is given in Fig. 5. The flexibility in ranking due to the fuzzy representation can be clearly seen. The operators and planners can set the different parameters to suit their system (as low, medium, high, etc. would have different numerical significance for different systems/variables) and thus flexibility is incorporated in the model.

b. Effectiveness of FCMI

The ranking of GENCOS on the basis of LI, RMP, NC and FCMI is compared in table 4 for 10 trading period. The significance of using FCMI for GENCOS classification becomes clear from table 4 which lists the values of the constituent

Table 4. FCMI and its Constituents LI, RMP, NC for IEEE 14 bus system

Trading period	GENCO	LI values		Rank	RMP Values		Rank	N.C. Values		Rank	FCMI (normalized)	Over all Rank
		Highest μ_{LI}	Next μ'_{LI}		Highest μ_{RMP}	Next μ'_{RMP}		Highest μ_{NC}	Next μ'_{NC}			
1	1	1.0 ,I	0.025,II	I	1.0 ,I	0.025,II	I	1.0 ,I	0.025,II	I	3.15 (0.1)	V
	2	0.93,III	0.30,II	III	0.85,III	0.47,II	III	0.91,III	0.35,II	III	10.31 (0.39)	IV
	3	0.83,IV	0.61,III	IV	0.73,III	0.68,IV	III	0.97,IV	0.79,V	IV	17.89 (0.71)	II
	4	1.0,V	0.16,IV	V	1,V	0.16,IV	V	1,V	0.16,IV	V	16.92 (0.67)	III
	5	0.97,V	0.62,IV	V	0.96,V	0.69,IV	V	0.99,V	0.67,IV	V	22.2 (0.89)	I
2	1	1.0 ,I	0.025,II	I	1.0 ,I	0.025,II	I	1.0 ,I	0.025,II	I	3.15 (0.1)	V
	2	0.96,III	0.24,II	III	0.9,III	0.37,II	III	0.92,III	0.33,II	III	10.22 (0.39)	IV
	3	0.99,IV	0.70,V	IV	0.99,IV	0.63,V	IV	0.99,IV	0.39,III	IV	19.7 (0.78)	II
	4	0.98,V	0.57,IV	V	0.97,V	0.63,IV	V	1,V	0.16,IV	V	20.19 (0.80)	II
	5	1,V	0.16,IV	V	1,V	0.16,IV	V	0.93,V	0.78,IV	V	19.05 (0.76)	II
3	1	1.0 ,I	0.025,II	I	1.0 ,I	0.025,II	I	1.0 ,I	0.025,II	I	3.15 (0.1)	V
	2	0.97,III	0.22,II	III	0.92,III	0.34,II	III	0.73,III	0.68,II	III	10.34 (0.39)	IV
	3	0.99,IV	0.61,V	IV	1,IV	0.54,V	IV	1,IV	0.45,V	IV	19.96 (0.79)	II
	4	1,V	0.16,IV	V	1,V	0.16,IV	V	1,V	0.16,IV	V	16.92 (0.67)	III
	5	0.98,V	0.53,IV	V	0.98,V	0.57,IV	V	1,V	0.24,IV	V	20.16 (0.80)	II
4	1	1.0 ,I	0.025,II	I	1.0 ,I	0.025,II	I	1.0 ,I	0.025,II	I	3.15 (0.10)	V
	2	0.99,II	0.19,III	II	0.87,II	0.09,III	II	0.84,II	0.61,III	II	8.07 (0.30)	IV
	3	0.74,III	0.67,II	III	0.99,II	0.36,III	II	0.99,III	0.38,II	II	11.72 (0.45)	III
	4	1,V	0.16,IV	V	1,V	0.16,IV	V	1,V	0.16,IV	V	16.92(0.67)	III
	5	0.9,III	0.38,II	III	0.91,II	0.54,III	II	0.89,II	0.88,III	II	14.87 (0.58)	III
5	1	1.0 ,I	0.025,II	I	1.0 ,I	0.025,II	I	1.0 ,I	0.025,II	I	3.15 (0.1)	V
	2	0.84,II	0.61,III	II	0.95,II	0.48,III	II	0.9,II	0.37,II	II	10.29 (0.39)	IV
	3	0.87,IV	0.58,III	IV	0.73,IV	0.69,III	IV	1,IV	0.49,V	IV	17.66 (0.70)	II
	4	1,V	0.16,IV	V	1,V	0.16,IV	V	0.57,IV	0.98,V	V	18.46 (0.73)	II
	5	0.91,V	0.85,IV	V	0.9,IV	0.87,V	IV	1,V	0.16,IV	V	21.54 (0.86)	I
6	1	1.0 ,I	0.025,II	I	1.0 ,I	0.025,II	I	1.0 ,I	0.025,II	I	3.15 (0.1)	V
	2	0.97,III	0.23,II	III	0.91,III	0.35,II	III	0.85,III	0.47,II	III	10.29 (0.39)	IV
	3	0.94,IV	0.83,V	IV	0.97,IV	0.79,V	IV	0.73,IV	0.68,V	IV	20.65 (0.82)	I
	4	1,V	0.16,IV	V	1,V	0.16,IV	V	1,V	0.16,IV	V	16.92 (0.67)	III
	5	0.99,V	0.73,V	IV	0.99,IV	0.67,V	IV	0.96,IV	0.69,V	IV	22.48(0.9)	I
7	1	1.0 ,I	0.025,II	I	1.0 ,I	0.025,II	I	1.0 ,I	0.025,II	I	3.15 (0.10)	V
	2	0.97,III	0.21,II	III	0.92,III	0.33,II	III	0.90,III	0.37,II	III	10.19 (0.39)	IV
	3	0.99,IV	0.38,V	IV	0.99,IV	0.39,III	IV	0.99,IV	0.63,V	IV	18.1 (0.72)	II
	4	1,V	0.16,IV	V	1,V	0.16,IV	V	0.97,V	0.63,IV	V	18.65 (0.74)	II
	5	0.95,V	0.72,IV	V	0.93,V	0.78,IV	V	1,V	0.16,IV	V	21.04 (0.84)	I
8	1	1.0 ,I	0.025,II	I	1.0 ,I	0.025,II	I	1.0 ,I	0.025,II	I	3.15 (0.1)	V
	2	0.85,III	0.47,II	III	0.73,III	0.68,II	III	0.92,III	0.34,II	III	10.48 (0.40)	IV
	3	1,IV	0.53,V	IV	1,IV	0.45,V	IV	1,IV	0.54,V	IV	19.6 (0.78)	II
	4	1,V	0.16,IV	V	1,V	0.16,IV	V	1,V	0.16,IV	V	16.92 (0.67)	III
	5	1,V	0.23,IV	V	1,V	0.24,IV	V	0.98,V	0.57,IV	V	19.06 (0.76)	II
9	1	1.0 ,I	0.025,II	I	1.0 ,I	0.025,II	I	1.0 ,I	0.025,II	I	3.15 (0.1)	V
	2	0.76,III	0.63,II	III	0.84,II	0.61,III	II	0.87,III	0.09,II	III	9.06 (0.34)	IV
	3	0.99,IV	0.40,V	IV	0.99,IV	0.38,III	IV	0.99,IV	0.36,III	IV	14.12 (0.55)	III
	4	1,V	0.16,IV	V	1,V	0.16,IV	V	1,V	0.16,IV	V	16.92 (0.67)	III
	5	0.92,V	0.81,IV	V	0.89,V	0.88,IV	V	0.91,V	0.54,III	V	19.25 (0.77)	II
10	1	1.0 ,I	0.025,II	I	1.0 ,I	0.025,II	I	1.0 ,I	0.025,II	I	3.15 (0.1)	V
	2	0.96,III	0.24,II	III	0.90,III	0.37,II	III	0.95,III	0.48,II	III	10.14 (0.39)	IV
	3	1,IV	0.57,V	IV	1,IV	0.50,V	IV	0.73,IV	0.69,III	IV	18.34 (0.73)	II
	4	0.98,V	0.52,IV	V	0.98,V	0.57,IV	V	1,V	0.16,IV	V	19.8 (0.79)	II
	5	1,V	0.16,IV	V	1,V	0.16,IV	V	0.9,IV	0.87,V	V	19.23 (0.76)	II

Table 5. Fuzzy representation of FCMI

Linguistic Category for FCMI	Class V	Class IV	Class III	Class II	Class I
A	0.2	0.35	0.45	0.6	0.9
B	0.3	0.15	0.2	0.15	0.2

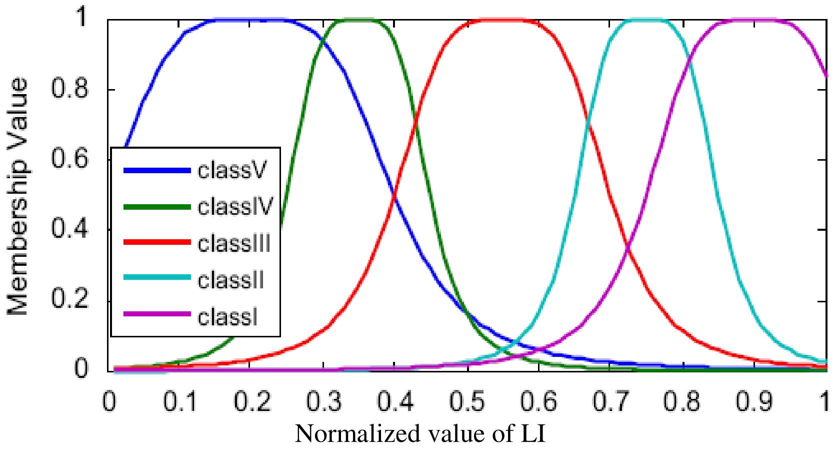


Fig. 2. Fuzzy representation of the LI index

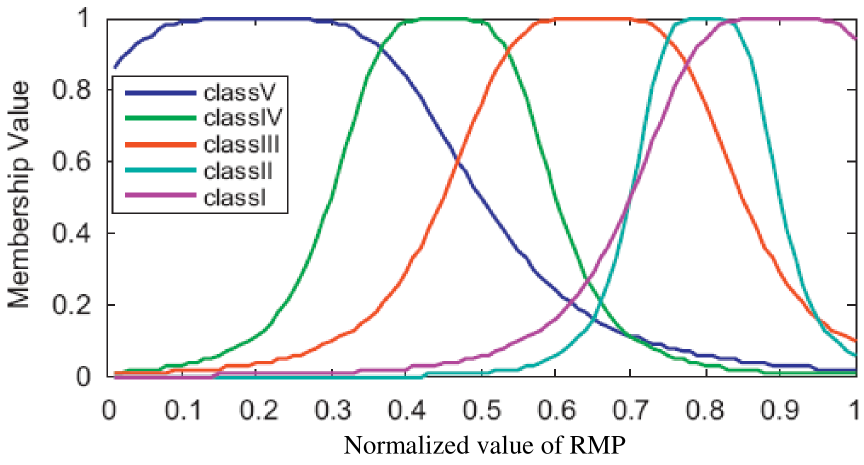


Fig. 3. Fuzzy representation of the RMP index

indices for 10 trading period. It can be observed that for a few trading period the classification remain same for all three indices but in other cases the ranking based on indices LI, RMP and NC is different. The composite index FCMI is hence very useful for ranking of GENCOS as it includes combined effect of all three individual indices. By using a composite index the effect of all three individual indices was effectively

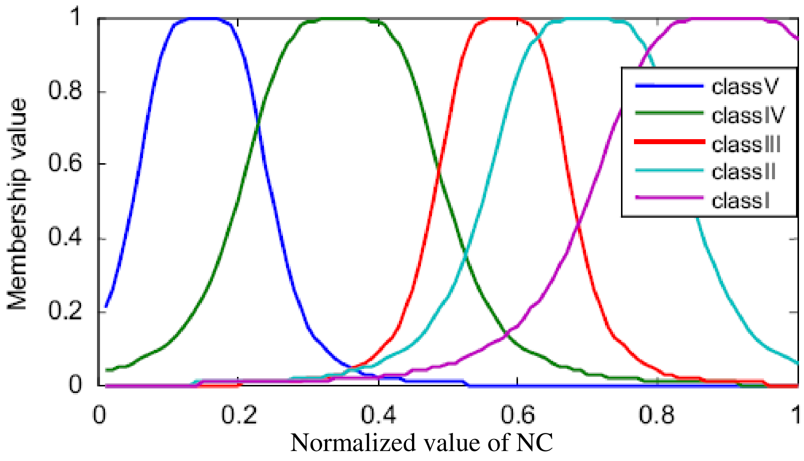


Fig. 4. Fuzzy representation of the NC index

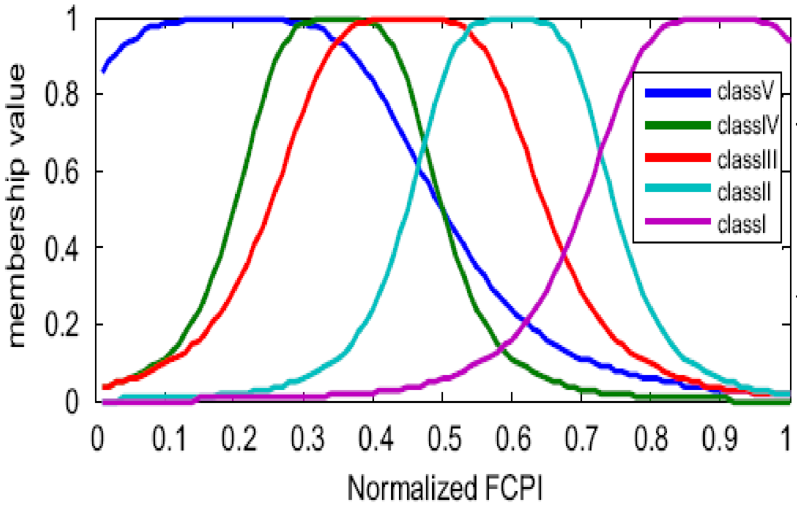


Fig. 5. Fuzzy representation of the FCMI index

included for market power assessment. The FCMI will be used to analyze the GENCOS behavior in power market for any particular trading interval for any given loading conditions.

6 Training and Testing Detail

The Levenberg– Marquardt algorithm [21, 22] was used for training the neural network. It is a variation of Newton's method. The conventional multi-layer perceptron (MLP) networks are usually trained using gradient descent based on back-propagation (BP) algorithm, which is too slow for practical problems. Recently, several high performance algorithms have been developed to train MLP models that converge 10 to 100 times faster than the BP algorithm. These algorithms are based on numerical optimization techniques like conjugate gradient, quasi-Newton and Levenberg–Marquardt algorithms. Out of these, Levenberg–Marquardt (LM) algorithm is found to be the fastest method for training moderate size feed forward neural networks [30]. It also has very efficient Matlab implementation. The proposed fuzzy-neural network is very advantageous as:

(i) No rule formation required and
(ii) Misranking is eliminated.

Hybridization of fuzzy logic with neural network has eliminated the need for deriving complex if-then rules by directly computing the membership values of composite performance index in all five severity classes. The proposed FNN-based method has an edge over conventional methods [23, 24, 25, 26, 27, and 28] that rank a pattern to a particular class based on its severity index because here any possibility of misranking is avoided by giving increased information in the form of membership values to neighboring classes. Simulations were carried out using MATLAB 7.0.1 on a Pentium IV processor, 2.8 GHz with 512GB RAM. The performance of the trained network was tested for 20 unseen trading periods. Market summary for 20 trading period are presented in Table 6.

The GMS represents the capacity that must be provided by a generation company to supply a given load in a congestion zone as the percentage of total load of the congested zone. The MRR represents the capacity that must be provided by a generation company (GENCO) to supply a given load in a congestion zone as the percentage of the maximum available capacity of the GENCO. Theoretically, if the MRR of a seller is large than zero the seller is said to have market power. Table 6 show the GMS and MRR for 20trading period respectively. For increased load demand, at trading period 4, MRR of GENCO 4 is 100% which indicate that GENCO4 used its maximum capacity. From table it can be easily concluded that as the demand is increased GENCO 4 and 5 offer high nodal costs with lowest MRR and GMS values try to exploit the market.

a. Architecture of the FNN for testing

The membership values of loads at all 14 buses along with a four digit topology number representing 20 trading period were used as inputs to the fuzzy-neural network making the number of neurons in the input layer equal to 59 (11X5+4). The first trading period is represented as (0001) and so on. There were five output layer neurons corresponding to the five membership classes of FCMI which reflect the market power assessment for each GENCO. Fuzzy rules for FCMI for five classes are given in table 7. GENCOS belonging to class 1 and 2 represent that they do not use market power. GENCOS belonging to class 3 represent that they partially utilize the market power to make their own profit, but do not exploit the market. GENCOS belonging to class 4 represent that GENCOS create a local market to maximize their profit, but still do not exploit the market. GENCOS belonging to class 5 represent that GENCOS fully utilize the market power to maximize their profit and fully exploit the market.

Table 6. Market summary for 20 testing pattern

Trading Period		GENCO1	GENCO2	GENCO3	GENCO4	GENCO5
1 (Pd=314.2)	MRR	60.34	27.04	00	51.81	35.91
	GMS	63.83	12.05	00	16.49	11.43
	Price ($/MWh)	37.26	38.93	39.94	41.04	40.72
2 (Pd=330.04)	MRR	60.41	27.11	18.96	35.14	49.30
	GMS	60.84	11.49	5.740	10.65	14.94
	Price ($/MWh)	37.28	38.97	40.38	40.70	40.99
3 (Pd=241.89)	MRR	58.81	26.33	00	15.78	3.34
	GMS	80.81	15.24	00	6.52	1.38
	Price ($/MWh)	36.82	38.43	39.69	40.32	40.07
4 (Pd=424.85)	MRR	57.62	27.81	44.62	100	66.74
	GMS	45.08	9.16	10.50	23.54	15.71
	Price ($/MWh)	36.48	39.47	40.89	49.19	41.33
5 (Pd=411.99)	MRR	60.66	27.12	16.19	71.65	48.81
	GMS	55.62	10.47	4.46	19.76	13.46
	Price ($/MWh)	37.53	38.98	40.32	41.43	40.98
6 (Pd=254.81)	MRR	59.18	26.56	2.94	27.57	00
	GMS	77.21	14.59	1.15	10.82	00
	Price ($/MWh)	36.93	38.59	40.06	40.55	39.98
7 (Pd=330.78)	MRR	60.7401	27.28	11.91	54.89	36.70
	GMS	61.04	11.55	3.60	16.59	11.09
	Price ($/MWh)	37.37	39.09	40.24	41.09	40.73
8 (Pd=367.25)	MRR	60.28	26.97	20.09	58.30	54.50
	GMS	55.99	10.55	5.61	16.29	15.23
	Price ($/MWh)	37.25	38.88	40.40	41.17	41.09
9 (Pd=359.69)	MRR	60.01	26.91	22.18	69.88	45.90
	GMS	55.46	10.47	6.17	19.43	12.76
	Price ($/MWh)	37.17	38.84	40.44	41.39	40.92
10 (Pd=331.87)	MRR	60.32	27.11	16.58	38.72	52.06
	GMS	60.42	11.43	4.99	11.67	15.69
	Price ($/MWh)	37.26	38.98	40.33	40.77	41.04
11 (Pd=386.23)	MRR	61.16	27.45	13.21	83.07	63.77
	GMS	52.64	9.95	3.42	21.51	16.51
	Price ($/MWh)	37.50	39.21	40.26	41.66	41.27
12 (Pd=278.48)	MRR	58.66	26.14	00	37.46	22.25
	GMS	70.02	13.14	00	13.45	7.99
	Price ($/MWh)	36.78	38.30	39.49	40.75	40.44
13 (Pd=250.18)	MRR	58.20	25.91	00	12.49	19.11
	GMS	77.3292	14.50	00	4.99	7.64
	Price ($/MWh)	36.65	38.14	39.45	40.25	40.38
14 (Pd=392.38)	MRR	60.50	27.17	29.93	85.02	55.30
	GMS	51.26	9.69	7.63	21.67	14.09
	Price ($/MWh)	37.31	39.02	40.60	41.70	41.10
15 (Pd=194.29)	MRR	51.83	23.00	00	00	00
	GMS	88.68	16.57	00	00	00
	Price ($/MWh)	34.83	36.09	37.42	37.31	38.49
16 (Pd=277.78)	MRR	58.59	26.06	00	37.16	22.15
	GMS	70.11	13.14	00	13.38	7.97
	Price ($/MWh)	36.76	38.25	39.65	40.74	40.44
17 (Pd=362.04)	MRR	60.87	27.40	23.93	50.56	62.16
	GMS	55.89	10.59	6.61	13.96	17.17
	Price ($/MWh)	37.41	39.18	40.48	41.01	41.24
18 (Pd=155.91)	MRR	40.73	17.96	00	00	00
	GMS	86.84	16.13	00	00	00
	Price ($/MWh)	31.65	32.58	33.28	33.58	33.65
19 (Pd=188.51)	MRR	49.70	22.03	00	00	00
	GMS	87.63	16.36	00	00	00
	Price ($/MWh)	34.22	35.42	36.16	37.18	36.81
20 (Pd=200.48)	MRR	52.95	23.58	00	00	00
	GMS	87.79	16.46	00	00	00
	Price ($/MWh)	35.15	36.50	37.15	38.03	38.12

Table 7. Fuzzy rules for FCMI

Class1	Class2	Class3	Class4	Class5
(NMP) GENCOS do not use market power	(NMP) GENCOS do not use market power	(PMP) GENCOS partially utilize the market power	(LMP) GENCOS create the local market	(FMP) GENCOS Fully utilize the market power

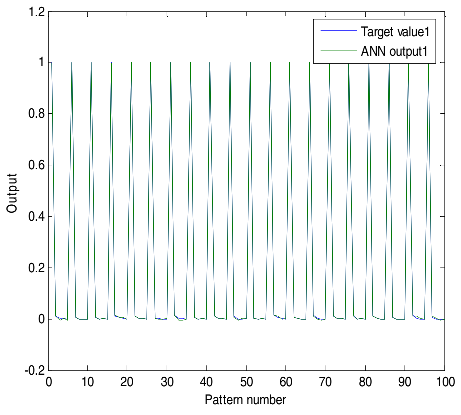


Fig. 6. Testing performance of FNN for Membership value of class 1

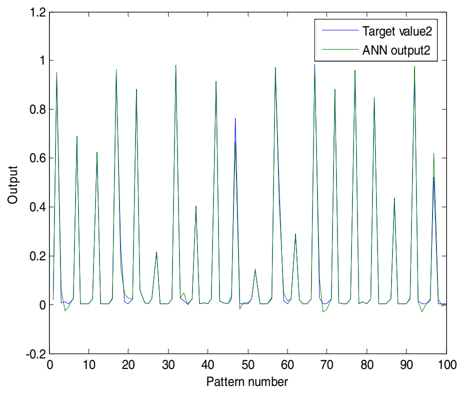


Fig. 7. Testing performance of FNN for Membership value of class2

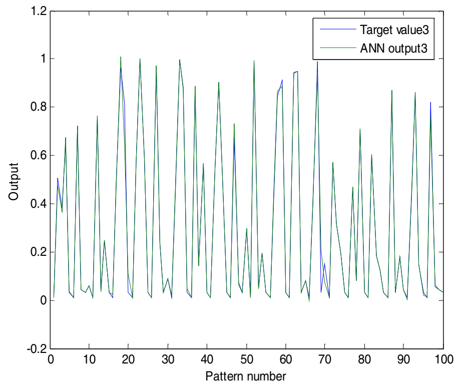


Fig. 8. Testing performance of FNN for Membership value of class 3

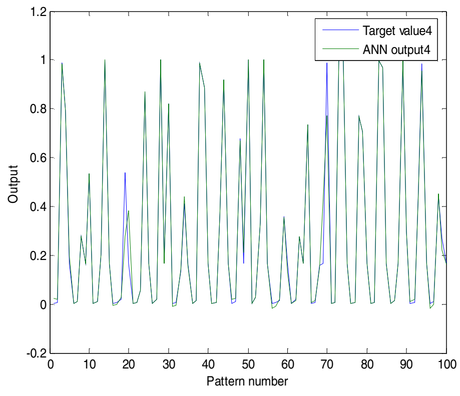


Fig. 9. Testing performance of FNN for Membership value of class4

Test results for all the 20 trading period are presented in Figs. 6–10 and it can be seen that fuzzy-neural network is capable of producing membership values of all the classes quite accurately. A GENCO belongs to the class for which it has highest membership.

Ranking of GENCOS for 20 testing trading period is shown in table 5. For each trading period GENCOS are ranked according to their FCMI values. For trading period 1 load is 314.2MW, the total load is shared by each GENCOS expect

GENCO3, which is clearly indicated by ranking as it belong to class1. However GENCO5 share the least load but exploit the market by offering high nodal cost and maximize the profit which is clearly indicated by ranking and from market summary table. For trading period 15,18,19 and 20 any GENCOS do not use market power as the load is low no congestion will occur and total load is shared by GENCO1 and 2 which is clearly indicated by ranking of GENCOS and also the ranking can be checked from market summary table. For trading period 2, 4, 5, 7, 8, 9, 10, 11, 14 and 17 demand is high congestion will occur, all GENCOS share the load, by offering high cost maximize their profit and exploit the market. It can be easily concluded that as the demand is increased GENCOS MRR will increase and they offer high cost to see the demand and exploit the market. The ranking of GENCO can be easily checked by market summary table which clearly indicate marginal cost, MRR and GMS values for each GENCO for each trading period.

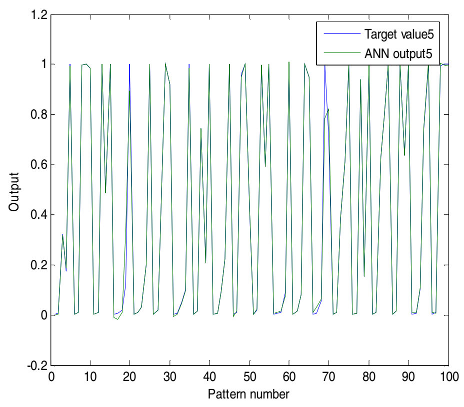


Fig. 10. Testing performance of FNN for Membership value of class 5

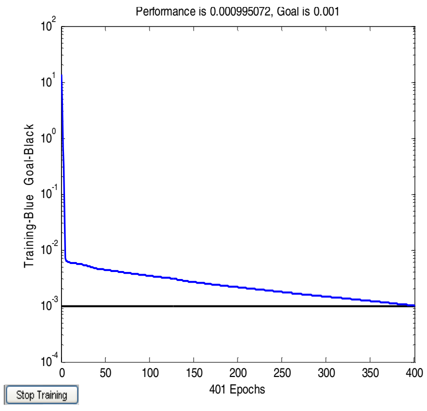


Fig. 11. Convergence characteristic for FNN

b. Market power assessment for each GENCO

The market power assessment of each GENCO for the proposed trading market is shown in figs 12-16. Out of the five GENCOS, GENCO1 always belong to class1 show that GENCO1 never use market power. GENCO2 belong to class2 and 3 show that GENCO2 partially try to utilize the market power but does not exploit the market. GENCO3, 4 and 5 belong to all classes clearly indicate that they use the market power to maximize their profit as per demand. GENCO 4 and 5 always try to use their market power as the demand is increased just because of their location in the system and also without their contribution demand cannot be fulfilled. Hence they raise their prices above the marginal cost. The proposed hybrid Fuzzy-neural network trained with LM algorithm is a model free estimator and its mapping accuracy is dependent on how closely the training patterns resemble the actual operating conditions. Being an intelligent system, it however has considerable fault tolerance and therefore can produce accurate results even for previously unseen operating conditions as long as they are within the same range. The computational time and complexity of conventional

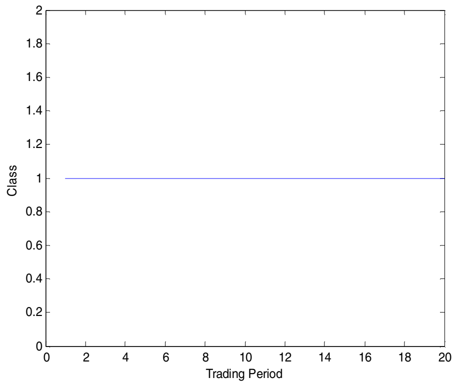


Fig. 12. Classes belonging to Genco1

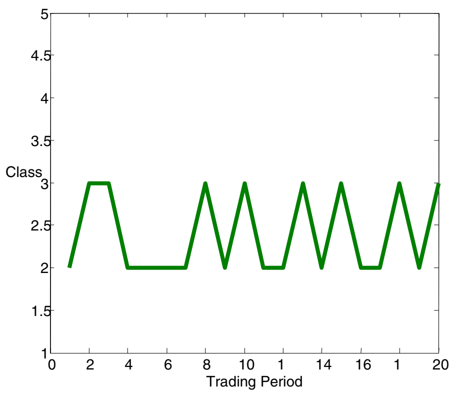


Fig. 13. Classes belonging to Genco2

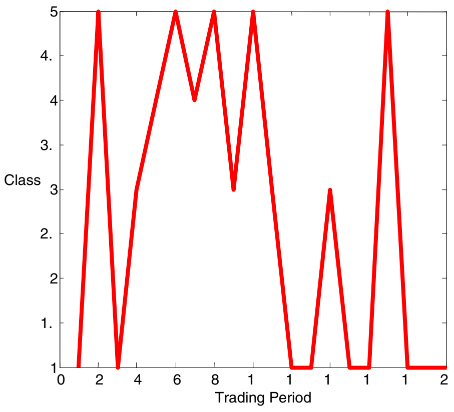


Fig. 14. Classes belonging to Genco3

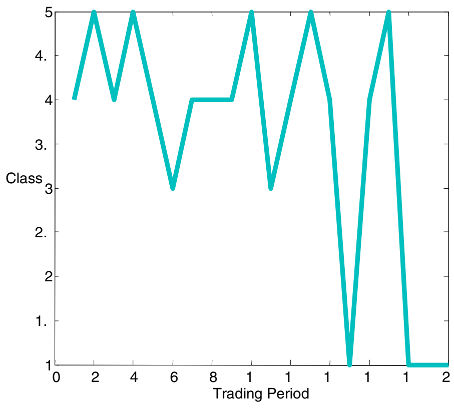


Fig. 15. Classes belonging to Genco4

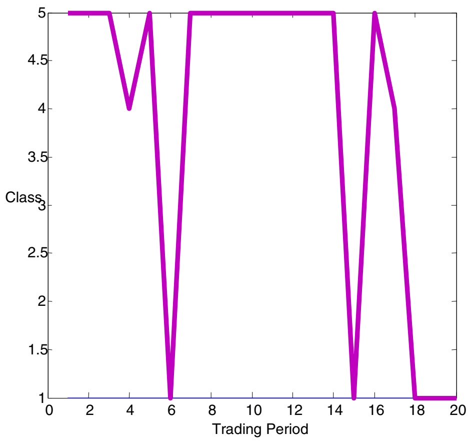


Fig. 16. Classes belonging to Genco5

approaches increase when all AC limits and load compositions are incorporated in the model, but in case of the proposed approach there will be no such effect as once it is trained off-line using data obtained from conventional methods, the results will be produced instantaneously, during the on-line application.

Discussion:

In this paper a multi output fuzzy neural network (FNN) is trained for market power assessment and for finding the on line market power ranking status of GENCOS in a competitive power system using a fuzzy composite market index (FCMI). This index is formulated by combining (i) Lerner Index, (ii) Relative market power and (iii) Nodal Cost. In the proposed FNN a trained multi-output neural network is being used as a fuzzy inference engine. The input of FNN consists of real loads and a bipolar code to represent a trading interval while the output consists of the fuzzy values of FCMI. To train the FNN a number of training patterns, covering the full operating range of the power system, are generated using the system data such as offer prices and operating constraints. OPF results are used to compute the above three market power indices and the corresponding FCMI. Once the network is trained it is capable

Table 8. Ranking of GENCOS for each Trading Period based on FNN

Trading Period	FCMI (Normalized, Rank)				
	GENCO1	GENCO2	GENCO3	GENCO4	GENCO5
1	0.1,I	0.4,II	0.1,I	0.68,IV	0.9,V
2	0.1,I	0.43,III	0.88,V	0.9,V	0.85,V
3	0.1,I	0.44,III	0.1,I	0.75,IV	0.9,V
4	0.1,I	0.38,II	0.59,III	0.9,V	0.78,IV
5	0.1,I	0.41,II	0.73,IV	0.77,IV	0.9,V
6	0.1,I	0.39,II	0.82,V	0.67,III	0.1,I
7	0.1,I	0.40,II	0.77,IV	0.79,IV	0.9,V
8	0.1,I	0.46,III	0.9,V	0.77,IV	0.87,V
9	0.1,I	0.39,II	0.64,III	0.78,IV	0.9,V
10	0.1,I	0.45,III	0.85,V	0.88,V	0.9,V
11	0.1,I	0.4,II	0.49,III	0.65,III	0.9,V
12	0.1,I	0.41,II	0.1,I	0.69,IV	0.9,V
13	0.1,I	0.49,III	0.1,I	0.9,V	0.82,V
14	0.1,I	0.39,II	0.59,III	0.64,IV	0.9,V
15	0.1,I	0.46,III	0.1,I	0.1,I	0.1,I
16	0.1,I	0.4,II	0.1,I	0.69,IV	0.9,V
17	0.1,I	0.42,II	0.83,V	0.9,V	0.74,IV
18	0.1,I	0.50,III	0.1,I	0.1,I	0.1,I
19	0.1,I	0.39,II	0.1,I	0.1,I	0.1,I
20	0.1,I	0.47,III	0.1,I	0.1,I	0.1,I

of predicting the FCMI values in five fuzzy classes (GENCO ranking) for any given operating scenario, on line, instantaneously, without bothering about the computational burden of OPF. The computational effort is required only for training the network which is an off line process.

This proposed technique is applied on an IEEE 14 bus system. All load buses with their demand is taken as input to FNN and FCMI computed after running OPF program is taken as output to FNN. Firstly we train the FNN and then testing is done with unseen load patterns. After testing and training it can be easily analyzed that at low demand GENCOS do not use market power means they do not raise their prices above the marginal price. As the demand is increased due to some reasons some GENCOSin this case GENCO 4 and 5 raise their prices above the marginal price just because of their location in the system. Without their contribution demand cannot be supplied hence they use their market power. By calculating FCMI we can easily identify which GENCO uses their market power and after identification they can be ranked.

Once we train FNN it requires only the current load information for computing the FCMI and provide GENCO ranking without having to run the full OPF for every load variation. The FCMI will be used to analyze the GENCOS behavior in power market for any particular trading interval for any given loading conditions**.** Since the training of ANN is extremely fast and test results are accurate, in future they can be directly floated to OASIS (open access same time information system) and any other web site. The ISO and customers can access this information instantly.

7 Conclusion

A comprehensive and dynamic market monitoring system has been proposed in this chapter to protect and improve the open electricity markets. Several important indices have been proposed as part of the market monitoring system, such as GMS, MRR, LI, RMP and NC all are combined together to give the single index FCMI. With this proposed market monitoring system handy, all participants will be able to have a better understanding of their markets and the policy makers will have a better gauge to measure the market behavior. It is also expected that this analysis will help make better market policies and find more incentives for everybody within the power system to improve the overall system operation and reliability as well as the market performance. It is strongly recommended that market participants and policy makers use this conundrums measurement and indices to locate and mitigate their market problems so that the power system is treated as a whole, not just as generation, transmission, load or other individual components.

In this paper a hybrid fuzzy neural network is developed for online ranking of GENCOS for each trading period. The proposed combined index developed using the effect of individual indexes is found to be very efficient for ranking GENCOS compare to methods which use only one index. Loads are modeled as fuzzy variables in contrast to the conventional deterministic approaches. The complicated task of fuzzy rule framing is not required here because a trained neural network serves as an inference engine. It has been demonstrated that the proposed method is particularly suitable for ranking of GENCOS lying on class boundaries because the fuzzy environment increases amount of information available and provides ranking within a severity class. The main advantage of this approach is that it requires only the current load information for computing the FCMI and corresponding GENCO ranking without having to run the full OPF for every load variation. The FCMI will be used to analyze the GENCOS behavior in power market for any particular trading interval for any given loading conditions.

References

[1] Nicolaisen, J., Petrov, V., Tesfatsion, L.: Market Power and Efficiency in A Computational Electricity Market with Discriminatory Double Auction Pricing. IEEE Transactions Evol. Compute. 5(5), 504–523 (2000)

[2] Bompard, E., Carpaneto, E., Chicco, G., Napoli, R.: Reactive Load Modeling Impacts on Nodal Prices in Pool Model Electricity Markets. In: IEEE Power Eng. Soc. Summer Meeting, WA, July 14-16, pp. 2150–2155 (2000)

[3] Bompard, E., Carpaneto, E., Chicco, G., Gross, G.: The Role of Load Demand Elasticity in Congestion Management and Pricing. In: IEEE Power Eng. Soc. Summer Meeting, WA, July 14-16, pp. 2229–2234 (2000)

[4] Marannino, P., Vailati, R., Zanellini, F., Bombard, E., Gross, G.: OPF Tools for Optimal Pricing and Congestion Managementin A Two Sided Auction Market Structure. In: IEEE Power Tech., Porto, Portugal, September 10-13, vol. 1 (2001)

[5] Hamoud, G., Bradley, I.: Assessment of Transmission Congestion Cost and Locational Marginal Pricing in A Competitive Electricity Market. IEEE Transactions on Power Systems 19(2), 769–775 (2004)

[6] Chen, H., Canizares, C.A.: Transaction Impact Analysis and its Application in Electricity Markets. In: Power Engineering Large Engineering Systems Conf., May 7-9, pp. 2–6 (2003)

[7] Anderson, E.J., Philpott, A.B.: Optimal Offer Construction in Electricity Market. Math. Oper. Res. 27, 82–100 (2002)

[8] Lamont, J.W., Rajan, S.: Strategic Bidding in An Energy Brokerage. IEEE Transactions on Power Systems 12(4), 1729–1733 (1997)

[9] Li, C.A., Svoboda, A.J., Guan, X., Singh, H.: Revenue Adequate Bidding Strategies in Competitive Electricity Markets. IEEE Transactions on Power Systems 14(2), 492–497 (1999)

[10] Zhang, D., Wang, Y., Luh, P.B.: Optimization Based Bidding Strategies in the Deregulated Market. In: IEEE Power Industry Computer Applications Conf., May 16-21, pp. 63–68 (1999)

[11] Visudhiphan, P., Ilic, M.D.: Dynamic Games Based Modeling of Electricity Market. In: IEEE Power Eng. Soc. Winter Meeting, New York, January 31-February 4, vol. 1, pp. 274–281 (1999)

[12] Wen, F.S., David, A.K.: Optimal Bidding Strategies and Modeling of Imperfect Information among Competitive Generators. IEEE Transactions on Power Systems 16(1), 15–21 (2001)

[13] Yang, L., Wen, F., Wu, F.F., Ni, Y., Qiu, J.: Development of Bidding Strategies in Electricity Markets Using Possibility Theory. In: Power System Tech. Conf., October 13-17, vol. 1, pp. 182–187 (2002)

[14] Kleindorfer, P.R., Wu, D.J., Fernando, C.S.: Strategic Gaming in Electric Power Markets. In: Proc. 33rd Annu. Hawaii Int. Conf. System Sciences, January 4-7, pp. 1345–1354 (2000)

[15] Bompard, E., Italino, F., Napoli, R., Ragazzi, E.: An IPV Auction Model for Strategic Bidding Analysis under Incomplete and Asymmetric Information. In: 13th Conf. Intelligent Systems Application Power Systems, August 31-September 3 (2003)

[16] Younes, Z., Ilic, M.: Generation Strategies for Gamming Transmission Constraints: Will The Deregulated Electric Power Market be An Oligopoly? In: IEEE Hawai Int. Conf. System Science, January 6-9, vol. 3, pp. 112–121 (1998)

[17] Gan, D., Bourcier, D.V.: Locational Market Power Screening and Congestion Management: Experience and suggestions. IEEE Transactions on Power Systems 17, 180–185 (2002)

[18] Peng, W., Yu, X., Yi, D.: Nodal Market Power Assessment in Electricity Markets. IEEE Transactions on Power Systems 19(3), 1373–1379 (2004)

[19] Abdul-Rahman, K.H., Shahidehpour, S.M., Daneshdoost, M.: AI Approach to Optimal VAR Control withFuzzy Reactive Loads. IEEE Transactions on Power Systems 10, 88–97 (1995)

[20] Lin, Y., George, A., Cunningham: A New Approach to Fuzzy Neural System Modeling. IEEE Transactions on Fuzzy System 3(2), 190–198 (1995)

[21] Hagan, M.T., Mehnaj, M.H.: Training Feed Forward Neural Networks with Marquardt Algorithm. IEEE Transactions on Neural Network 5(6), 989–993 (1994)

[22] Saini, L.M., Soni, M.K.: Artificial Neural Network based Peak Load Forecasting using Levenberg–Marquardt and Quasi-Newton Methods. IEEE Proceeding Generation, Transmission and Distribution 149(5), 578–584 (2002)

[23] Chauhan, S.: Fast Real Power Contingency Ranking Using Counter Propagation Network. Electric Power Systems Research, 343–352 (2005)

[24] Liu, C.C., Chang, C.S., Su, M.C.: Neuro-Fuzzy Networks for Voltage Security Monitoring Based on Synchronized Phasor Measurements. IEEE Transactions on Power Systems 13, 326–332 (1998)

[25] Greene, S., Dobson, I., Alvarado, F.L.: Contingency Ranking for Voltage Collapse via Sensitivities From A Single Nose Curve. IEEE Transactions on Power Systems 1, 232–240 (1999)

[26] Nahman, J., Okljev, I.: Fuzzy Logic and Probability Based Real-Time Contingency Ranking. Electrical Power & Energy Systems 22, 223–229 (2000)

[27] Pandit, M., Srivastava, L., Sharma, J.: Voltage Contingency Ranking using Fuzzified Multilayer Perceptron. Electric Power Systems Research 59(1), 65–73 (2001)

[28] Pandit, M., Srivastava, L., Sharma, J.: Fast Voltage Contingency Selection Using Fuzzy Parallel Self-organizing Hierarchical Neural Network. IEEE Transactions on Power System 18, 657–664 (2003)

[29] Pandit, M., Srivastava, L., Singh, V., Sharma, J.: Coherency Based Fast Voltage Contingency Ranking Employing Counter Propagation Neural Network. Engineering Applications of Artificial Intelligence 20(8), 1133–1143 (2007)

[30] Chaturvedi, K.T., Srivastava, L., Pandit, M.: Levenberg Merquardt Algorithm Based Optimal Load Dispatch. In: Presented in IEEE Power India Conference 2006, New Delhi, April 10-12, pp. 10–12 (2006); Proceedings on IEEEXPLORE site

[31] Chaturvedi, K.T., Pandit, M., Srivastava, L., Sharma, J., Bhatele, R.P.: Hybrid Fuzzy-Neural Network-Based Composite Contingency Ranking Employing Fuzzy Curves For Feature Selection. Neurocomputing 73, 506–516 (2009)

[32] `http://www.pserc.cornell.edu/matpower`

Chapter 3
Coaching Robots: Online Behavior Learning from Human Subjective Feedback

Masakazu Hirkoawa[1] and Kenji Suzuki[2]

[1] Dept. of Intelligent Interaction Technologies, University of Tsukuba, 1-1-1 Tennodai, Tsukuba, Ibaraki, 305-8573, Japan
hirokawa@ai.iit.tsukuba.ac.jp

[2] Faculty of Engineering, Information and Systems, University of Tsukuba, 1-1-1 Tennodai, Tsukuba, Ibaraki, 305-8573, Japan
kenji@ieee.org

Abstract. This chapter describes a novel methodology for behavior learning of an agent, called Coaching. The proposed method is an interactive and iterative learning method which allows a human trainer to give a subjective evaluation to the robotic agent in real time, and the agent can update the reward function dynamically based on this evaluation simultaneously. We demonstrated that the agent is capable of learning the desired behavior by receiving simple and subjective instructions such as positive and negative. The proposed approach is also effective when it is difficult to determine a suitable reward function for the learning situation in advance. We have conducted several experiments with a simulated and a real robot arm system, and the advantage of the proposed method is verified throughout those experiments.

1 Introduction

In general, most of the methodologies for behavior learning consist of the following steps: Performing an action, evaluating the result of the action and modifying the learning parameters accordingly. The evaluation step is important in order to modify each parameter. Although the best way to achieve learning is to design an evaluation function which immediately gives the correct evaluation for every action, it is often not practical to design an evaluation function in advance due to the necessity of covering the whole state space. Autonomous machine learning methods, for example reinforcement learning (RL), have attracted attention for many years [12]. In RL, the agent updates the expected reward, called the state-value, according to a simple evaluation function, called the reward function, and acquires the action rules to maximize the summation of rewards. Therefore, RL can be applied to an environment even with delayed rewards. For this reason, RL can help to acquire a desired behavior in a real environment. Most conventional methodologies based on RL are based on the assumption that the agent will be able to get its first reward in a reasonable time. At the beginning of learning, the agent explores the state space randomly until it discovers the first reward, and then it starts learning based on that reward. Therefore, if the possibility of discovering a reward within the explorable area with respect to the initial state is low, the agent will not learn for a considerable period of time. Moreover, this problem cannot be solved by learning methodologies that focus on improving the learning efficiency based on rewards. To avoid this problem, we should design the reward function carefully. Although the learning efficiency is strongly affected by the reward function, methodologies to design the reward function are still not available. Consequently, it is required to design the reward function depending on the task and the environment according to our own experience. As mentioned above, although there are many techniques available to improve the learning efficiency based on the reward, we still need human knowledge to design the reward function. Especially when the agent tries to learn a complicated task in a real environment, the suitable reward function is very difficult to be created in advance. In such cases, it is necessary to find the optimal reward function for the learning algorithm used through trial and error, and this is usually very hard.

Several works have been done in order to involve human trainers in the online behavior learning of an agent. Thomaz et.al have proposed a learning framework called Socially Guided Machine Learning [14][15]. In their work, the agent had a learning mechanism based on RL and also had a channel to interact with a human trainer while learning. Thus, they developed an interactive learning agent, and it was verified that the learning performance is improved by getting feedback from the human trainer. Several intuitive ways which reduce the load on humans' instruction and acquire complex and diverse behaviors through the interaction with the trainer, such as Imitation learning (IL)[11] and Programming by demonstration (PbD)[3], attracted considerable attention in the past few decades. Although several studies have shown that those techniques are effective for real environments, there are still some problems. For instance, Inamura et al [5] and Jakel et al [6] have proved that a humanoid or anthropomorphic robot could generate its behavior by imitation. Atkenson et al [1][2] have introduced a method for behavior learning from demonstration using RL, and verified the effectiveness of their proposed method through the pendulum swing up task. Also if we do not know the exact goal of the task until we observe the movement, we cannot define the evaluation function nor the reward function in advance. For example, if we consider intuitive instructions such as ``walk well" or ``dance beautifully," it is not easy to design a task and define the evaluation function. In these tasks, only human decision based on subjective view of human is a cue for the learning agent. Moreover, it is also necessary to prepare different interfaces between the trainer and the agent for demonstrations in different environments. These issues are further hampered by the necessity of detailed information, for example the trajectory of the joint movement from the trainer regarding the desired behavior, because IL and PbD focus on how to imitate the trainer's behavior correctly.

On the other hand, the requirement for RL to learn is the reward function which can be defined in every task as a specific function. We assume that RL is useful for behavior learning, but the question of designing the reward function remains unanswered. Learning through interaction with a human trainer can realize action acquisition without a reward function, but giving too much concrete information regarding each task and its environment decreases its general versatility. Thus, the teaching signals from the trainer to the agent should be as simple and intuitive as possible. We propose a new learning framework which allows human trainers to give subjective and simple feedback to the agent asynchronously and in real time, to lead the agent towards learning the desired behavior. Since this framework is inspired by human-to-human skill transfer process, it is intuitive for human trainers, and we define this framework as Coaching a robot, opposed to Teaching a robot. The biggest difference between Teaching and Coaching is: conventional Teaching methodologies require adequate evaluation function in advance, while Coaching, as indicated in Fig. 1, allows the agent to learn not only the desired behavior but also the adequate evaluation function as an internal model of human trainer, simultaneously. In this paper, we introduce how to achieve the implementation of the Coaching framework on a typical RL agent.

In the typical RL method, designing the reward function and running learning algorithms are separated processes, but the proposed method aims to achieve both processes in parallel. By using this method, the human trainer can support the agent's learning and reduce the load of designing a reward function. Moreover, we use abstract and primitive binary values as the evaluation quantity of Coaching, namely *positive* or *negative*. Thus, the purpose of this study is to realize interactive and intuitive behavior learning without any technical knowledge about machine learning or the use of special interfaces.

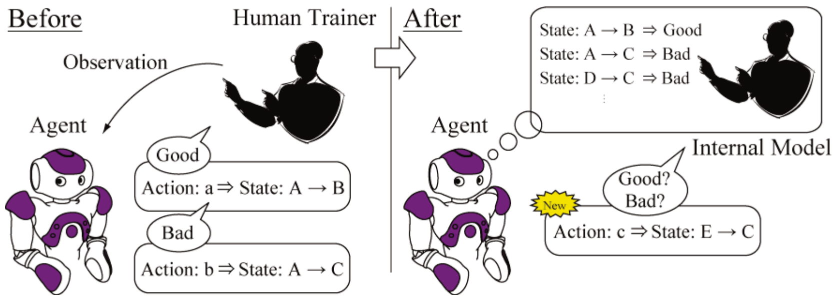


Fig. 1. Coaching is an interactive learning methodology in which a human trainer can assist the agent in an intuitive manner by giving subjective feedback in real time in addition to the learning mechanism implemented in the agent

2 Methodology

2.1 Coaching

In the conventional method for controlling a robotic system, it is essential to build a model of the plant that includes its dynamic specifications. The process is usually hard and time-consuming in a machine with a complicated structure. Machine learning methodologies have been proposed to reduce the load of such processes. However, as we described in the previous section, human effort is still needed to design the evaluation function.

On the other hand, Coaching aims to achieve behavior acquisition without a mathematical model or prior design of the evaluation function, which is achieved by implementing a mechanism to infer the subjective evaluation given by a trainer interactively and learning the behavior based on that evaluation. In addition, Coaching allows the emergence of different results of behavior learning. This means, even for the same task, there is a possibility of a different result emerging depending on the trainer. We have proposed the original idea of Coaching [10] and have demonstrated that a biped robot can adjust its own parameters for balancing and walking based on the human subjective evaluation. Riley et al. have developed a system which can refine the motion of a humanoid by subjective evaluations from a human [8]. In the above studies, the agent can modify its own parameters based on the evaluation given by the trainer, but the agent does not have the ability to learn automatically by the internal value like RL.

In contrast, the proposed method aims to implement Coaching using an agent that has a learning ability based on its own internal values. As shown in Fig. 2, the trainer observes the agent's behavior and gives feedbacks at random times, and the agent interprets it in order to modify its evaluation or reward function. In the meantime, the robot agent continues not only to estimate the trainer's internal evaluation function by considering the causality and consistency of given feedback but also to learn from interaction with the surrounding environment. Moreover, this method can be applied to other autonomous machine learning methods, for example RL, as an extrapolating algorithm.

To learn the reward function from such simple and abstract feedback, the agent needs to interpret the given feedback in order to modify the reward function as intended by the human trainer. Fig.3 shows the basic idea of interpretation method, consisting of two processes which are *Causality Detection* and *Error Detection* of the given feedback. *Causality Detection* is the process in which the agent determines specific states that motivated the trainer to give the feedback.

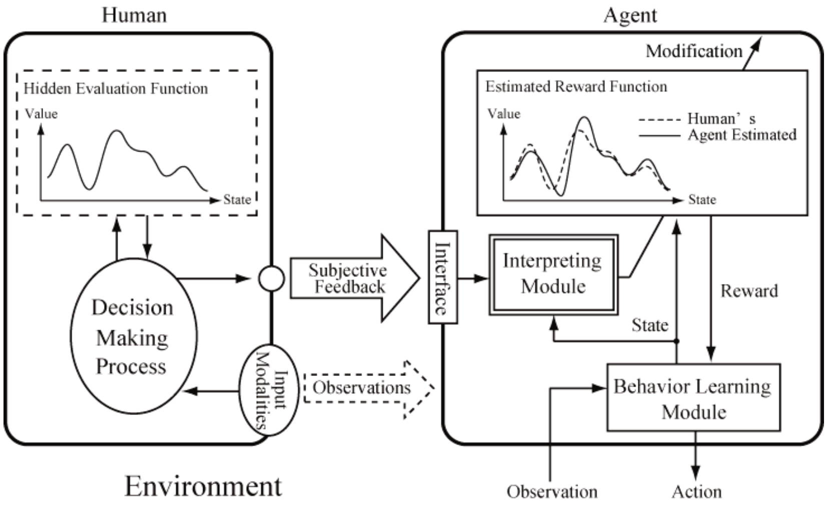


Fig. 2. Learning model of the proposed method. Human trainer can intervene the learning process of the agent by updating the reward function based on Coaching as a learning assistance.

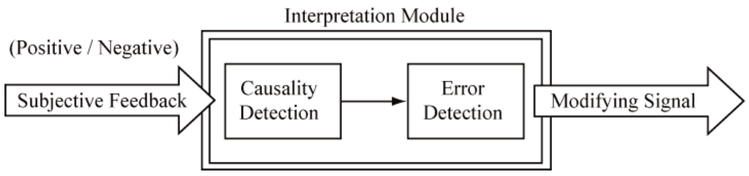


Fig. 3. "Causality Detection" means the determination of target behavior according to characteristics of human evaluation. "Error Detection" means the verification of feasibility of human feedback.

Fig. 4 illustrates an example of a time series of state variable x_i. As shown in this figure, when the trainer's feedback indicated by *Coaching* is given to the agent, the target behavior of that evaluation is considered in a time-range from the minimum delay T_1 to the maximum delay T_2. This is a characteristic of human trainer and subjective feedback. Since this time delay(T_1, T_2) depends on the task or the robot's hardware, we need to conduct a preliminary experiment, described in Sec 3.2, to investigate the time delay for a given task.

In addition, the consistency of human's feedback signal plays an important role. While Coaching, it can happen that the human trainer sometimes gives inconsistent feedbacks to the agent, in other words, different feedbacks to the same robot behavior is given occasionally. In this case, the agent carefully observes the consistency of the human evaluation, by comparing it to current and previous feedbacks to a similar behavior. If the feedback is inconsistent, it will not be accounted for, and the update of reward function does not occur. This mechanism is also included in the interpretation module as *Error Detection* (Fig.3).

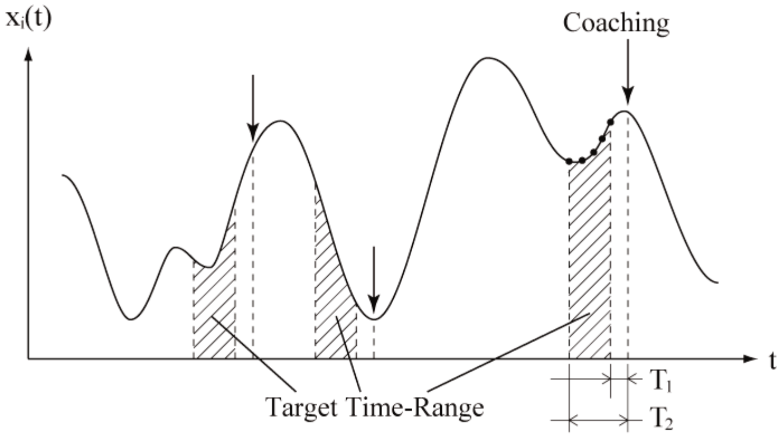


Fig. 4. Target Time-Range: Although the human trainer gives feedbacks to the agent soon after observing the agent's behavior, there is a certain time delay. The agent needs to interpret that the feedback could be given to a specific past series of actions.

2.2 Reinforcement Learning in Continuous State-Action Space

This study attempts to achieve behavior learning in a real environment. Thus, we must deal with an agent operating in a continuous state-action space. In this study, the approximated expression of continuous state-action space by using a Radial Basis Function (RBF) network, with reference to the following works [9][4][7][13], is used.

$$\mathrm{S}(\boldsymbol{x}_t) = \sum_k w_k b_k(\boldsymbol{x}_t) \tag{1}$$

$\boldsymbol{x}_t$ is the observed state variable at time t, $b_k(\boldsymbol{x})$ is the basis function unit number k, and w_k is a weight variable for k. The basis function is given by the Gaussian function,

$$b_k(\boldsymbol{x}) = exp\left(-\frac{\|\boldsymbol{x} - \mu_k\|}{\rho\sigma_k^2}\right) \tag{2}$$

where μ_k and σ_k correspond to the center and standard deviation of the Gaussian function. ρ denotes the stretch of each units, and it is fixed as a constant number in this paper. The state value $V(\boldsymbol{x}_t)$ and the action output $u(\boldsymbol{x}_t)$ can be expressed as:

$$V(\boldsymbol{x}_t) = \sum_k w_k b_k(\boldsymbol{x}_t) \tag{3}$$

$$u(\boldsymbol{x}_t) = \sum_k v_k b_k(\boldsymbol{x}_t) + n_t \tag{4}$$

w_k and v_k are weight variables, and n_t is a random number aiming at exploring the state space. The agent updates its own state value and action output by repeating the following steps.

$$\delta = r(\boldsymbol{x}_t) + \gamma V(\boldsymbol{x}_t) - V(\boldsymbol{x}_{t-1}) \tag{5}$$

$$w_k \leftarrow w_k + \alpha\delta b_k(\boldsymbol{x}) \tag{6}$$

$$v_k \leftarrow v_k + \beta\delta u(\boldsymbol{x}) b_k(\boldsymbol{x}) \tag{7}$$

δ is a TD error, $r(\boldsymbol{x}_t)$ is a reward function, γ is the discount rate and α and β are learning coefficients.

2.3 Implementation of Coaching

To implement Coaching on the above mentioned RL agent, the agent must interpret the human subjective feedback for updating the reward function, and this is done in account of a target behavior which is defined as a series of states and actions $u(\boldsymbol{x}_t)$ in the past between $t - T_1$ and $t - T_2$. The reward function should also be expressed as a continuous formula. In the same way, using an RBF network, we defined the reward function as:

$$r(\boldsymbol{x}_t) = r_{init} + \sum_k w_k b_k(\boldsymbol{x}_t) \tag{8}$$

The first term, r_{init}, refers to the initial reward function, which defines the goal state of the task given at the beginning of the learning algorithm, and the second term means the dynamic reward which is updated by Coaching during learning.

When the human trainer noticed the target behavior, and gave the evaluation at time t. Using T_1 andT_2, the state variable X is defined as:

$$X = \{\boldsymbol{x}_{t-T_2}, \boldsymbol{x}_{t-T_2+1}, \cdots, \boldsymbol{x}_{t-T_1}\} \tag{9}$$

$$\equiv \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots . \boldsymbol{x}_n\} \tag{10}$$

It is assumed that the target behavior happens during X. If the evaluation is "positive", it is appropriate to increase the reward around the state set X. Thus, the weight parameters of the RBF network are updated in a similar way to RL:

$$\delta_i = \lambda^{n-i}R + \gamma r(\boldsymbol{x}_i) - r(\boldsymbol{x}_{i-1}) \tag{11}$$

$$w_k \leftarrow w_k + \alpha\delta_i b_k(\boldsymbol{x}_i) \tag{12}$$

Where R is the immediate reward of Coaching and λ $(0 < \lambda \leq 1)$ is a discount rate. By iterating i from n to 1, a gradually decreasing gradient from the state $\boldsymbol{x}(n)$ to $\boldsymbol{x}(1)$ is generated on the reward function with consideration for the consistency of the history of human feedback. For example, if $r(\boldsymbol{x}_i)$ is bigger than $r(\boldsymbol{x}_{i-1})$ before updating, the agent had already learned the transition of state form $\boldsymbol{x}_{i-1}$ to $\boldsymbol{x}_i$ is helpful to get more reward. In addition to the use of the discounted reward, $\lambda^{n-i}R$ the modifying signal δ_i is enhanced. In contrast, if the subtraction derives negative value $(r(\boldsymbol{x}_i) - r(\boldsymbol{x}_{i-1}) < 0)$, the agent had learned that the transition will not be helpful to the task, the latest feedback will make conflict with the past feedbacks, so the modifying signal will be restricted. It can be considered that this updating method, showed as Eq.11,12, generates the reward function based on the consistency of human feedback.

On the other hand, if the evaluation is "negative", by applying the following process to $r(X)$:

$$\begin{gathered} if\ \|\boldsymbol{x}_i - \mu_i\| \leq d \\ w_k = 0 \end{gathered} \tag{13}$$

The agent will quickly forget its mistaking knowledge about that area of the state space around X, and restart learning, where d is determined as a threshold of the distance to determine if μ_k is enough close to x_i, d is set same as the standard deviation of RBF unit σ.

2.4 The Basic Problem and the Solution Approach

In this section, we define the basic problem which should be solved by the proposed method (Fig.5). The upper row of Fig.5 shows a situation in which a simple reward function can be defined only around the target state in advance, due to the lack of information about the environment or difficulty of the task. As stated in the introduction, from a reward accessibility point of view, it is clear that this situation is difficult for conventional reward-based learning methods.

By using the proposed method, we consider that updating the reward function according to the progress of the agent in incremental steps is effective in improving the learning efficiency, as shown in the lower row of Fig.5.

Furthermore, this approach is an attempt to integrate the prior design of the reward function into the learning mechanism.

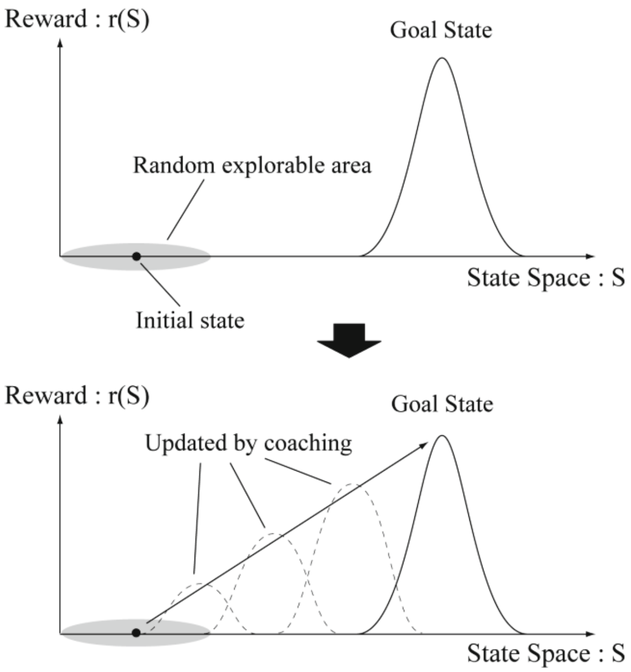


Fig. 5. The basic problem of Coaching. The upper row shows difficult situation for conventional reward-based learning method, and the lower row shows the solution approach by Coaching against this problem.

3 Reward Function and Its Learning Efficiency

In this section, we discuss the learning efficiency according to different reward functions. As described in the previous section, it is difficult to design the reward function in advance because the method for guiding the agent to accomplish the given task is not known. Several methodologies to determine the appropriate reward function have been proposed, but none of them significantly guarantees the task accomplishment even if the robot could theoretically achieve it.

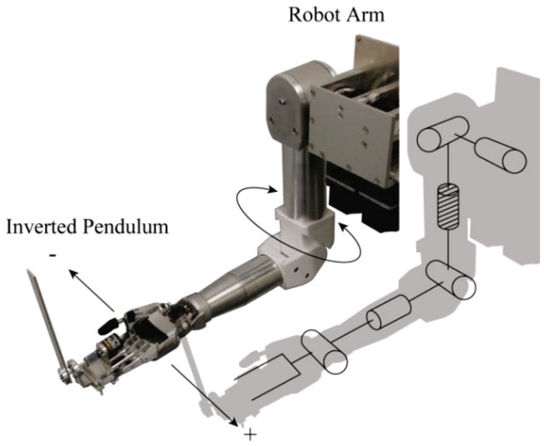


Fig. 6. Experiment setup: 6 DOF Robot Arm Platform. This setup is for both of simulation and real robot experiment.

In order to clarify the learning efficiency of different reward functions, we conduct a simple learning task to swing up and keep balance of an inverted pendulum by 6 DOF robot arm, as shown in Fig.6, in a simulated environment. As a learning agent, conventional RL agent (actor-critic) was implemented in the robot with three different reward functions as illustrated in the upper graph of Fig. 7.

In this experiment, the state space is defined as a 2 dimensional continuous space in terms of pendulum angle θ and angular velocity ω, and the agent can add a bidirectional force to the root of the pendulum as an action output $u(t)$. The experiment procedure is as follows:

1. Set the initial posture of the pendulum as vertically downward.
2. Start the action and measure the total time which the inverted pendulum spends within the target posture range of $\pi \pm \varepsilon$[rad], where ε is a small value.
3. Repeat the above steps for 100 iterations.

The subsequent experiments described in Sec. 4 have been conducted by the same RL agent with the same state-action space.

Three typical examples of reward functions are considered from the view point of different opportunities for obtaining the reward while learning.

1. Reward is given all over the state space (r_1)
2. Reward is given at a part of the state space (r_2)
3. Reward is given solely at a part of the goal state (r_3)

$$r_1(\theta) = (1 + \cos(\theta - \pi))/2 \tag{14}$$

$$r_2(\theta) = \begin{cases} (1 + \cos(2\theta - \pi))/2 & (|\theta| > \pi/2) \\ 0 & \end{cases} \tag{15}$$

$$r_3(\theta) = \begin{cases} 1 & where \quad (\left||\theta| - \pi\right| < \varepsilon) \\ 0 & \qquad else \end{cases} \tag{16}$$

The lower figure of Fig.7 shows the experimental results with a simulated environment. The x-axis represents the number of episode in learning, and y-axis represents the task accomplishment, which is regarded as the total period where the tip of the inverted pendulum stays at the target angle with ± 5[deg]. The target angle is set to the top, and configurations of the simulated robot such as its state-action space and the task are defined same as the experiment in Sec.3.

The number of successful trials increases by using the reward function r_1, but it could not achieve the task at all with the reward function r_3. In the conventional policy of behavior learning, the agent continues to explore over the state space in a random manner until a reward is found. Although this is not a difficult task but it is necessary to find an appropriate reward function such as r_1. The reward function is designed by the system designer, but an appropriate reward function, which helps successful learning, is not always known in advance. The easiness of design and the learning efficiency is a trade-off, and the system designer must carefully determine it for a complicated task. In the Coaching framework, the human trainer is involved in assisting the learning of agent, by observing the agent behavior in order to modify the reward function while the agent learns through a trial-and-error procedure.

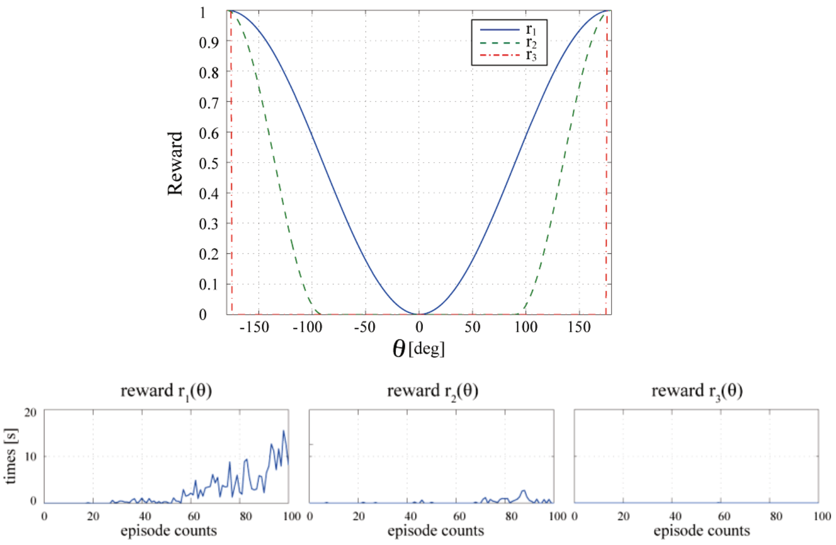


Fig. 7. Three different reward functions $\boldsymbol{r_1(\theta), r_2(\theta), r_3(\theta)}$ used in the learning task of an inverted pendulum (Upper figure), and simulation results (Lower figures)

4 Evaluation of Behavior Learning

4.1 Characteristics of Human Evaluation

When a human trainer gives an evaluation in real time, there will be a time delay between the evaluation timing and the target behavior. Thus, to determine the target behavior of that evaluation, the time delay as a characteristic of human evaluation must be measured. The result of an experiment aiming at measuring the time delay by several subjects is shown in Fig.8.

From Fig.8, we can see that the difference of the average between subjects is only about 100[ms], and the deviation within each subject is small. It is, thus, possible to narrow down the time delay to a constant range of time. Consequently, we defined two kinds of time constants (T_1, T_2) that correspond to the minimum and maximum delay time as below.

$$T_1 = 300[\text{ms}],\ \ T_2 = 800[\text{ms}] \tag{17}$$

By introducing these time constants, searching all the action history in an episode for coaching feedback is avoided. In this study, the time delay is obtained in advance through a preliminary experiment. However, it is possible to embed it in the process in order to obtain the time delay at the initial stage of learning.

In both simulation and real robot experiments, RL agent observed the state and performed its actions with a frequency of 10[Hz]. Therefore, around 5 states are contained in the time range, and the amount of calculation is small enough for online learning.

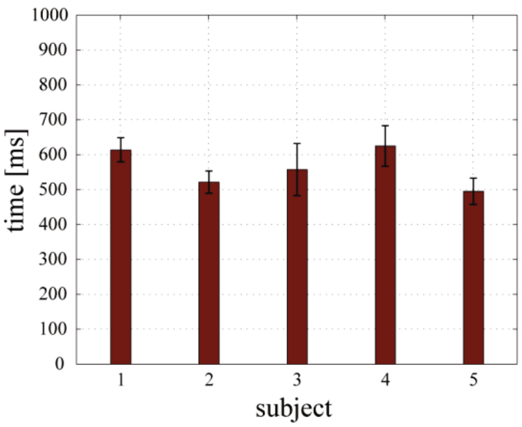


Fig. 8. Result of the experiment to measure the time delay of human evaluation. The procedure is a measurement of the reaction time where the subjects selectively did 2 kinds of reactions for 10 times against 2 kinds of stimulation. Average and standard deviation were calculated for 5 subjects.

4.2 Experiment with a Simulated Robotic Agent

We evaluate the performance of the proposed method through a learning experiment in simulated environment similar to the one used in Sec.3.1. Three subjects participated to this experiment as trainers. The instructions given to the subjects were as follows:

- The purpose of this task is to make the agent learn to keep the balance of the inverted pendulum.
- Give a “positive” or “negative” evaluation if you think that the agent did or didn't do an action that brings it closer to achieving this task.

The initial reward function, denoted r_{init} at Eq.8, is given as:

$$\mathrm{r}(\theta) = \begin{cases} 1 & where \ \ |\theta| = \pi \\ 0 & else \end{cases} \tag{18}$$

This reward function allows the agent to obtain the reward only at the target state. Moreover, it is necessary to move the robotic arm to right and left in synchronization with the oscillation of the tip of the pendulum to explore this state space widely. The probability that such a movement is generated from a random output is very low. Thus, this learning environment is included in the type of the problem described previously (Sec.2.3).

4.3 Results

Fig.9 shows the results of the experiment. The graph, Fig.9(a), corresponds to the case of learning without Coaching. In this case the agent could not learn the desired behavior. On the other hand, when Coaching was used, Fig.9(b,c,d), the agent acquired the knowledge of how to keep balancing the pendulum only when the trainer was subject #1 and #3. The reason why the subject #2 could not make the agent acquire the behavior was that the subject gave too many “negative” evaluation compared to “positive” evaluation. In the proposed algorithm, “negative” evaluation reset the knowledge, hence does not give any information for further learning. This problem is one of the considerations for future work. However, the proposed method successfully improved the learning ability in overall, even when the reward function could only be defined out of the area explorable by the agent.

To evaluate the feasibility of our approach in the simulation experiment, we verified the transition of the reward function modified by subject #3 whose performance was best. Fig.10 shows the reward function at the end of episodes 0, 6, 23 and 64. We can see that the peak of the reward function has been moving from the initial state (indicated by black line in the top-left figure) towards the goal state (indicated by black lines). Consequently, the solution approach we described shown in Fig.5 was achieved. However, Coaching in some cases may not help the robot, as shown by the results of subject #2. It can be said that this is because the robot reflected the difference of the Coaching strategies of trainers. However, regarding the reward function and the state value, it can be said that the proposed method is an effective behavior acquisition method in situations where the conventional methodologies are not very effective.

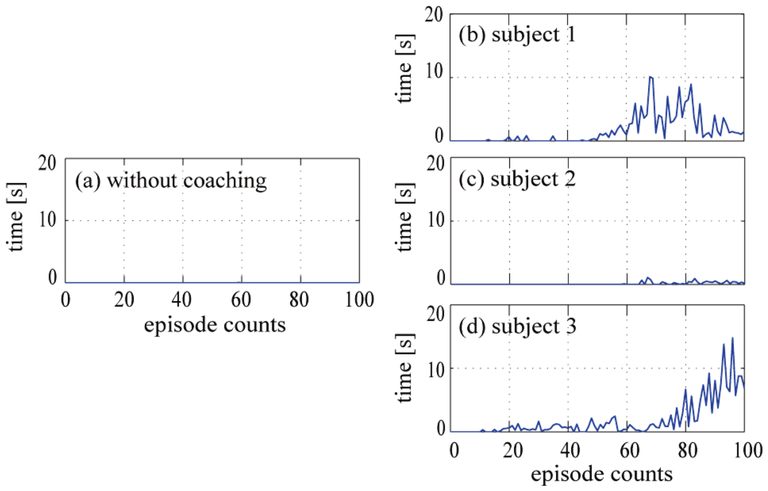


Fig. 9. Results of simulation experiment. The horizontal axis is the number of episodes and the vertical axis is the time the agent could keep balancing.

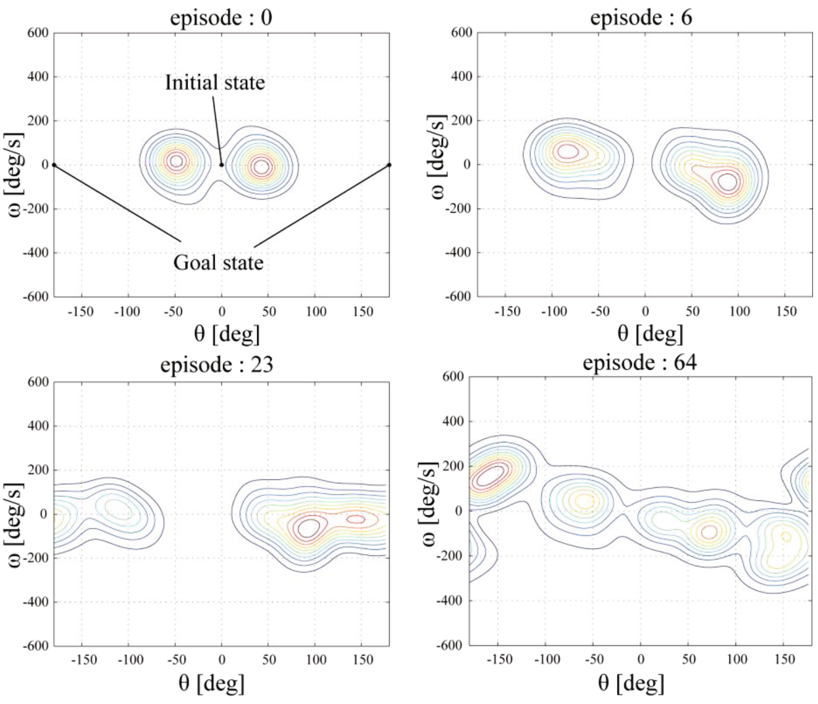


Fig. 10. The transition of the reward function updated by subject #3

4.4 Experiment with a Real Robotic Agent

Finally, an experiment on the same task by using a real robot arm was conducted. The robot arm with 6 degrees of freedom shown in Fig.6 was used. In this study, we controlled only the upper arm joint, which is shown shaded in the figure, to swing the inverted pendulum. The result of this experiment is shown in Fig.11. The proposed method has shown significant improvement compared to the conventional RL with the same reward function, as the latter, completely failed to achieve the task.

We could see the results are similar to those discussed in the previous simulation experiment, in particular, the non-monotonous nature seen during the learning process. In this experiment, the total time in which the agent were able to keep the pendulum upright was less than 1 second due to several limitations of the robot, such as sensors, actuator torque and sampling rate. Note that, we did not attempt to estimate these robot parameters, but just like in the simulation, the agent was able to find a solution to the task. Therefore, it can be said that the proposed method potentially has the capability to accelerate the learning efficiency of RL agent, in particular, in the early stage of learning.

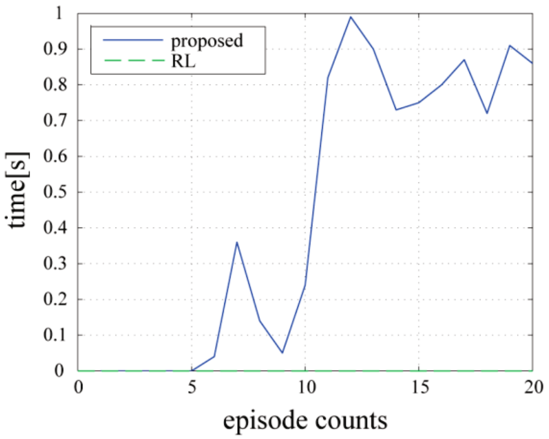


Fig. 11. An example of the experimental results

5 Conclusions

In this chapter, we proposed a novel methodology for behavior learning, called *Coaching*. This method allows a human trainer to intervene in the learning process of an agent by giving subjective evaluations such as *positive* and *negative*. The agent updates its own reward function based on the evaluation which is enhanced by considering the characteristics of human evaluation. Then, we confirmed the effectiveness of the proposed method using both simulated and a real robot.

The learning task which we used for the experiment, inverted pendulum, had been studied well, and various methodologies to achieve the task have been proposed. However, generally, these methodologies include an element of heuristics for customization to each task. In particular, our method has an advantage in designing the reward function, which allows users to modify the function during the learning phase.

Currently, we are working towards the development of a universal interface device for Coaching an agent having real body in a real environment. For future works, we will address the refinement of the learning algorithm to utilize negative evaluations and to verify the robustness with respect to learning parameters.

References

1. Atkenson, C.G., Schaal, S.: Robot learning from demonstration. In: Proc. of 14th Intl. Conf. on Machine Learning (1997)
2. Atkenson, C.G., Schaal, S.: Learning tasks from a single demonstration. In: Proc. of IEEE Intl. Conf. on Robotics and Automation, pp. 1706–1712 (1997)
3. Cypher, A., Halbert, D.C., Kurlander, D., et al.: Watch what I do: programming by demonstration. MIT Press, Cambridge (1993)
4. Doya, K.: Reinforcement learning in continuous time and space. J. Neural Computation 12(1), 219–245 (2000), doi:10.1162/089976600300015880
5. Inamura, T., Toshima, I., Tanie, H., et al.: Embodied Symbol Emergence Based on Mimesis Theory. J. Robotics Research 23(4), 363–377 (2004), doi:10.1177/0278364904042199
6. Jakel, R., Schmidt-Rohr, S.R., Xue, Z., et al.: Learning of probabilistic grasping strategies using programming by demonstration. In: Proc. of IEEE Intl. on Robotics and Automation, pp. 873–880 (2010)
7. Kamatani, H., Kitayama, K., Fujimura, A., et al.: Reinforcement learning in continuous state space. In: SICE Tohoku Chapter Workshop (2006)
8. Marcia, R., Ude, A., Atkenson, C., et al.: Coaching: An Approach to Efficiently and Intuitively Create Humanoid Robot Behaviors. In: Proc. of IEEE Intl. Conf. on Humanoid Robots, pp. 567–574 (2007)
9. Morimoto, J., Doya, K.: Reinforcement learning of dynamic motor sequence: learning to stand up. In: Proc. of IEEE Intl. Conf. on Intelligent Robots and Systems, pp. 567–574 (1998)
10. Nakatani, M., Suzuki, K., Hashimoto, S.: Subjective-Evaluation Oriented Teaching Scheme for a Biped Humanoid Robot. In: Proc. of IEEE Intl. Conf. on Humanoid Robots (2003)
11. Schaal, S.: Is imitation learning the route to humanoid robots? Trends in Cognitive Sciences (1999), doi: 10.1016/s1364-6613(99)01327-3
12. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (1998)
13. Tamosiunaite, M., Asfour, T., Florentin, W.: Learning to reach by reinforcement learning using a receptive field based function approximation approach with continuous actions. J. Biological Cybernetics 100(3), 249–260 (2009), doi:10.1007/s00422-009-0295-8
14. Thomaz, A.L.: Socially Guided Machine Learning. PhD thesis, Massachusetts Institute of Technology, Cambridge (2006)
15. Thomaz, A.L., Hoffman, G., Breazeal, C.: Experiments in Socially Guided Machine Learning: Understanding How Humans Teach. In: Proc. of the 1st Annual Conf. on Human-Robot Interaction, pp. 359–360 (2006)

Chapter 4
Persian Vowel Recognition Using the Combination of Haar Wavelet and Neural Network

Mohammad Mehdi Hosseini and Abdorreza Alavi Gharahbagh

Department of Electronic and Computer Engineering, Islamic Azad University, Shahrood Branch, Shahrood, Iran, P.O. Box 36155/163
{hosseini_mm,R_alavi}@iau-shahrood.ac.ir

Abstract. lips movement is an important parameter in speech recognition. The Lip image segmentation has a significant role in lips movement analysis. Major problems that any vowel recognition (especially Persian) method is faced are low chromatic in lip region, low contrast luminance, overlap between the lip and facial skin color, and similarity between lips movement in some vowels after detecting lips. In this paper, a new automatic and quick approach for the lip extraction based on using the Haar wavelet is proposed. The output of this proposed approach is used as a feature vector for a hybrid neural network. The proposed algorithm for lip image segmentation uses the color space CIE L*U*V* and CIE L*a*b* in order to improve the contrast between the lip and the other face regions. After this step, the lips are modeled and a feature vector with longitudinal and angular parameters is extracted from the proposed lips model.

This feature vector has been used as an input for a feedforward backpropagation hybrid neural network. The proposed method has been applied to 2200 tested images and the accuracy is about 79% that shows about 15% enhancement in compare with similar methods.

Keywords: Haar wavelet, lip reading, neural network, vowel recognition, segmentation.

1 Introduction

Both lips localization and segmentation are important steps in various applications such as automatic speech reading, MPEG-4 compression, special effects, facial analysis and emotion recognition. In this context, lip analysis is useful for verifying speech and lip synchronization in order to minimize the scope for fraudulent access to services controlled by multimodal biometric personal-identity-verification systems. In addition, it has been shown that the lip image analysis can provide a control mechanism for selecting the most appropriate face model (e.g. open mouth or closed mouth) whenever the aim is either face verification or face recognition. However, the lip localization and segmentation in images or videos is a challenging problem owning to unknown face poses, varying lighting conditions, low image qualities, and background clutters. Thereby, extracting lip gesture from the above referred lip image sequence is so important. Various techniques have been developed to achieve a good and robust segmentation. Zhang used hue and edge features to achieve mouth localization and segmentation [1]. Eveno et al [2] detected characteristic points in the mouth region and then used a parametric model to fit the lip. In their study, they proposed preprocessing algorithm for the lip segmentation. They transformed the RGB color space onto the YUV color space at first and used just the "V" component for segmentation. Although they could reduce the processing time, their method is sensitive to the color contrast and the noise. Caplier [3] proposed a method which employed the edge direction for the lip segmentation. He used the active shape model to describe the mouth. Both detecting and tracking the lip contour are important in speech reading. Tracking the lip contour can be done well if the contour initialization has been adjusted correctly at the first frame of the image. Nowadays, the contour initialization at the first frame has been done manually. Therefore, this method has been failed whenever it applied on the real world mouth images [3].

Recently, the statistical methods have been developed to extract face features and particularly the mouth. Coots et al. [4] introduced active shape model (ASM) and active appearance model (AAM). Wang [5] and Liew [6] proposed FCMS[1] and SFCM[2] algorithms for the lip extraction. Both methods integrate the color information along with different kind of spatial information into a fuzzy clustering structure. However, the above methods should be used in indoor situations with controllable lighting condition.

Different methods have been developed to recognize vowels according to the audio and visual features. Neural network has widely been used in many of these methods. Shinchi has used a simple neural network for lip reading [7]. Matthews and Stork have used time delay neural network (TDNN) and recursive neural network for lip reading [8, 9]. Although the wavelet transform has been successfully applied in edge detection, there are few reports on using this method for lip segmentation. The problem arises because the outer labial contour of the mouth has very poor color distinction in compare to its skin background. In this paper, the authors proposed the automatic lip localization and segmentation algorithm based on the captured full-frontal faces with perfect quality. The Lip features extraction has done by Haar Wavelet as input parameters to a neural network system for vowel recognition. The suggested algorithm is based on the Wavelet transform, while the color space, CIEL*U*V* and CIEL*a*b* are used in conjunction with Wavelet transform.

2 Proposed Method

2.1 Lip Localization

The lip localization is performed in two steps. The first step is face detection and the second step is extraction of the lip Region-Of-Interest (ROI) from the image.

[1] Spatial Fuzzy C-Means Clustering.
[2] Fuzzy C-means With Shape Function.

2.1.1 Face Detection

In this study, the developed method by Bayesteh is used for face detection [10]. The proposed algorithm includes two stages and according to color segmentation, the algorithm finds possible areas for face region. Then, it involves face verification by using a cascade classifier. Thereby the face is determined as a rectangle with width W and height H, Fig1.

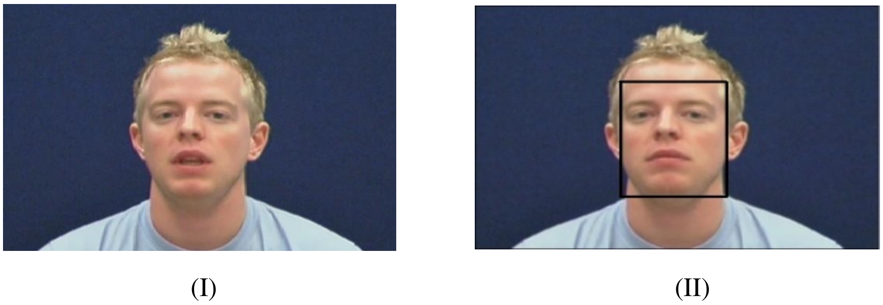


Fig. 1. (I) input image (II) face detector outputs

2.1.2 Lip Region of Interest

After face detection step, the following steps are followed for lip extraction:

- The face is divided into two equal halves, up half and down half. In this step, pupils are located in up half and lip is located in down half.
- The up half is divided to two equal regions, left and right respectively. After that, by using horizontal and vertical histograms, pupils' position can be obtained. The distance between pupils is named ED.
- From the middle part of pupils, a length of 1.2× ED is selected downward, then, with this center a rectangle of length ED and width of 0.7× ED is plotted, Fig2.This method can correct image tilt by using pupil positions and rotating image but any image rotate can be reduced efficiency.

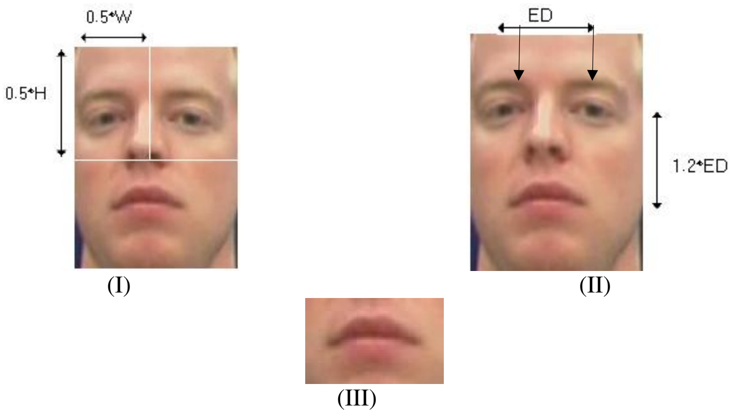


Fig. 2. I) step 1 and 2 II) step 3 III) lip region

2.2 Lip Segmentation

After finding the lip location, the lip segmentation algorithm in five steps is applied, Fig3.

2.2.1 Pre Processing

The lightning condition is changed to normalize the Luminance level in original image by using the following log-function [10]:

$$g(x,y) = k + \frac{\log(f(x,y)+1)}{d \times \log(t)} \tag{1}$$

Where $f(x,y)$ is the original and $g(x,y)$ is the pre-processed image. The set parameters (K, d, t) are adjusted to control the location and the shape of the curve. In this study, (K, d, t) are evaluated as (12, 0.5, 2) respectively.

2.2.2 Color Transform

The colors of the lip and the skin region usually overlap therefore; an especial color space should be chosen to show the small variations. The distance between any two points in color space is proportional to the perceived color difference; therefore, a uniform color space is required. After this step, the RGB image is transformed to the

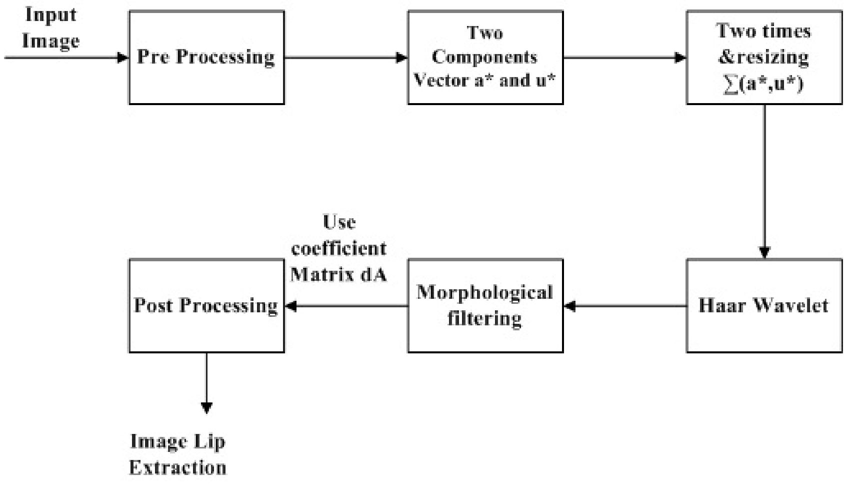


Fig. 3. Block diagram of the lip segmentation algorithm

CIEL*U*V* and CIEL*a*b color space. The vector color $\{L^*, a^*, b^*, u^*, v^*\}$ for any image can be determined by using the following equations [11]:

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} 0.490 & 0.310 & 0.200 \\ 0.177 & 0.813 & 0.011 \\ 0.000 & 0.010 & 0.990 \end{pmatrix} \begin{pmatrix} \mathrm{R} \\ \mathrm{G} \\ \mathrm{B} \end{pmatrix} \tag{1-1}$$

$$L^* = \begin{cases} 116(Y')^{1/3} - 16 & \text{if } Y' > 0.008856 \\ 903.3\mathrm{Y}' & \text{otherwise} \end{cases} \tag{1-2}$$

$$a^* = 500\left(K_1^{1/3} - K_2^{1/3}\right) \tag{1-3}$$

$$b^* = 200\left(K_2^{1/3} - K_3^{1/3}\right) \tag{1-4}$$

$$K_{\mathrm{i}} = \begin{cases} \phi_{\mathrm{i}} & \text{if } \phi_{\mathrm{i}} > 0.008856 \\ 7.787\phi_{\mathrm{i}} + \dfrac{16}{116} & \text{otherwise} \end{cases} \tag{1-5}$$

$$\text{for i} = 1,2,3 \text{ and } \phi_1 = \mathrm{X}' = \frac{X}{X_0}, \phi_2 = Y' = \frac{Y}{Y_0}, \phi_3 = Z' = \frac{Z}{Z_0} \tag{1-6}$$

$$u^* = 13\mathrm{L}^*(\mathrm{u}' - \mathrm{u}_0) \tag{1-7}$$

$$v^* = 13\mathrm{L}^*(\mathrm{v}' - \mathrm{v}_0) \tag{1-8}$$

$$u' = u = 4X / (X + 15Y + 3Z) \tag{1-9}$$

$$v' = 1.5v, v = 6Y / (X + 15Y + 3Z) \tag{1-10}$$

Where X_0, Y_0, Z_0, u_0 and v_0 are the values of X, Y, Z, u and v for the reference white, respectively. The reference white in the transformation is defined as $\{\mathrm{R} = \mathrm{G} = \mathrm{B} = 255\}$. The color vector L^*, a^*, b^*, v^*, u^* is adopted to represent the color information; Parameters $\{a^*, u^*\}$ are used in this study.

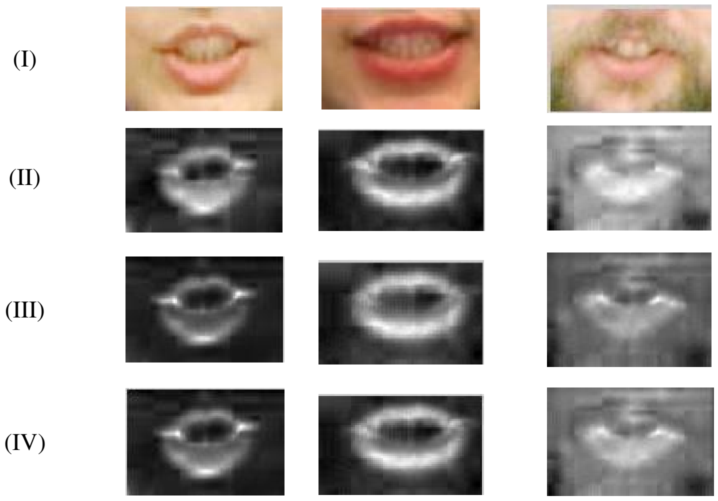


Fig. 4. I) picture of the lips II) vector color a* III) vector color u* IV) output of sum a* and u*

Although in that case there will be more statistical dependence between two vectors $\{a^*, u^*\}$, the main difference between lip and face area is the reddish lip color of all races and in the two selected vectors this color is the most effective. Other vectors usually vary in different races because color of face is different. This assumption is tested in different images and in various scenarios and the selected vectors showed better results. As a result, the other remained parameters are being waived, fig 4.

2.2.3 Wavelet

Unlike the Fourier transform that basic functions are sinusoids and redundant, the wavelet transform is based on short-duration waves, called wavelets, of different frequency and restricted duration. Wavelet is also named multi-resolution transform. This characteristic makes the wavelet transform suitable for joint time-frequency analysis in comparison with the Fourier transform [12]. In this case, the main advantage of using the wavelet is detecting the feature that might not be recognized in the beginning. The two dimensional fast wavelet transform (2-D FWT) can be used for image analysis. In this case, there are four sub-band images in each level. They are referred to dA, dH, dV and dD where, dA, dH, dV, and dD denote the coefficients matrix, horizontal detail, vertical detail, and diagonal detail of a sub-band image, respectively. Please note that the size of every subband images is equal to the half of the input image.

2.2.4 Lip Segmentation

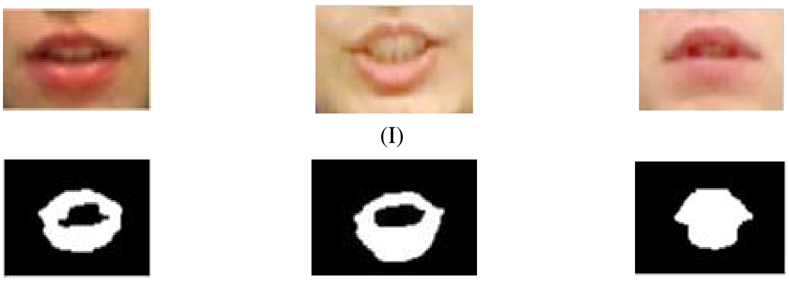
After pre-processing and transforming the image to the CIEL*U*V* and CIEL*a*b color space, the lip segmentation procedure is done as follows:

- The two vector components $\{a^*, u^*\}$ are added to each other and obtained image is resized to be matched with the original image. In the first analysis, a^* and u^* is used separately but the result was so poor. In the second examination, a^* and u^* is used in parallel way, the result of this test was better than the first analysis. Overall, summing a^* and u^* had shown the best result between the above methods.

- The 2-D Haar wavelet transform and product is performed and four different matrixes, (dA, dH, dV, dD), are determined. The dA matrix is sufficient for the lip region extraction, therefore, three other matrixes, i.e. dH, dV, dD, are discarded. After this stage, the morphological filtering and the post processing are employed to increase the accuracy.

2.2.5 Morphological Filtering

In this stage, the Grayscale morphological closing and opening with an 8-neighborhood rectangular structuring element is used to smooth membership map and eliminate small erroneous blobs and holes.



(II)

Fig. 5. (I) original lip image (II) extracted lip region

2.2.6 Post Processing

In this stage the output of previous step should be converted to a binary image. First of all, all image pixels are scanned and preliminary labels are assigned to nonzero pixels and recording label equivalence is restored in a union-find table. Note that resolving the equivalence classes has done by using the union-find algorithm in which reliable pixels are based on the resolved equivalence classes. After this step, a Gaussian filter is used to smooth the image and eliminate some under size points. The final result is shown in fig 5 for different cases.

3 Vowel Recognition

Vowels and consonants are the basic elements of each language. In Persian, there are 6 distinct vowels demonstrated as " اَ، اِ ، اُ ، آ ، او، ایِ "which are fairly similar to English vowels ″ a, e, o, â, i and u ″. In this study, a new method is proposed for the vowel recognition. In this method, some lip features that have been extracted in former sections are applied to an appropriate neural network.

3.1 Feature Vector

To describe the shape of lips, two groups of features can be used. The first group is the features which describe the mouth by using the size of some physical quantities or geometrical concepts that are named as linear parameters. The second group of characteristic is an angular feature that is used in the recognition process. Based on a segmented lip image, the key feature points of the lips are extracted as follow:

- Normalized width (mouth opening in horizontal direction).
- Height (mouth opening in vertical direction).
-Average vertical distance of points 5-3 and 5-9.
- Angle between points corner left and right lips.

Fig. 6 shows the selected feature points in a lip contour.

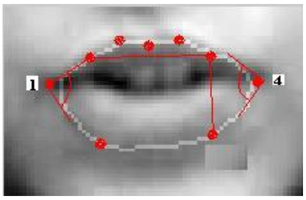


Fig. 6. Mouth feature point

3.2 Neural Network

The neural networks have been widely used as a solution for various problems such as speech classification and recognition [13]. Some neural networks such as Radial basis functions network and feed forward backpropagation network have been tested as classifiers. Several experiments showed that a feed forward backpropagation network that used in binary mode had better performance. Some of RBF limitations in this work in compare with feedforward backpropagation network are as follow:

- RBF networks are local approximators, whereas multilayer perceptrons such as feedforward backpropagation networks are global approximators.
- RBF networks have a single hidden layer, whereas multilayer perceptrons can have any number of hidden layers.
- The output layer of a RBF network is always linear; whereas in a multilayer perceptron it can be linear or nonlinear (this is noticeable in proposed classifier).
- The activation function of the hidden layer in an RBF network computes the Euclidean distance between the input signal vector and parameter vector of the network, whereas the activation function of a multilayer perceptron computes the inner product between the input signal vector and the pertinent synaptic weight vector.

A feed-forward network has a layered structure. Each layer consists of units which receive their input from units of a layer directly below and then send their output to units in a layer directly above the unit. There are no connections within a layer. The N_i inputs are fed into the first layer of N_h hidden units. The input units are merely 'fan-out' units and no processing has been done in these units. The activation of a hidden unit is a function F_i of the weighted inputs plus a bias, as given in the following equation:

$$y_k(t+1) = F_k\left(\sum_j w_{jk}(t)\, y_j(t) + \theta_k(t)\right) \tag{1-11}$$

The output of the hidden units is distributed over the next layer of hidden units, until the last layer of hidden units, of which the outputs are fed into a layer of No output units.

The backpropagation algorithm has emerged as the workhorse for the design of a special class of layered feedforward networks known as multilayer perceptrons (MLP). A multilayer perceptron has an input layer of source nodes and an output layer of neurons (i.e., computation nodes). These two layers connect the network to the outside world. In addition to these two layers, the multilayer perceptron usually has one or more layers of hidden neurons (in this case one). This is because these neurons are not directly accessible. The hidden neurons extract important features that are contained in the input data.

Backpropagation algorithm involves two phases:

1- Forward Phase: During this phase the free parameters of the network are fixed, and the input signal is propagated through the network layer by layer. The forward phase finishes with the computation of an error signal $e_i = d_i - y_i$

Where d_i is the desired response and y_i is the actual output produced by the network in response to the input x_i.

2- Backward Phase: During this phase, the error signal e_i is propagated through the network in the backward direction. During this phase adjustments are applied to the free parameters of the network in order to minimize the error e_i in a statistical sense.

Backpropagation learning may be implemented in one of two basic ways:

1- Sequential mode (also referred to as the on-line mode or stochastic mode): In this mode of BP learning, adjustments are made to the free parameters of the network on an example-by example basis. The sequential mode is best suited for pattern classification such as the proposed work.
2- Batch mode: In this mode of BP learning, adjustments are made to the free parameters of the network on an epoch by- epoch basis, where each epoch consists of the entire set of training examples. The batch mode is best suited for nonlinear regression.

Some methods are used to make backpropagation learning performance better such as:

a- Use of neurons with anti symmetric activation functions (e.g., hyperbolic tangent function) in preference to non symmetric activation functions (e.g., logistic function).
b- Shuffle the training examples after the presentation of each epoch; an epoch involves the presentation of the entire set of training examples to the network.

c- Follow an easy-to-learn example with a difficult one.

d- Preprocess the input data to remove the mean and decorrelate the data.

e- Arrange for the neurons in the different layers to learn at the same rate. This method is implemented by assigning a learning rate parameter to neurons in the last layer that is smaller than those at the front end.

The problem is a multiclass classification, so the one-vs-all method is used in this study. This kind of strategy has been used in machine learning widely with named the entity recognition. A classifier for each character has been assigned in a way that the network has six classifiers. The input of these classifiers is feature vector and the maximum output of these classifiers has been selected as the result.

The chosen network is a two layer neural network with 20 inputs, 25 hidden layer neurons, 6 outputs (corresponding to 6 Persian vowels) and tan-sigmoid activation function. This network had better results than other conditions. Fig.7 shows the neural network structure.

4 Material and Methods

In this work, a suitable data base has been prepared in normal lightning condition and frontal view without any rotation or tilt. 22 persons with different ages between 22 to 45 and usual skin color were asked to utter Persian words in which, each word contained a single vowel. Fig. 7 shows a single frame of each vowel for three different persons. Subsets of 2200 image sequences from the database are used. This includes the 40 monosyllabic Persian words for each speaker. The sizes of image sequences for the frontal views are 512× 384 pixel. The two, three and four layer neural network with 20 inputs, 1 output and a different number of hidden layer neurons in range of 10 to 35 is tested as classifier. Two types of networks has been implemented, feedforward backpropagation structure and redial basis function structure. All conditions such as train and test data and other network parameters are the same. For all networks 80 percent of data is assigned as training and 20 percent as testing. For choosing this data, cross validation algorithm is used. After this step, all networks are tested by a new data set with 45 random image sequences selected from original database. Persian language had 6 classes of vowels, for this reason 6 parallel neural networks is used (for each vowel a different neural network is trained). All networks have similar structure and nodes but the activation functions can be different. At the end, the chosen network is a two layer neural network with 20 inputs, 25 hidden layer neurons, 1 output (at final step the system selected maximum output of 6 networks corresponding to 6 Persian vowels) and tan-sigmoid activation function. This structure had better results than other conditions. Fig.8 shows the selected neural network structure.

It should be notified that the speed of the proposed algorithm and accuracy of the results are better than other methods because classifiers do not conflict with each other and each classifier train immediately in the training process.

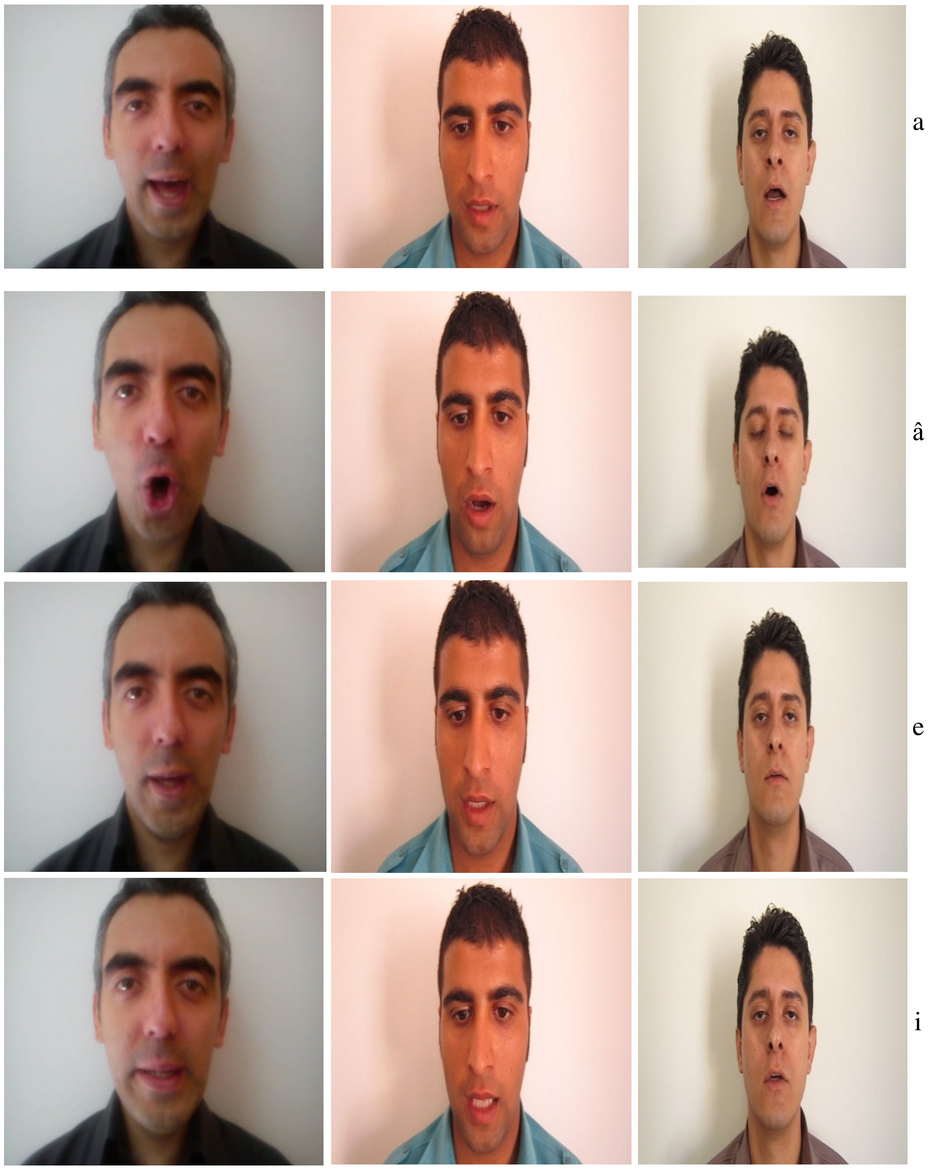


Fig. 7. Single frames of each vowel for three different persons

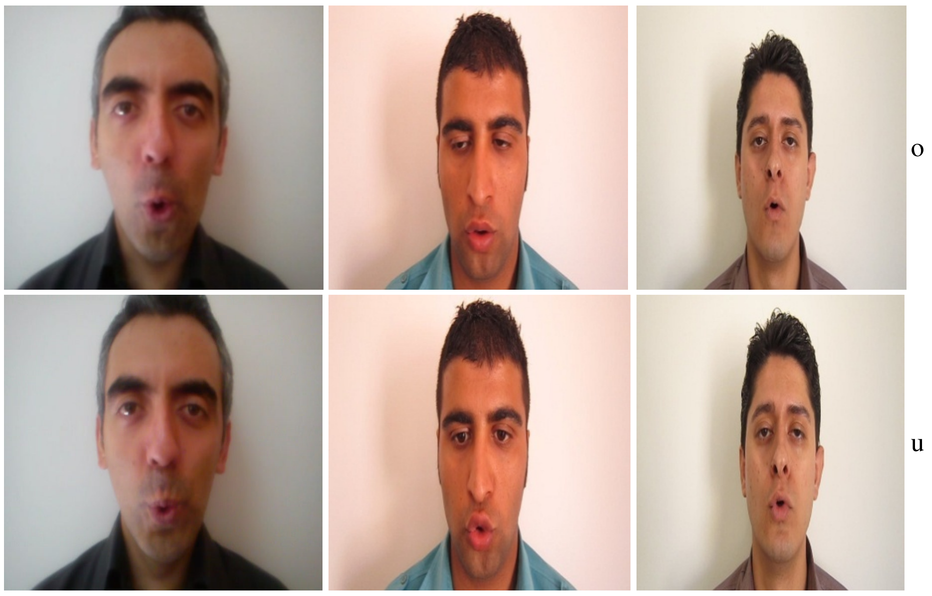


Fig. 7. (*continued*)

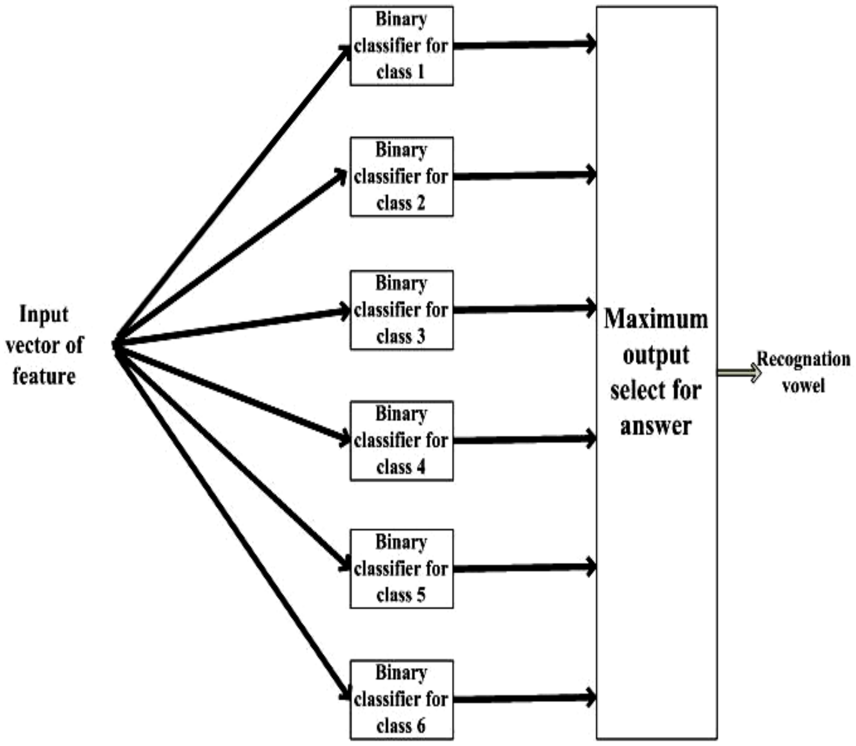

Input vector of feature
Binary classifier for class 1
Binary classifier for class 2
Binary classifier for class 3
Binary classifier for class 4
Binary classifier for class 5
Binary classifier for class 6
Maximum output select for answer
Recognation vowel

Fig. 8. Structure Neural Network used for Classification

5 Results

The proposed network has been trained and tested in three different conditions. In the first condition, the longitudinal parameters are used as input vectors to determine vowels. In the second stage, a combination of longitudinal and angular parameters are used as input vectors and in the third stage, a combination of two above network outputs are used for determining each class.

Obtained Results from these structures are shown in table 1, 2, 3. Results in table 1 are poorer in compare with other tables because visual characteristics in this table are involved simultaneously in vowel formation. The results in table 2 show higher accuracy.

By examining the results in Tables 1 and 2, it can be seen that high accuracy of vowel " اِ (e) , اُ (o) ,اَ (a), ای (i) " is obtained by using simultaneously longitudinal and angular parameters while the status of the remaining vowel recognition (" آ (â) " and "او (u) ") is more appropriate by using only the longitudinal parameters. This problem can be attributed to the type of boundary changes that is made by mouth during the vowel Pronunciation. To increase accuracy and reduce false results, above networks are merged as one network that works with two different modes. The first mode only uses longitudinal features and second mode uses longitudinal and angular features simultaneously. The output of these modes combined as the result. Figure 9 shows the block diagram of implemented model.

As it can be seen in table 3, by using proposed method the accuracy increases in vowel recognition. The total accuracy in the proposed method is 79% . In addition, It can be seen that the best results of vowel recognition corresponded to " اَ (a) "and " او (u) " is because these vowels make the most visual changes on the mouth. The worst results are corresponded to the vowel of " اِ (e)" and "ای (i) " and that is

Table 1. The input vowel and number of vowel recognitions when longitudinal parameters are used

Input / Vowel Recognition	اَ (a)	(e) اِ	(o) اُ	(â) آ	(i) ای	او (u)
(a) اَ	**70**	2	1	22	0	5
(e) اِ	2	**61**	27	0	1	9
(o) اُ	3	5	**72**	2	0	18
(â) آ	0	0	25	**70**	0	5
(i) ای	0	4	30	0	**65**	1
(u) او	3	0	13	4	0	**80**

because of least visual changes. Sadeghi [14] has done similar work for Persian vowels and obtained accuracy of about 64.4%. Their proposed algorithm was evaluated by employing it in recognition of 6 main Persian vowels. Shinchi [7] has done works with accuracy of 70% in Japanese vowels. He has classified 5 Japanese vowels uttered by 2 persons. While the process burden in our method is not changed significantly in comparison with referred methods, the performance in the proposed method is improved.

Table 2. The input vowel and number of vowel recognitions when longitudinal and angular parameters are used simultaneously

Input / Vowel Recognition	اَ (a)	(e) اِ	(o) اُ	(â) آ	(i) ای	او (u)
(a) اَ	**85**	0	0	10	0	5
(e) اِ	4	**67**	22	0	0	6
(o) اُ	1	5	**64**	0	0	30
(â) آ	0	0	22	**78**	0	0
(i) ای	0	2	28	0	**70**	0
(u) او	2	0	22	3	0	**73**

Table 3. Experiment for classes

Input / Vowel Recognition	اَ (a)	(e) اِ	(o) اُ	(â) آ	(i) ای	او (u)
(a) اَ	**89**	0	0	11	0	0
(e) اِ	0	**70**	21	0	0	9
(o) اُ	0	0	**79**	0	0	21
(â) آ	0	0	20	**80**	0	0
(i) ای	0	0	28	0	**72**	0
(u) او	0	0	15	0	0	**85**

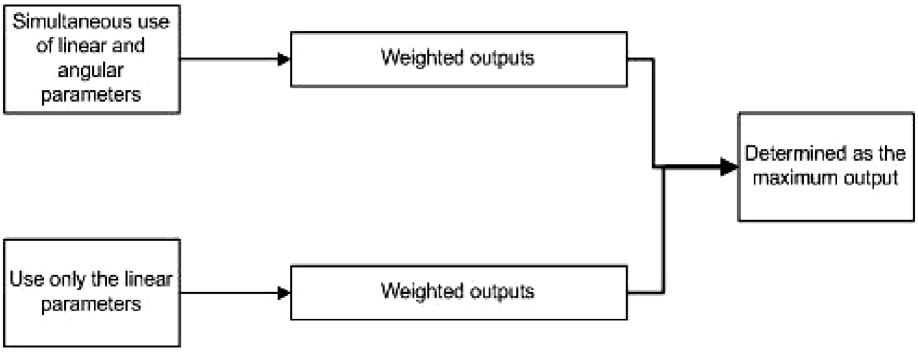

Simultaneous use of linear and angular parameters
Weighted outputs
Determined as the maximum output
Use only the linear parameters
Weighted outputs

Fig. 9. Proposed algorithm to determine output

6 Conclusion

It has been demonstrated that incorporating into an automatic speech recognition system can significantly improve the performance, especially in noisy environment. It is known that extracting a lip automatically from a low contrast image, for example in a movie frame, is a difficult job and very important in vowel recognition. In this paper, a lip segmentation approach based on using the wavelet across the ‹a› and ‹u› component of the CIEL*U*V* and CIEL*a*b color space is proposed. The proposed lip segmentation algorithm is able to exploit the spatial interactions between neighboring pixels through using the wavelet. In particular, our algorithm produces better segmentation without obligation of determining the optimum threshold for every face images. After this exact segmentation, a suitable feature vector extracted from segmented lips. This feature vector is used as input for a feedforward backpropagation hybrid neural network to determine Persian vowels.

The proposed method has been simulated and applied on 2200 tested images. The accuracy of results for proposed structure is about 79% that shows about 15% enhancement in compare with similar methods.

References

1. Zhang, X., Mersereau, R.M.: Lip Feature Extraction Towards an Automatic Speechreading System. In: ICIP, Vancouver, BC, Canada (2000)
2. Eveno, N., Caplier, A., Coulon, P.Y.: A Parametric Model for Realistic Lip Segmentation. In: International Conference on Control, Automation, Robotics and Vision, ICARCV (2002)
3. Caplier, A.: Lip detection and tracking. In: Proceeding of the 11th International Conference on Image Analysis and Processing, Palermo, Italy, pp. 8–13 (2001)
4. Cootes, T.F.: Statistical Models of Appearance for Computer Vision. Technical report (2004), `http://www.isbe.man.ac.uk/bim/refs.html`
5. Wang, S.L., Lau, W.H., Leung, S.: Automatic Lip contour extraction from color image. Pattern Recognition, 2375–2387 (2004)

6. Liew, A.W.C., Leung, S.H., Lau, W.H.: Lip contour extraction from color images using a deformable model. Pattern Recognition, 2949–2962 (2002)
7. Shinchi, T., et al.: Vowel recognition according to lip shapes using neural networks. In: Proc. of IEEE, Tottori, pp. 1772–1777 (1998)
8. Matthews, L.: Extraction of Visual Feature For Lipreading. IEEE Transaction on Pattern Analysis and Machine Intelligence 24(2), 198–213 (2002)
9. David Stork, G., et al.: Neural Network Lipreading System for Improved Speech Recognition. In: Pro. IJCNN, pp. 285–295 (1992)
10. Bayesteh, A., Faez, K.: Boosted Bayesian kernel classifier method for face detection. In: Proceeding of Third International Conference on Natural Computation, ICNC, pp. 533–537 (2007)
11. Hosseini, M.M., Ghofrani, S.: Automatic Lip Extraction Based On Wavelet Transform. In: IEEE, GCIS, China, pp. 393–396 (2009)
12. Mallat, S., Zhong, S.: Characterization of Signals from Multiscale Edges. IEEE Transaction on Pattern Analysis and Machine Intelligent 14(7) (July 1992)
13. Yuhas, B.P., et al.: Neural network models of sensory integration for improved vowel recognition. In: Proc. IEEE, pp. 1658–1668 (1988)
14. Sadeghi, V.S., Yaghmaie, K.: Vowel Recognition using Neural Networks. IJCSNS International Journal of Computer Science and Network Security 6(12), 154–158 (2006)

Chapter 5
The Reproduction of the Physiological Behaviour of the Axon of Nervous Cells by Means of Finite Element Models

Simona Elia and Patrizia Lamberti

Dept. of Electronic and Computer Engineering, University of Salerno, Italy
{selia,plamberti}@unisa.it

Abstract. This paper describes 3D Finite Element modelling solutions for a segment of a nervous cell axon, which take into account the non linear and time varying dynamics of the membrane surrounding it in order to reproduce its physiological behaviour, in terms of Action Potentials (AP) elicitation and its temperature dependence. The axial-symmetry of the system is exploited in order to conduct a more efficient analysis. A combination of the so called Hodgkin-Huxley equations modelling the dynamics of the membrane voltage-controlled ionic channels, together with the Maxwell equations in Electro Quasi-Static approximation, describing the electromagnetic behaviour of each medium, is tackled in a numerical procedure implemented in a commercial Finite Elements multiphysical environment. The usefulness of Finite Elements in order to have interesting quantitative responses (field shape and axon physiological behaviour) is investigated. Two different models are presented here. One exploits the typical thin layer approximation for the axon membrane, proving to be useful when the field solution inside the membrane domain is not a matter of interest. Its performances are compared with the other model, which is introduced in order to obtain a more realistic representation of the studied system: the axon membrane is here realized with a non-linear active medium (exploiting its equivalent electric conductivity) allowing the reproduction of the electric potential also inside the membrane. The passive electrotonic nature of the membrane and the elicitation of an AP in presence of different stimuli are computed and the results are in keeping with the predicted ones. Finally the AP temperature dependences and the propagation effect are reproduced by using the corresponding "best" numerical model, i.e. the coarse one without membrane for the temperature, the more detailed with membrane for the propagation, leading to a trade off between the computational effort and the objective of the analysis. The models open a wide range of applications and extensions in order to understand the true behaviour of a complete neuron.

Keywords: Axon, Neuron, Hodgkin-Huxley equations, FEM.

Introduction

Computational modelling and analysis in biology and medicine have received major attention in recent years in order to understand, analyze and predict the complex mechanisms of biological systems [1]-[3]. Neural prosthetics can considerably widen the lifespan and health quality of people and thus the mechanisms of neuron firing and transmission of signals are increasingly investigated [1]. In order to study the influence of electrical signals on the nervous cells for setting appropriate stimulation protocols and to design efficient equipment, proper models are needed, capable of describing the phenomena occurring at the interface between neural cells and stimulating electrodes [4]-[7].

In order to obtain a model able to predict cell response phenomena the nearest possible to the reality when they are "artificially" induced by external stresses, the first step is to test its capability to reproduce the "natural" phenomena. i.e. the physiological behaviour of the axon. In this paper a realistic cylindrical shaped segment of a nervous cell axon (the neuronal structure carrying nervous signals) is considered. Its electromagnetic physiological behaviour, at different temperatures, is studied by using two alternative 3D models, both implemented exploiting the Finite Element Method (FEM) in axial-symmetry for the solution of the Maxwell Equation in its Electro Quasi Static (EQS) formulation, coupled with the so called Hodgkin-Huxley (HH) equations [8]. This problem is typically studied by using compartmental models for the axon behaviour ([9]-[11]) and solved by means of the cable theory that doesn't furnish the shape of the electromagnetic field around and inside the axon. As a consequence a direct quantitative study of the interaction of the neuron with the applied electric stimulus, the dependences of the responses on the geometric features of the electrode and the influence of the electromagnetic characteristics of the external medium are not easy to obtain or also sometimes even impossible to achieve. Vice versa a "field solution" of the problem leads to the availability, for example, of the voltage profile inside all the modelled domain [12]. Therefore it can be useful, in order to understand the interaction of the axon with other neighbouring structures, such as another neuron. In that case some interesting results can be obtained, capable to take in to account, for example, the ephaptic effect among axon fibers or between them and somas or dendrites [13]. To this aim, the lumped-circuit quantities of the HH electrophysiological model can be transformed into parameters adapt to an electromagnetic field solution study, as in [14]. Thus here, the EQS formulation of the Maxwell equations, describing the relevant phenomena, is faced and the non linear differential equations describing the membrane behaviour are efficiently and accurately combined with the FEM solution in a numerical procedure performed by using a commercial software (COMSOL Multiphysics$^{®}$), implementing their effects by means of opportune outward normal current densities on the boundary. This approach leads to a coarse FEM model of the axon that, by exploiting the concept of thin-layer approximation, is useful and computationally efficient when the field solution inside the membrane domain is not matter of interest. Moreover the possibility to model the axon membrane is also considered here: the axon membrane domain is instead realized with a non-linear active medium (exploiting its equivalent electric conductivity) in order to obtain a more realistic model of the axon. This approach leads to a different FEM model of the axon allowing the reproduction of the electric potential also inside the membrane. This way, the electric potential and its variation along all the modelled region is given with a corresponding computational cost, lower for the first model, higher for the second one. A comparison of the two models is performed in order to assure their coherence with literature specific behaviour as it concerns the correct reproduction of the passive membrane electrotonic response and the elicitation of APs in presence of different stimuli. In this paper, to assure the solution the ability to reproduce the physiological behaviour at different environmental conditions, as experimentally found by HH, the temperature-dependency of the voltage-controlled ionic channels is also introduced in our FEM models, as reported by HH [15]-[17], to accurately describe channels activation/ inactivation properties. The efficiency of the FEM approach with respect to variation in the quality of the numerical solution is performed by comparing the predicted behaviour with respect to the experimental one given in [17]. Finally the AP temperature dependences and the propagation effect are reproduced by using the corresponding "best" numerical model, i.e. the coarse one without membrane domain analysis in temperature, the more detailed with membrane for the reproducing propagation phenomenon, leading to a trade off between the computational effort and the objective of the analysis. Due to their simple implementation, the proposed models can be easily used to simulate the behaviour of more complex nervous structures. The paper is organized as follows: in Section 1 the two alternative model are introduced starting from the more realistic one (Model A), that reproduces the presence of the membrane; it is, then, followed by the introduction of the coarse one (Model B), that is obtained by considering the thin-layer approximation of the membrane domain. Here a comparison between the two models is conduced with respect to assigned input current, leading to assure the capability of both models to reproduce the electrotonic behaviour and the AP elicitation. The Section 2 is dedicated to the introduction of the temperature dependency in the physics of the membrane and used to test the capability of a FEM approach to predict the behaviour with respect to variation in its "numerical" quality. Finally in the Section 3 the AP temperature dependences and the propagation effect are reproduced by using the corresponding "best" numerical model, leading to the conclusion reported at the end of the paper and opening a wide range of future work, there indicated too.

1 Proposed FEM Axon Models

The schematic structure of an axon segment of nerve cell surrounded by its membrane (or axolemma) is pictured in Figure 1.

It is a tubular shaped structure, delimited by three cylindrical surfaces, to delimitate three different domains: the axoplasm, the axolemma and the extracellular domain. The first one is the internal part of the neuron consisting of cytosol, the second one, which is representative of the membrane of the axon, is the domain in which the exchange of ionic charges (Na^+, K^+ and leakage) determines the active electromagnetic behaviour of the neuron and it is delimited between the two first coaxial cylindrical surfaces with a very thin thickness (~nm); in the end, the third domain, representing the external medium in which the neuron is immersed, characterized by electromagnetic properties similar to the first domain, is delimited by the second and third coaxial cylindrical surfaces, with a thickness of the order of the micron. Due to the evident axial symmetry, the three dimensional axon segment of the nerve cell can be studied by considering only the pink-highlighted two dimensional section in Figure 1 and by modelling the system in a cylindrical coordinates system. It is thus possible to pass from the three 3D domains represented in Figure 1 to the three 2D rectangular domains in the more realistic model (i.e. Model A, reported in Figure 2a) or the two 2D rectangular domains divided by a discontinuity boundary in the computationally-reduced model (i.e. Model B, reported in Figure 2b), in which the thin layer approximation of the membrane domain is used [18].

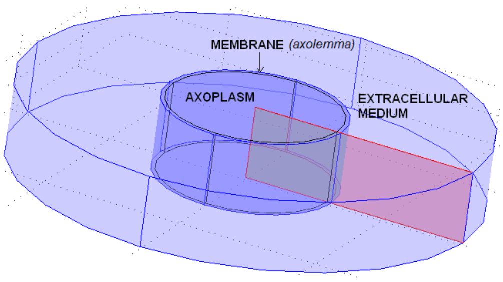


Fig. 1. The axon slice under analysis (3D sketch). The section in r-z plane is highlighted in pink.

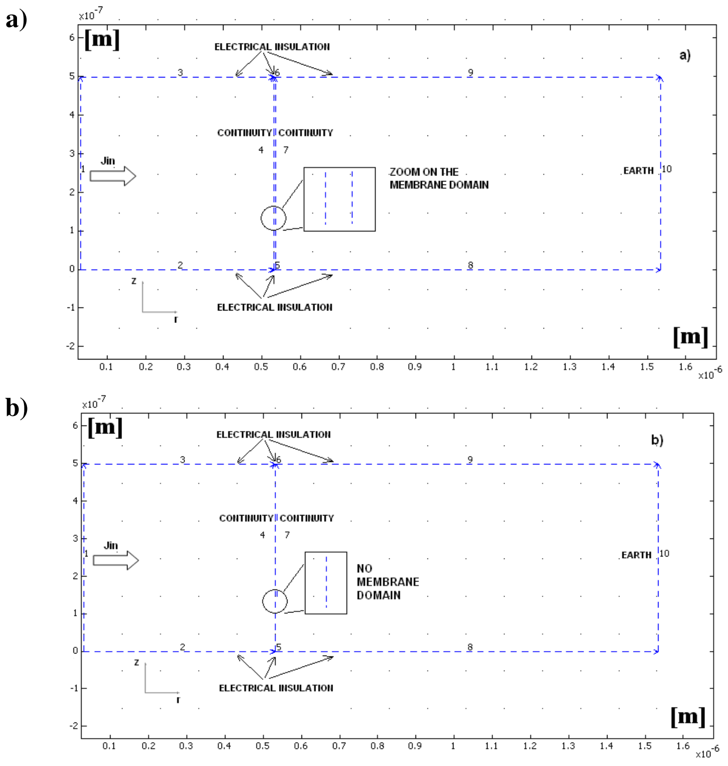


Fig. 2. Axial-symmetric 2D section in r-z plane, with boundary conditions chosen: a) Model A; b) Model

1.1 Model A – Axon with Membrane Domain

The basic geometry used to model the axon, in this case, is reported in Figure 2a. In particular, we model a section of 1.505×0.5μm^2 (0.5μm×0.5μm for the axon domain, D_a, 5nm×0.5μm for the membrane domain, D_m, and 1μm×0.5μm for the external medium represented by D_e). The small size of the system with respect to the characteristic wavelength of the typical electromagnetic fields over the whole structure and the low contribution of the energy associated to the magnetic field compared to that stored in the electric field allow the adoption of the Electro QuasiStatic (EQS) approximation of Maxwell equations. Therefore on this geometry the physiological behaviour of the whole system is obtained by using the 2D axial symmetric transient analysis set of equation of the Quasi-Static Electric AC/DC module, the time dependent analysis of the Partial Differential Equation (PDE) mode and the extrusion tool, offered by the adopted software environment: COMSOL Multiphysics. Sub-domains D_a and D_e are implemented as linear, homogeneous and isotropic dielectric materials, described by their constant electric conductivity, σ_a and σ_e, and dielectric permeability, ε_a and ε_e respectively. The corresponding values are reported in Table 1. On D_m, besides a constant permittivity ε_m, a non linear equivalent conductivity σ_m defined by (2) and an external current density depending on the voltage across the membrane are used in order to approximate the nonlinear behaviour of the medium with respect to the imposed electric field (according to the HH model of the membrane). In particular, HH circuit-equations must be "converted" to obtain their electromagnetic *field equivalent* [19].

First of all, since membrane thickness is very small, it can be looked at as a parallel plate capacitor. Therefore its dielectric and equivalent conductivity can be derived from values found in literature [8]. In particular, once defined all the constant parameters as in Table 1, the dielectric constant per unit area is

$$\varepsilon_m = \frac{C_m d_m}{\varepsilon_0} \tag{1}$$

whereas membrane equivalent conductivity σ_m can be derived by HH overall membrane conductance, G_m, defined as a function of the Sodium (*Na*), Potassium (*K*) and Leakage conductances (*l*) [16] and depending on transmembrane voltage (TMV) through the so called channel activation variables. Then, σ_m becomes:

$$\sigma_m = G_m \cdot d_m \tag{2}$$

where

$$G_m = \sum_i G_i \quad \text{with } i \in \{Na, K, l\} \tag{3}$$

Table 1. Parameters appearing in the model

Parameter	**Value**	**Description**
ε_m	5.65	Membrane relative dielectric constant
C_m [μF/cm^2]	1	Membrane capacitance per unit area
d_m [nm]	5	Membrane thickness
G_{Namax} [mS/cm^2]	120	Conductance per unit area of the Na channel
G_{Kmax} [mS/cm^2]	36	Conductance per unit area of the K channel
G_l [mS/cm^2]	0.3	Conductance per unit area of the leakage channels
E_{Na} [mV]	55	Nernst voltage due the Na concentration
E_K [mV]	-72	Nernst voltage due the K concentration
E_l [mV]	-49.38	Nernst voltage due other ionic concentrations
σ_{Ax} [S/m]	0.5	Axoplasm conductivity
ε_{Ax}	80	Axoplasm dielectric constant
σ_{Ext} [S/m]	1	External medium conductivity.
ε_{Ext}	80	External medium dielectric constant

The expressions of ionic channel conductances, reported in (4.a) and (4.b) show their connection with the activation/inactivation variables *m*, *n* and *h*, implicitly defined by the differential equations set (5):

$$G_{Na} = G_{Na\max} m^3 h \tag{4.a}$$

$$G_K = G_{K\,max} n^4 \tag{4.b}$$

$$\frac{dx}{dt} = \alpha_x \cdot (1 - x) - \beta_x \cdot x \tag{5}$$

where $x \in \{m,n,h\}$.

The transfer rate coefficients α_x, β_x, in (5), are not constant numbers but, as shown in Table 2, depend on the value of the voltage across the axon membrane – the so-called transmembrane voltage, TMV here defined as $V_m(x,y,z,t)=V_m(r,\phi,z,t) = V_m(r,z,t)$ – through *V'*, which represents the TMV deviation from the resting value V_{stat}=-60mV.

Table 2. Expressions of the transfer rate coefficients and corresponding initial values [1/s]

	α_x		β_x	
x	**expression**	**initial value**	**expression**	**initial value**
n	$1000\frac{0.1-0.01V'}{e^{(1-0.1V')}-1}$	*58.19*	$1000\frac{0.125}{e^{0.0125V'}}$	*125*
m	$1000\frac{2.5-0.1V'}{e^{(2.5-0.1V')}-1}$	*223.56*	$1000\frac{4}{e^{(V'/18)}}$	*4000*
h	$1000\frac{0.07}{e^{0.05V'}}$	*70*	$1000\frac{1}{e^{(3-0.1V')}+1}$	*47.42*

The HH trans-membrane current density equation for a unit area patch of membrane can be expressed as:

$$I_m = C_m \frac{dV_m}{dt} + \sum_i (V_m - E_i)G_i \ \text{ with } i \in \{Na, K, l\} \tag{6a}$$

Separating the term depending on V_m and the one depending on the Nernst potentials, I_m becomes

$$I_m = C_m \frac{dV_m}{dt} + V_m \sum_i G_i - \sum_i E_i G_i = C_m \frac{dV_m}{dt} + V_m G_m - J_e \tag{6b}$$

with *i∈{Na,K,l}*. J_e is modeled as an externally impressed current, taking into account the Nernst potential, and is defined as

$$J_e = \sum_i E_i G_i \ \text{ with } i \in \{Na, K, l\} \tag{7}$$

Furthermore, the equation of continuity implemented everywhere over the FEM model can be written as

$$\nabla \cdot \frac{\partial(\varepsilon_i \nabla V)}{\partial t} + \nabla \cdot \left(\sigma_i \nabla V - \bar{J}_{ext}\right) = 0 \tag{8}$$

where

$$\bar{\mathrm{J}}_{\mathrm{ext}} = \begin{cases} 0 & \text{over } \mathrm{D_a} \cup \mathrm{D_e} \\ Je \cdot \hat{r} & \text{over } \mathrm{D_m} \end{cases} \tag{9}$$

The continuity equation (8) must be implemented on the whole model, whereas the HH equations system is associated only to the membrane domain $\mathrm{D_m}$. As the three voltage-controlled conductances G_{Na}, G_K and G_l are meaningful *only on membrane domain* and not externally, they require to be *only locally* defined. The flexibility of COMSOL Multiphysics® proves useful in handling variables, as well as in the post-processing phase. In the simulation session a set of *PDEs* is coupled to the Electrostatic module: the first one is employed in order to solve equation (8) with respect to the so-called dependent variable (in this case *electric potential*, *V*), whereas the second one is introduced to solve the three differential equations in *m, n, h (dependent variables)*, representing channel activation variables according to the HH model ([8]), as shown in the equations reported in system (5).

In order to obtain the voltage values along both sides of membrane, point by point along the z coordinate, the "extrusion" feature is conveniently employed. n fact, the equations implemented there, explicitly depend on TMV, $V_m(r,z\ t)$:

$$V_m(r,z,t) = V_i(r,z,t) - V_o(r,z,t) \tag{10}$$

where V_i and V_o are the voltage across the boundaries 4 and 6, respectively (Figure 2a). This way the HH lumped-circuit quantities are "translated" into parameters adapt to a field solution study, as previously highlighted. It must also be noticed that, while ε_m obtained is a constant, σ_m depends on $V_m(r,z,t)$.

1.2 Model B – Axon with Thin Layer Approximation

The basic geometry used to model the axon in this case is reported in Figure 2b. In particular, we model a section of 1.5×0.5μm^2 (0.5μm×0.5μm for the axon domain, D$_a$, and 1μm×0.5μm for the external medium represented by D$_e$). Also in this case, the physiological behaviour of the whole system is obtained by using the 2D axial symmetric transient analysis packet of the Quasi-Static Electric AC/DC module whereas the time dependent analysis of the PDE mode packet adopted is in weak form and the possibility to perform a thin layer approximation ([18]) is exploited in order to have a computationally less costly model, as in [14]. In fact, the cell membrane is an extremely thin structure that increases the simulation time and memory request in FEM. This applies to the short axon segment under analysis and it is especially true in the perspective of a generalization of the model to a whole axon. Indeed, if it were necessary to simulate the behaviour of a very long neuron (i.e. a motoneuron), this would result in a form factor (length of the axon divided by membrane thickness) that could also be of the order of 10^9. In order to simplify meshing and to greatly reduce simulation time and memory request it is useful to employ a thin layer approximation for the membrane. It is, thus, possible to completely avoid the physical realization of the corresponding thin domain, by substituting it, with an interface surface. This leads to an alternative model, B (Figure 2b), that completely satisfies the hypotheses of applicability of the approximation:

1) there is a substantial difference between membrane domain conductivity and those of the other two domains;

2) lateral boundaries are insulated (null net flux);

3) current density components along ϕ and z are negligible with respect to that along r-axis.

In particular, it is possible to approximate the potential distribution along the membrane thickness as being linearly varying from V_o to V_i. Thus, by using the continuity equation for the current, it is easy to derive the expression for an equivalent current density J_{eq} [14]:

$$J_{eq} = \sigma_m \frac{(V_2 - V_1)}{d_m} + J_e + \frac{\varepsilon_m \varepsilon_0}{d_m} \frac{\partial (V_2 - V_1)}{\partial t} \tag{11}$$

where V_1 and V_2 represent the voltage values along the membrane boundaries 4 and 7 of Figure 2, respectively. This equation can be implemented by using two different Electrostatics packets in order to allow the solver to "see" interface surface (substituting the membrane domain of the model A) once as belonging to axoplasm D$_a$ and once to the external medium domain D$_e$. It is clearly expectable that voltage on that boundary will have a discontinuity (V_2-V_1) almost equal to the value that the TMV would have reached, if the membrane were really implemented in the model as a 2D domain. Thus, V_1 is set as an *active* variable only in the axoplasm domain, V_2 only on the external medium domain, while both are defined on their interface. J_{eq} is imposed as an input current density on this boundary, as in [14]. In addition, an alternative formulation of the three non linear differential equations reported in (5) must be provided on this surface where all expressions are locally defined. The idea is to substitute the *PDE*s formulation, adopted in the version A of the model, with its *weak formulation (for boundaries)*. This choice allows to handle all the equations in the integral form, multiplying both sides of each equation by a test function and then integrating.

1.3 Model A vs. Model B

In order to make a fair comparison between the two modelling solutions, some FEM common parameters are adopted as reported in Table 3.

Table 3. Parameters used for comparing the two models

Calculus and mesh parameters	Value
Simulation times [s]	$0:10^{-4}:0.02$
Relative tolerance	10^{-4}
Absolute tolerance	10^{-8}
Max. element size scaling factor	1
Element growth rate	1.3
Mesh curvature factor	0.3
Mesh curvature cut off	0.001

The same initial and boundary conditions are fixed everywhere, exception made for the various settings related to membrane domain, since it is not present in the Model B. This settings induce the meshes pictured in Figure 3 in which the increase is evident in the number of elements due to the presence of a thin structure, represented by the membrane domain in the Model A (Figure 3a), if compared with the Model B (Figure 3b).

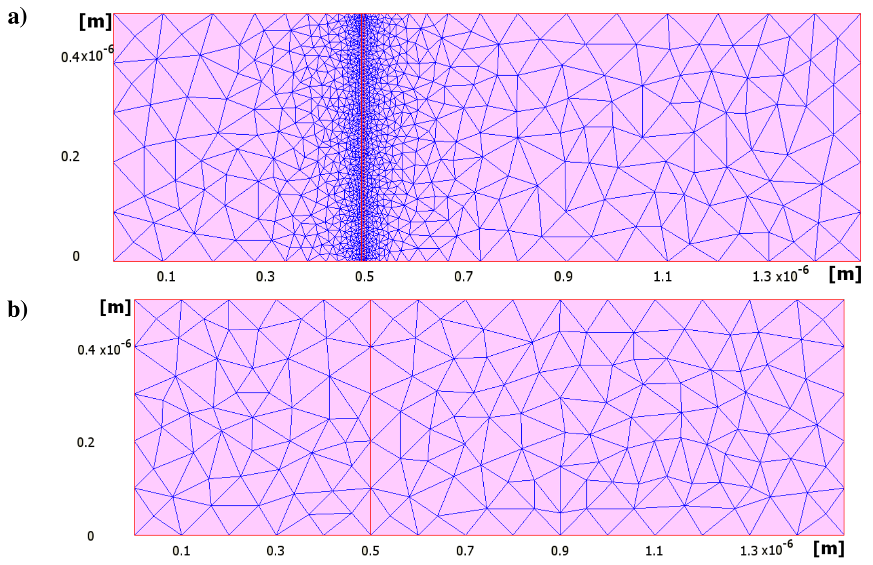


Fig. 3. Meshed 2D geometry: a) model A with membrane, b) Model B without membrane

The simulation is carried out, by fixing all initial conditions from nominal resting values. The iterative procedure is stopped when the numerical variations are *sufficiently* negligible leading to the “equilibrium” steady state conditions. This condition is adopted as a starting point for studying the cellular responses elicitation by applying a square window current density stimuli of different amplitude and duration to boundary 1 (Figure 2). In Table 4 the degrees of freedom and the number of elements for the two different models are reported together with the simulation duration, necessary to achieve equilibrium (the stable stationary condition for the neuron at rest).

Table 4. Figures of merit concerning the two FEM models and corresponding simulation time

PARAMETER/MODEL	**A**	**B**
Degrees of freedom	7086	685
Number of elements	2378	300
Simulation duration for the equilibrium state	13.00 s	2.63 s

By looking at the Table 4, the difference is evident of almost one order of magnitude for the considered figures of merit, concerning the more detailed FEM Model A with respect to the coarse one (Model B), leading to a five times larger simulation time for the reproduction of the axon behaviour for the adoption of the first model. In Table 5, instead, the case of 20ms of membrane behaviour simulation is reported when it undergoes a stimulus-induced response. The considered current stimulus is given by 4 rectangular shape obtained by combining a short-d (2ms) or long-D (19ms) duration and low-m (0.03A/m^2) or high-M (0.3 A/m^2) magnitude.

Table 5. Simulation times (in [s]) for the computation of 20ms of membrane behaviour

	d/m	**d/M**	**D/m**	**D/M**
Model A	83.64	185.59	119.31	183.71
Model B	19.79	48.96	26.89	42.70

In this case an appropriate current density (J_{in}, the square window shown in the Inset of Figure 4a) is applied at $r=r_1=1$nm, very close to the symmetry axis, in order to trigger the excitable membrane (if current density stimulus where injected exactly at r=0μm, current density would have been undefined). A great advantage is offered by Model B in the dynamic case too, as far as stimulation length is concerned. Indeed, for all the considered stimuli, in order to obtain the time shape of the TMV in a time window of 20ms, a simulation time is found of also more then 3’ for the Model A, with respect to almost 40 s for the Model B, as reported in Table V. In Figure 4, the membrane responses obtained at r=0.6μm z=0.2μm for the Model A (blue curves) and Model B (red curves) for each considered case are depicted.

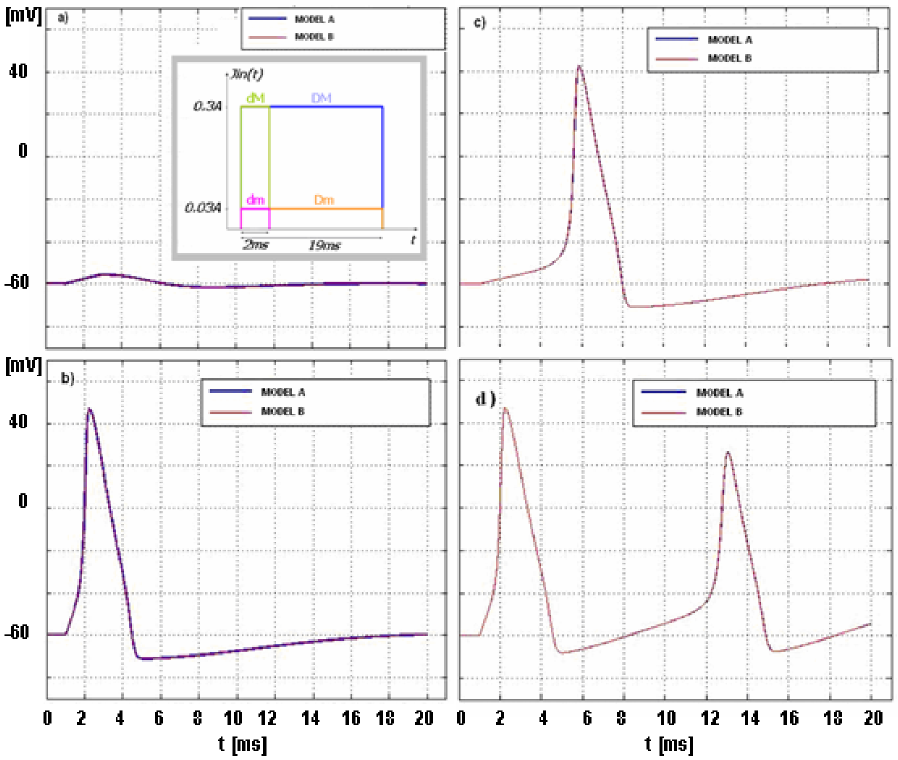


Fig. 4. Membrane response in terms of $V_m(0.6\mu m, 0.2\mu m, t)$: a) *dm*, b) *dM, c) Dm*, d) *DM*. Inset: Input stimulus parameters

In order to compare the two models, the relative difference between the maximum value reached by the AP, obtained by means of Model A with respect to that of Model B is used (the value calculated with Model A is considered as reference value due to the fact that it comes from the more realistic model). The behaviours obtained with both the models are totally in accordance with theoretical expectations [20]. As reported in Figure 4a, in the first case (*dm*) the stimulus is not sufficient to elicit any AP (sub-threshold behaviour, whose parameters, rise time and amplitude, are those expected) showing a passive electrotonic nature of the membrane, being it approachable (at least in first approximation) as an R-C circuit. The relative error between the two models is computed as 0.04%. In the second and in the third cases (see Figures 4b and 4c), an AP is observed (relative error 0.01% and 0.05% respectively), while in the fourth one, since both strength and duration of the stimulus pulse are high (see [7],[20]), two APs are excited, the second of which is lower than the other, because refractory period is not respected. In this last case, depicted in Figure 4d, the relative error is the same as in the case a), i.e. 0.01%, as it is expected due to the fact that the input is the same at the time instant in which the maximum TMV value is reached. It is interesting to observe how membrane responses, in the four corresponding cases almost coincide in the two modelling approaches, with a relative variation not higher then 0.05%. Therefore the coarse Model B seems to be a very good representation of the reality if compared with the fine Model A and the best choice in terms of computational effort. Actually the Model A furnishes the voltage profile along the *r* direction all over the structure considered for the 3D axon sketch (Figure 1), whereas in the Model B no information is available for the axolemma membrane. Therefore, in presence of a lower computational effort, the adoption of the Model B can be conveniently considered in order to evaluate phenomena external to the membrane domain, otherwise the "more realistic" Model A must be adopted.

2 AP Temperature Dependence and Feature of a FEM Approach

The simulations described in the previous section are carried out supposing an operation temperature of 6.3°C, such as the one used by HH in their measurement phase [8] and simulated in [20], with respect to the same current input stresses. Moreover, the channels conductances exhibit a time behaviour that is described by an opportune time constant and that is governed by the activation/inactivation of the corresponding ionic species. The temperature dependency of the gating process, according to [15], can be described by:

$$\frac{dx}{dt} = [\alpha_x(1-x) - \beta_x x] \cdot 3^{\frac{T-6.3}{10}} \tag{12}$$

with $x \in \{m,n,h\}$. This yields to:

$$\frac{dx}{dt} = [\alpha'_x(1-x) - \beta'_x x] \tag{13}$$

leading the original rate coefficients reported in Table 2 to change into α'_x and β'_x thanks to the introduction of the temperature-dependent factor $3^{(0.1T-0.63)}$. The time constant τ_x of the "channel-gating" processes results scaled of the same factor leading to a faster dynamic:

$$\tau'_x = \frac{1}{\alpha'_x + \beta'_x} = \frac{\tau_x}{3^{\frac{T-6.3}{10}}} \tag{14}$$

Adding also temperature dependence to the model, i.e. using the new transfer rate coefficients in the FEM Model A and B, it is easy to obtain the TMV for temperatures within the theoretical validity range of the HH model ([11],[21]). Nevertheless, as far as the interest is on the capability of the FEM approach to reproduce the temperature dependency not necessary in the membrane domain, the behaviour of the TMV with respect to that for only the coarse model (Model B) is performed. In particular, by considering the same temperature experiment performed and theoretically predicted by HH in [17], the obtained time shape of the TMV is reported in Figure 5, showing a good reproduction of the results.

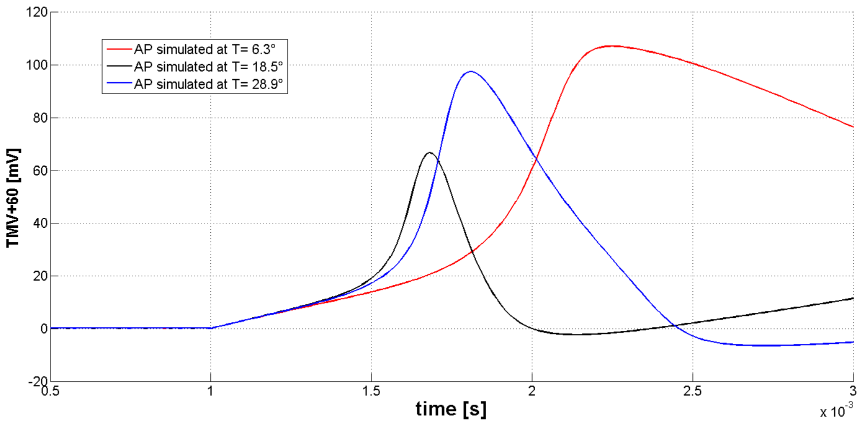


Fig. 5. Simulated TMV behaviour as in [17], obtained with the FEM-Model A

In order to assure the reliability of the FEM solution in terms of the quality of the numerical approximation of the "actual behaviour", i.e. the experimental one, in Table 6 a comparison is reported on the TMV maximum value estimation between a base meshing of the geometry and a more refined one.

Table 6. Comparison of the mesh-cost vs accuracy of the predicted data

PARAMETER/MODEL	Base Mesh	Refined Mesh
Degrees of freedom	7086	27280
Number of elements	2378	9512
Simulation duration	13.00 s	133s
Relative error @T=6.3°C	-1.4312%	-1.4313%
Relative error @ T=18.5°C	-1.6730%	-1.6735%
Relative error @ T=28.9°C	-10.4043%	-10.4045%

In particular, by increasing the approximation of the geometry with a mesh-refinement, an increase on the computational cost in terms of simulation times of a factor ten is found (i.e. a simulation time from 13s to 133s), due to an increase of approximately 3.5 times for the figure of merit of the FEM model (i.e. degrees of freedom and number of elements). Furthermore the improvement on the estimation of the experimental data derived from a more discretized model is negligible, i.e. the difference is only on the fourth digit of the relative error. This low difference is obtained also in correspondence of the higher temperature, where it is known that the temperature dependency of the HH model itself starts to be less realistic [17]. As a consequence it is possible to assume that also a "soft" discretization of each region is necessary and that a "light" simulation is enough in order to perform a satisfactory study of the phenomenon.

3 Best Numerical Model with Respect to the Objective of the Analysis

In order to study two different problems, we need to select the best numerical model, ensuring the desired analysis without introducing a not necessary hard computational effort. In particular, if we are interested, for example, in the reproduction of the spiking effect due to an increase in temperature or to a study finalized to quantitative interpreting the number of APs or, in the end, to determining the maximum TMV value assumed along the longitudinal extension of the axon, the Model B can be adopted. Whereas, if we are interested on the visualization of the behaviour inside the membrane during AP propagation, and its effect around the external medium, the Model A with membrane can be more conveniently adopted. In the two following subsections the results are reported and some comments are indicated.

3.1 Temperature Dependence of the Firing Effect

A parametric study of the TMV with respect to the temperature is made by performing numerical simulation on the Model B, which leads to the time behaviours depicted in Figure 6.

In particular the range [9-21]°C is considered with a step of 3°C for each case. This picture shows how, as theoretically expected [17], the spiking of the neuron is affected by a change in the temperature. Moreover, it can be observed, in Figure 6 and, particularly, in Fig 6 a and b) (zooming on the first AP) that the maximum value

a)

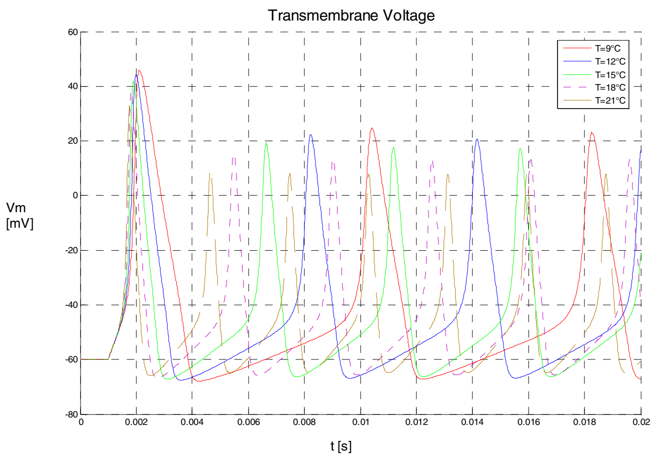
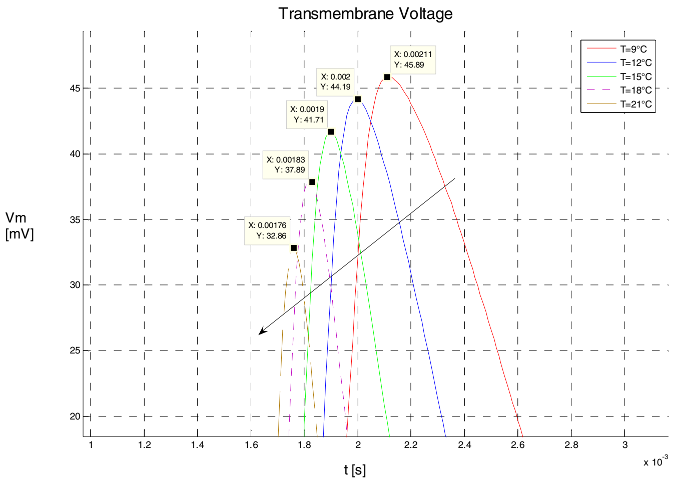


Fig. 6. Temperature dependency of the AP behaviour: a) number of APs in the same time window; b) first AP peak; c) TMV behaviour at a particular temperature (T=18.3°C)

b)



c)

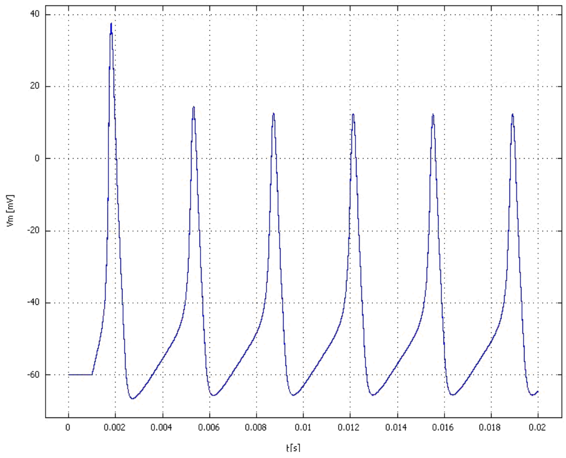


Fig. 6. (*continued*)

assumed by the *Vm(t)* gets lower and lower increasing the value of the ambient temperature, while the duration of the spikes gets minor in contrast with the number of them (three spikes for the case at T=9°C and seven for T=21°C).

In particular, as a particular exemplification case in Fig 5c) we have chosen to show when the temperature assumes a particular value of 18.5°C. In this case the membrane response results in a sequence of six APs, shorter than the two observed at lower temperature reported in Figure 4d). In particular we also check the temperature effects on channels gating process by observing that in turn, channels time constants affect membrane dynamics, which can be represented by the time dependences of the channels conductances ([22]-[23]), reported in Figure 7.

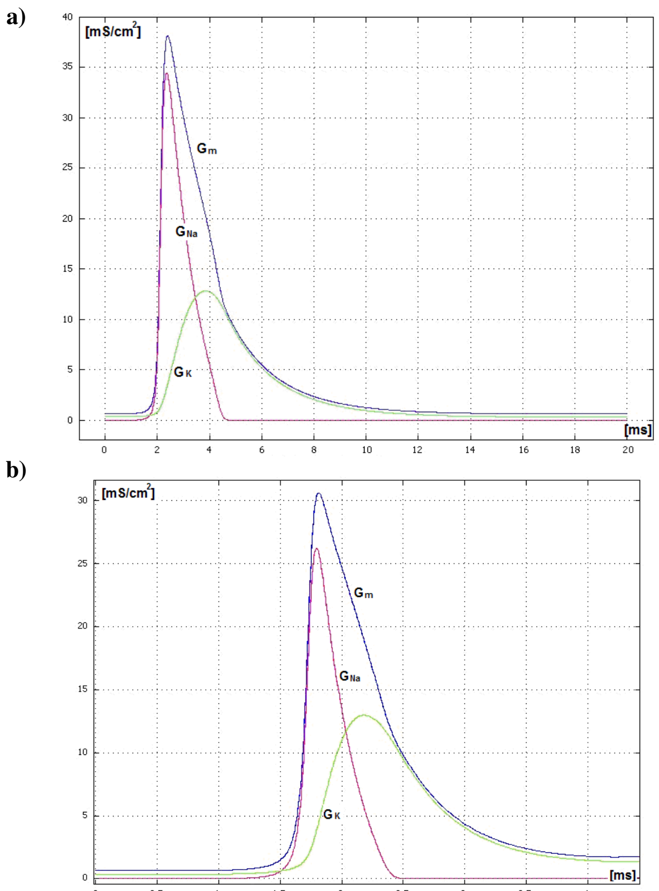


Fig. 7. Temporal dynamic shapes for G_{Na}, G_K, G_l at different temperatures: a) T= 6.3°C; b) T=18.3°C

The maximum of the conductance for each channel is reached faster at higher temperature - case reported in Figure 7b) with T=18.3°C -whereas the duration of the opening of the channel, i.e. a significant value for each conductance, is longer at lower temperature - case depicted in Figure 6a- with T=6.3°C) as confirmed by equation (14).

3.2 The Propagation of the AP along the Membrane Domain

In order to obtain the visualization of the AP propagation inside the membrane, and its effect around the external space, the Model A with membrane is here conveniently adopted. Specifically, in accord to HH experimental setup, once the resting state conditions have been achieved all over the structures, a potential difference, beyond the natural excitement threshold [24], can be fixed across membrane at any transversal section (in this case at z = 0) of the model to elicit a local action potential.

This propagates along the considered axon segment, thanks to the well-known physiological mechanisms of non-myelinated fibres, whose reproduction was the objective of this simulation phase. In particular, this is achieved by fixing a 15mV voltage difference across axon membrane at z=0, thus obtaining the propagation effect shown in Figure 8.

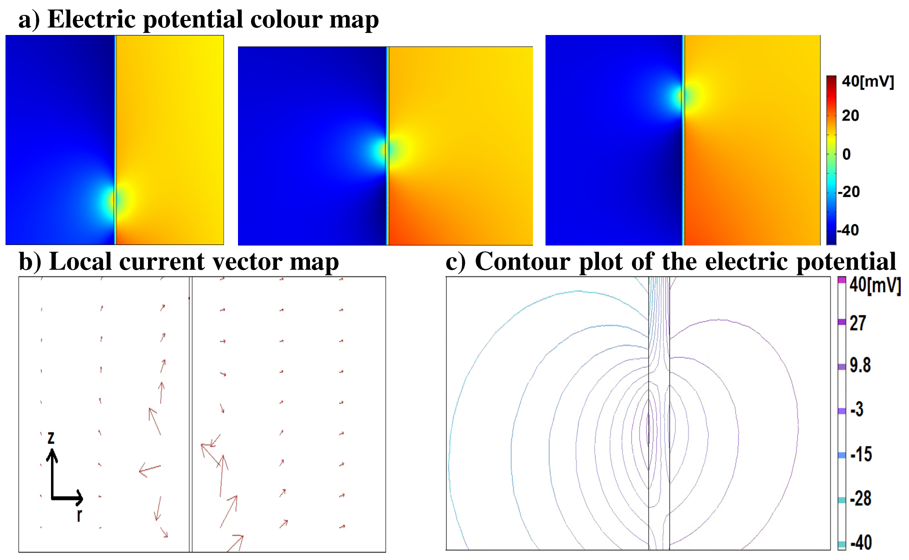


Fig. 8. a) Propagation phenomenon: the moving active zone (z∈ [0.02,0.3]µm and r [0.25,0.75] µm). Potential map at three different times of pulse conduction. **b)** Simulation results for local currents in an activated zone. **c)** Zoom in an active zone: electric potential lines inside and outside membrane (z∈ [0.09, 0.13]µm and r [0.45,0.54] µm).

The explanation of these results is the presence in a certain instant of an AP in an area (the *active zone*, emulated constraining the value of TMV at z=0). This implies that the inner side of the membrane is electrically "more positive" with respect to the outer one. The charge distribution non-homogeneity, thus created, induces longitudinal potential gradients; these in turn generate electric currents (known as *local currents*) in both intra and extra-cellular media, whose lines merge into the active zone (Figure 8b and 8c). All this process results, as expected theoretically, in the activation of the other near areas interested by these charge fluxes. Simulation results for model A are reported to show equipotential lines distribution within an activated section of membrane domain (Figure 8c).

Conclusions and Future Work

It is shown that the two FEM models, here described, allow to simulate the electrophysiological behaviour of a portion of nervous cell axon, to carry out the simulation of the static, underthreshold and active dynamic behaviour, to reproduce action potentials. The maximum deviation in the prediction of the two models in correspondence of the TMV peak-value has proved to be almost 0.05% with respect to an increase in the computational cost of a factor five. It is also shown that the quality of the FEM discretization is not relevant with respect to the capability of the solution method in order to reproduce the expected behaviour. In particular the capability of the FEM solution to reproduce the temperature dependency of the TMV is investigated leading to assume that also a “soft” discretization of each region is sufficient and that a “light” simulation is enough in order to perform a satisfactory study of the phenomenon. Both the proposed models grant the calculation of the electric potential distribution in the space. The Model A, with the direct representation of the axolemma, is unavoidable if the behaviour of the electromagnetic characteristic inside the membrane is matter of interest, whereas the second coarser one (Model B) is computationally more efficient in all the other cases. Therefore, the AP temperature dependences and the propagation effects are reproduced by using the corresponding “best” numerical models, i.e. the coarse one without membrane for the temperature, the more detailed with membrane for the propagation, leading to a trade off between the computational effort and the objective of the analysis. The models are a very useful starting point for a wide range of future works for different application and extension in order to understand, for example, the true behaviour of a complete neuron. It is now possible, without dealing with enormous form factors, to simulate a whole non-myelinated fibre, to introduce more detailed geometry in the structure such as soma and dendrites, implementing their behaviour simply considering locally differentiated channel densities and translating them into opportune conductances per unit area. Also for this topics opportune considerations on the computational cost must be taken into account: when a lower computation effort is required, the adoption of the Model B can be conveniently considered in order to evaluate phenomena external to the membrane domain, otherwise the “more realistic” Model A should be adopted.

Acknowledgment. The results here presented have been partially shown at the European COMSOL Conference 2009, oct 14-15 2009, Milan, Italy [25].

References

[1] Finn, W., LoPresti, P.: The Handbook of Neuroprosthetic Methods. CRC Press (2003)

[2] Talelea, S., Gaynor, P.: Non-linear time domain model of electropermeabilization: Response of a single cell to an arbitrary applied electric field. Journal of Electrostatics 65, 775–784 (2007)

[3] Koch, C., Segev, I.: Methods in Neuronal Modeling: From Synapses to Networks. MIT Press, Cambridge (1989)

[4] Ying, Z., et al.: Micro-stimulator Design for Visual Prosthesis based on Optic Nerve Stimulation. In: International Symposium on Biophotonics, Nanophotonics and Metamaterials, pp. 139–142 (2006)

[5] Berger, et al.: Brain-Implantable Biomimetic Electronics as Neural Prosthetics. In: Proceedings of the 1st International IEEE EMBS Conference on Neural Engineering, Capri Island, Italy, March 20-22 (2003)

[6] Daniel, J., et al.: Chronic Intraneural Electrical Stimulation For Prosthetic Sensory Feedback. In: Proceedings of the 1st International IEEE EMBS Conference on Neural Engineering, Capri Island, Italy, March 20-22 (2003)

[7] Moulin, C., et al.: A New 3-D Finite-Element Model Based on Thin-Film Approximation for Microelectrode Array Recording of Extracellular Action Potential. IEEE Transactions on Biomedical Engineering 55(2), 683–692 (2008)

[8] Hodgkin, A.L., Huxley, A.F.: A quantitative description of membrane current and its application to conduction and excitation in nerve. J. Physiol. 117, 500–544 (1952)

[9] McIntyre, C.C., Grill, W.M.: Microstimulation of spinal motoneurons: a model study. In: Proceedings of the 19th International Conference - IEEE/EMBS, Chicago, IL, USA, October 30-November 2 (1997)

[10] Woo, J., Miller, C.A., Abbas, P.J.: Biophysical Model of an Auditory Nerve Fiber With a Novel Adaptation Component. IEEE Transactions on Biomedical Engineering 56(9), 2177–2180 (2009)

[11] Greenberg, R.J., Velte, T.J., Humayun, M.S., Scarlatis, G.N., de Juan Jr., E.: A Computational Model of Electrical Stimulation of the Retinal Ganglion Cell. IEEE Transactions on Biomedical Engineering 46(5), 505–514 (1999)

[12] Tupsie, S., Isaramongkolrak, A., Paolaor, P.: Analysis of Electromagnetic Field Effects Using FEM for Transmission Lines Transposition. World Academy of Science, Engineering and Technology 53, 870–874 (2009)

[13] Holt, G.R., Koch, C.: Electrical Interactions via the Extracellular Potential Near Cell Bodies. Journal of Computational Neuroscience 6(2), 169–184 (1999)

[14] Stickler, Y., Martinek, J., Rattay, F.: Modeling Needle Stimulation of Denervated Muscle Fibers: Voltage–Distance Relations and Fiber Polarization Effects. IEEE Transactions on Biomedical Engineering 56(10), 2396–2403 (2009)

[15] Hodgkin, A.L., Katz, B.: The effect of temperature on the electrical activity of the giant axon of the squid. J. Physiol. 109, 240–249 (1949)

[16] Keynes, R.D.: The ionic movements during nervous activity. J. Physiol. 114, 119–150 (1951)

[17] Huxley, A.F.: Ion movements during nerve activity. Annals of the New York Academy of Sciences 81, 221–246 (1959)

[18] COMSOL Multiphysics 3.2 Reference Manual, Thin Film Resistance application example

[19] Izhikevich, E.M.: Which Model to Use for Cortical Spiking Neurons? IEEE Transaction on Neuronal Networks 15(5), 1063–1070 (2004)

[20] Moulin, C.: Contribution à l'étude et à la réalisation d'un système électronique de mesure et excitation de tissu nerveux à matrices de microélectrodes. Thèse Institut National des Sciences Appliquées de Lyon, p. 163 (2006)

[21] Rattay, F.: Electrical Nerve Stimulation: Theory, Experiments and Applications. Springer (August 2001) ISBN: 321182247X

[22] Malmivuo, Plonsey, R.: Bioelectromagnetism. Principles and Applications of Bioelectric and Biomagnetic Fields. Oxford University Press (1995)

[23] Taglietti, Casella, C.: Elementi di fisiologia e biofisica della cellula. La goliardica Pavese Editore, Italy (1997)

[24] Rattay, F.: Analysis of the electrical excitation of CNS neurons. IEEE Transaction on Biomedical Engineering 45(6), 766–772 (1998)

[25] Elia, S., Lamberti, P., Tucci, V.: A Finite Element Model for The Axon of Nervous Cells. In: COMSOL Europe Conference 2009, October 14-16, pp. 1–7 (2009) ISBN/ISSN: 978-0-9825697-0-2

Chapter 6
A Study of a Single Multiplicative Neuron (SMN) Model for Software Reliability Prediction

S. Chatterjee*, J.B. Singh, S. Nigam, and L.N. Upadhyaya

Dept. of Applied Mathematics, ISM, Dhanbad-826004, India
chatterjee_subhashis@rediffmail.com

Abstract. This chapter presents a single multiplicative neuron model for predicting software failure has been proposed. Standard back propagation and real coded genetic algorithm with mean squarer error as a fitness function are used for optimizing the parameters. The performance of the proposed model is validated using some real software failure data. A comparative study between some existing software reliability models and the proposed model is presented using different comparison criteria.

Keywords: Single Multiplicative Neuron (SMN), Back Propagation, Genetic Algorithm, Software Reliability Growth Model (SRGM), Fault, Time Between Failures (TBFs).

1 Introduction

Computers play a very important role in modern civilization as these are being used in many areas like bank, defense, medical science, and other safety critical systems. The work performed by the computer is solely carried out by software. A software failure can lead to economic loss and some times may even cause loss of human lives. Therefore, the demand of highly reliable software is increasing day by day. Hence, study of software reliability has become a very crucial issue. Software reliability is defined as the probability of failure free operation of a software in a specified environment for a specified period of time. Software reliability technique is a tool for reducing or eliminating failures of software system. Since 1970's many software reliability growth models (SRGMs) have been proposed to estimate reliability, cost and release time etc. of software system. Some important SRGMs have been considered in [1,2,3]. Most of these SRGMs use system test data to predict the number of defects remaining in the software. In general the utility of SRGM is related to its predictive ability, i.e., the number of remaining faults predicted by the model should be close to the number of faults present in the software. Most of the existing SRGMs in the literature follow non homogeneous poisson process (NHPP) with different mean value functions and they are based on different assumptions. Some of them are immediate fault correction, perfect debugging, uncorrelated failures, etc.

* Corresponding author.

These are not true in real testing environment. Also, due to various assumptions these models are capable of analysing some particular software failure data only. As a result many SRGMs have been developed with different predictive performance. Considering these facts, Karunanithi et al. [4, 5] first proposed neural network based SRGMs as an alternative to parametric SRGMs. In recent years artificial neural network (ANN) approach has proven to be a universal approximator for any non linear continuous function with an arbitrary accuracy. Consequently it has become an alternative method in software reliability modeling, evaluation and prediction [6, 7, 8]. In [4, 5] Karunanithi et al. have taken execution time as input and predicted the cumulative number of faults. Later many authors Cai et al.[6], Tian and Noore [7, 8], Khoshgoftar and Szabo [9], Park et al.[10], Adnan et al.[11], Aljahadali et al.[12], Ho et al.[13], Sherer [14], Hu et al.[15], Su et al.[16], Zheng [17] have also applied neural network approach in software reliability modeling and found better results over parametric SRGMs. Some existing ANN based SRGMs have used single-input and single output neural network architecture [4, 5, 10] whereas some authors have used multiple delayed input and single output architecture [7, 8, 11, 12].

Generally in ANN models, inputs are combined using the summing operations. Multiplication plays an important role in between neural modeling of biological behavior and computing and learning with artificial neural networks [18]. Michel Shmitt [19] used the concept of the multiplicative neural network which contains units that multiply their inputs instead of summing them and the complexity of computing and learning for multiplicative neural networks has been discussed. In [20] Yadav et al. has proposed a single multiplicative neuron (SMN) model inspired from single neuron computation in neuroscience [21, 22]. This model is based on polynomial architecture. They have used it for time series prediction and function approximation. Later Zhao et al. [23] proposed a particle swarm optimization based single multiplicative neuron model and predict some well known time series.

In this chapter a multiple delayed input SMN model has been proposed for prediction of software fault using historical software failure data. First time the application of SMN in the area of software reliability has been studied. The proposed SMN is a neural network comprising of simple structure and less parameters as compared to traditional ANN based SRGMs. The proposed SMN network does not contain any hidden layer and one does not need to determine the monomial structure in the sense of number of hidden layer prior to training of neuron model. Mostly traditional SRGMs based on neural network have relationship between failure sequence number and failure time data. Here the interrelationship among the failure sequence number is considered, i.e., inputs and outputs both are the cumulative failure number. A standard back propagation (BP) and a real coded genetic algorithms (RCGA) have been used to optimize the parameter, i.e., weight and biases of network and the results have been shown separately.

Rest of the chapter is organized as follows: In section 2 SMN has been described. In section 3 the standard back propagation and genetic algorithm have been provided for optimization of network parameters. Section 4 represents results and discussion. Section 5 describes comparison with some existing parametric SRGMs. Finally section 6 contains the concluding remark.

2 The Single Multiplicative Neuron Model

The structure of a generalized SMN model with learning algorithm is given in Figure 1.

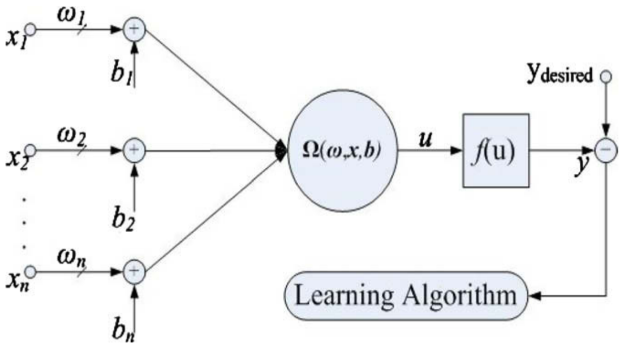


Fig. 1. The structure of generalized SMN

where $(x_1, x_2, \ldots, x_n)$, $(w_1, w_2, \ldots, w_n)$, and $(b_1, b_2, \ldots, b_n)$ are the input pattern, weight and biases respectively. The operator Ω in the figure is a multiplicative operation and given as

$$\Omega = \prod_{i=1}^{n} (w_i x_i + b_i) \tag{1}$$

The output function is the *logsig* function defined as Eq. (2)

$$y = \frac{1}{1 + e^{-u}} \tag{2}$$

here u is equal to Ω

3 Learning Rule for the Single Multiplicative Neuron Model

This section describes an application of Back Propagation and a real coded Genetic Algorithm in optimizing weights and biases of SMN model during its learning. The learning algorithm has been used to minimize the error between the original and predicted values.

3.1 Back Propagation (BP) Learning Algorithm

The standard BP algorithm based on steepest descent gradient approach has been adopted to train the proposed SMN model and minimize the error function E defined as

$$E = \frac{1}{2n}\sum_{p=1}^{n}(y_p - y_p^d)^2 \tag{3}$$

where y_p is the output of the network and y_p^d is the desired output for the p^{th} input pattern. Using the steepest descent gradient approach and the chain rules for the partial derivative, the update rules for the weight and biases of the model can be obtained from the following equations:

$$w_i^{new} = w_i^{old} + \Delta w_i \tag{4}$$

$$b_i^{new} = b_i^{old} + \Delta b_i \tag{5}$$

where

$$\Delta w_i = -\eta\frac{dE}{dw_i} = -\eta\frac{1}{n}\sum_{p=1}^{n}\left((y_p - y_p^d)y_p(1-y_p)\frac{u}{w_i x_i + b_i}x_i\right), \tag{6}$$

$$\Delta b_i = -\eta\frac{dE}{db_i} = -\eta\frac{1}{n}\sum_{p=1}^{n}\left((y_p - y_p^d)y_p(1-y_p)\frac{u}{w_i x_i + b_i}\right), \tag{7}$$

and η is the learning rate parameter that controls the convergent speed. The above process is iterative until predefined error goal is reached.

3.2 The Genetic Algorithm

Genetic algorithm (GA) is a directed random search technique that is widely applied in optimization problem [24, 25, 26]. In general, genetic algorithm is used in the area of neural network for designing the structure and optimizing the neural network parameters. In ANN based SRGMs genetic algorithm has been applied successfully for optimizing the network architecture [7, 8, 27] and the parameters, i.e., the weight and biases [28] to predict the cumulative faults as well as TBFs of software. GA is a optimization technique in which the solution space is searched by generating a population of candidate individuals. Each individual is called a chromosome which represents a configuration of a solution. A general procedure of standard GA is as follows: (i) create an initial population of chromosomes (initial set of solution) at random, (ii) the chromosomes are computed by a defined fitness function, (iii) select subpopulation for next generation reproduction, (iv) generate a new population by applying the genetic operators of crossover and mutation to this selected subpopulation. Normally the new population has a greater average fitness than preceding population and the above process is repeated until a predefined criterion is reached.

The most common representation of traditional GA is binary, i.e., the chromosome consist of a set of genes which are characters belonging to an alphabet {0,1}. RCGA is an alternative of the representation issue, which is particularly natural when tackling optimization problems of parameters with variables in continuous or discontinuous domains. In the RCGA, a chromosome is a vector of floating point numbers corresponding to design variables. The RCGA is more efficient because the real number representation is conceptually close to real design space and length of the chromosome becomes shorter. In [26] it has been shown that for real valued numerical optimization problems, real number representations outperforms binary representation because they are more consistent, precise and lead to faster execution.

The combination of genetic algorithm and SMN network for training consists of three phases: (i) In the first phase the representation of connection weights and biases is decided, i.e., whether to use a binary string form or directly use a real number form. Since in this work RCGA has been used, the weights and biases will be real number in chromosome. (ii) In the second phase the evaluation on the fitness of these connection weights are made. The mean square error criterion has been used as a fitness function defined in equation (9). (iii) In the third phase the evolutionary process such as selection, crossover and mutation operations by genetic algorithm according to its fitness has been applied.

A chromosome consists of all network weight and biases. One gene of chromosome represents a single weight value. Suppose there are two input neurons in the network so the chromosome consists of four genes, i.e., two weights and two biases, i.e., $c = (w_1, w_2, b_1, b_2)$

The stepwise procedure for the training of SMN network using RCGA is as follows:

Step 1. A population of chromosome is created.

Step 2. Evaluate the fitness function (given in equation 9) for each chromosome in the population.

Step 3. Create an intermediate population by extracting members from the current population using a selection operator.

Step 4. Create a new population by applying the genetic operator of crossover and mutation to this intermediate population.

Step 5. Repeat the above steps until a given condition is attained.

Fitness Function: Fitness function 1/(1+E) has been chosen, where E is the mean square error

(MSE) defined as

$$E = \frac{1}{n}\sum_{p=1}^{n} (y_p - y_p^d)^2 \tag{8}$$

where y_p is the output of the network and y_p^d is the desired output for the p^{th} input pattern.

$$fitness = \frac{1}{1 + E} \tag{9}$$

Selection Mechanism: The roulette wheel selection is used to create the intermediate population. The selection can be done by assigning the probability $p_s(c_i)$ to the chromosome c_i defined as

$$p_s(c_i) = \frac{fitness(c_i)}{\sum_{j=1}^{p} fitness(c_j)} \tag{10}$$

4 Results and Discussion

The performance of proposed SMN model has been tested using two real software failure data. The proposed ANN based SRGM has been used to predict cumulative number of faults. Here, the cumulative number of faults at current stage of testing has been predicted using cumulative number of faults of previous stage of testing. Hence, the output of the proposed model becomes the function of pervious cumulative faults, i. e.,

$x_i, x_{i+1}, \ldots, x_{i+k-1}$ ($i = 1, 2, \ldots$) are used to predict the value x_{i+k}, where k is the number of delayed input node and x_i is the cumulative fault at i^{th} time point.

$$\text{i.e. } x_{i+k} = f(x_i, x_{i+1}, \ldots, x_{i+k-1}) \qquad i = 1, 2, \ldots \tag{11}$$

Here k input delay means k number of past input used to deliver the current output, e.g., suppose the number of the delays in input node is 2, i.e., the value of $k = 2$ then the values of x_1, x_2 are used to predict x_3 (for $i = 1$). The number of delayed input nodes have been determined by trial and error. In this paper initially two delayed inputs has been considered and then increased one by one up to 5. The number of delayed input has been increased from two to five for the selection of training data set and delay. The optimal values of training data set and a delay has been selected based on the performance of the network. The performance has been measured on the basis of fitness function (MSE) value for specific dataset. The data set has been split in two sets: one for training and one for testing. All the inputs are normalized to values in the range of 0.1 to 0.9. The MATLAB 7.6 Neural Network Tool Box and Genetic Algorithm Tool Box have been used for computation. The learning rate η of BP Algorithm set as 0.6 and the maximum iterating epoch as 5000. Population size of GA is set as 20, the maximum generation is set as 500 and other parameters are set as default values.

Result Using Data Set 1: The effectiveness of proposed model has been tested on a Real-Time Control Systems data given in [1]. The software for monitor and real-time control systems consists of about 200 modules and each module has, an average, 1000 lines of a high-level language code like FORTRAN. This data set records the software failure detected during the 111-day testing period. Initially the entire data set is divided in various segments like 50, 70, 90 and 100 percent respectively, then the MSE values have been calculated for all the segments of the available data with different values of input nodes. The computed values of MSE have been given in Table 1. Table 1 show that the 70% of the data with three number of delayed input nodes has a minimum MSE value. Therefore, the first 70% has been used to train the network and estimate the network parameter. The remaining 30% has been used for testing the performance of the model. The corresponding graph has been shown in Fig.2. Fig. 2 shows that, the predictive accuracy of the proposed model is good. It also shows that, prediction is good for both the training algorithm: BP as well as GA.

Table 1. Different set of training data with corresponding MSE values corresponding to various delays for Data Set 1

DATA2	**2 input delay**	**3 input delay**	**4 input delay**	**5 input delay**
50%	2637.10	1528.10	1724.80	1757.20
70%	226.5400	169.9646	188.0801	190.9004
90%	266.6707	180.5346	200.4515	204.4155
100%	250.4367	183.4476	208.6639	208.8315

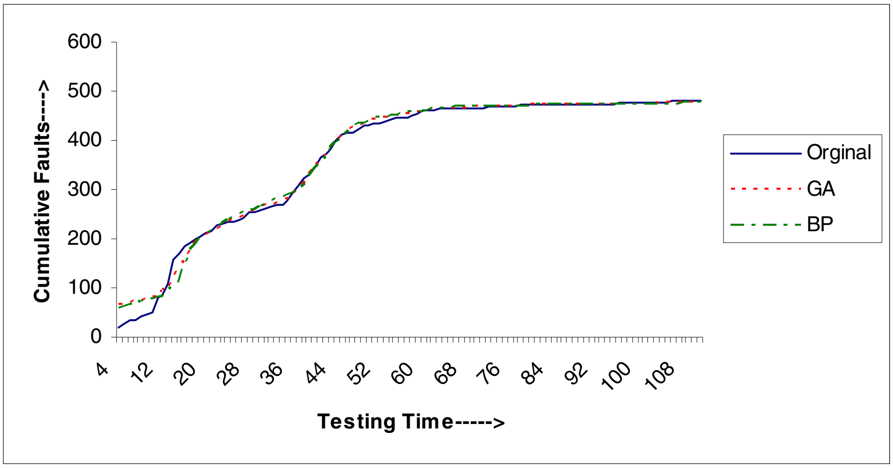


Fig. 2. Prediction of cumulative faults vs. testing time for DATA2

Results Using Date Set 2: A real time control application software failure Data 7 given in the CD ROM attached with the book published by Lyu [3] has also been examined by proposed model. The software consists 870,000 lines of code. The test time reported in days and the cumulative faults captured corresponding to each day. This data set contains 109 observations. This data set records the software failure detected during the 111-days of testing period. Here also the data set is divided in various segments like: 50, 70, 90 and 100 percent. Then the MSE values have been calculated for all the segment with different number of input nodes. The computed values of MSE have been shown in Table 2. Table 2 shows that 70% data with three numbers of delayed inputs has a minimum MSE value. Therefore, the first 70% has been used to train the network and estimate the network parameters. The remaining 30% data have been used for testing. Corresponding graph has been given in Fig 3. Fig 3 illustrates the predictive accuracy of model for both the training algorithm: BP as well as GA.

Table 2. Different set of training data with corresponding MSE values corresponding to various delays for Data Set 2

DATA7	2 input delay	3 input delay	4 input delay	5 input delay
50%	985.5196	572.7014	588.5983	576.4567
70%	229.9461	191.5603	205.6274	199.1753
90%	240.3755	208.4754	224.4585	219.1785
100%	237.8068	205.7068	220.0236	216.4302

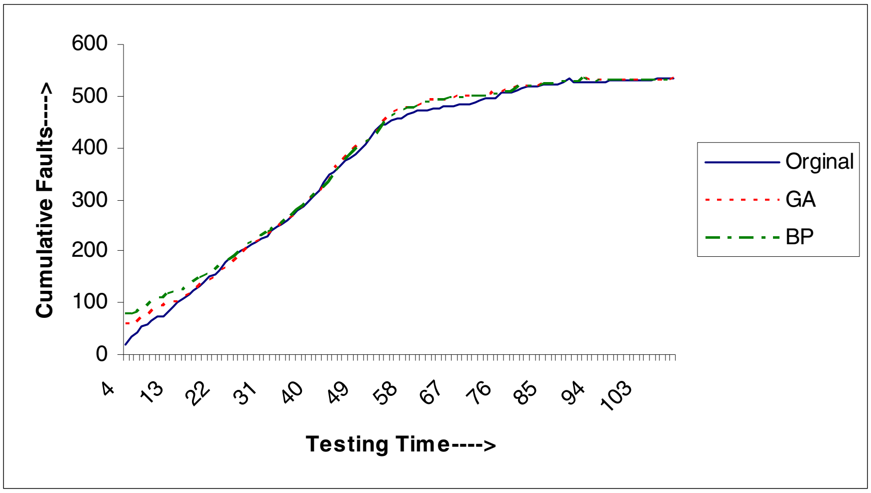


Fig. 3. Prediction of cumulative faults vs. testing time for DATA7

5 Comparison

In this section a comparative study of the proposed SMN model with some important existing parametric SRGMs [29, 30, 31] and feed forward neural network based SRGMs[4, 5] has been carried out. Here feed forward neural network with 3 input delays has been considered. The comparison has been made by evaluating some performance criteria like root mean square error (RMSE), coefficient of multiple determination $\left(R^2\right)$ and u-plot. Above mentioned criterions have been defined as follows:

(i) Root Mean Square Error (RMSE): $$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(y_i - \widehat{y}_i\right)^2}, \qquad (12)$$

where y_i and $\widehat{y}_i$ are the observed and predicted faults respectively and n is the total number of observations. The minimum RMSE represents less fitting error and better performance.

(ii) Coefficient of Multiple Determination (R^2): $R^2 = 1 - \frac{SSE}{SS_{yy}}$, (13)

Where SSE= $\sum_{i=1}^{n} (y_i - \hat{y}_i)^2$. SSE measures the deviation of the observation from their predicted values and $SS_{yy} = \sum_{i=1}^{n} \left(y_i - \overline{y} \right)^2$ ($\overline{y}$ is the mean of the observation). SS_{yy} measures the deviations of the observations from their mean. R^2 measures the percentage of the total variation about the mean accounted for the fitted curve, its range is 0 to 1. Small value indicates that the model does not fit the data well and the larger R^2 value is, the better the model explains the variation in data [2].

(iii) u- plot: The general technique for detecting systematic objective difference between predicted and observed failure behavior is called u-plot [3, 16]. The purpose of u-plot is to determine if the predictions are on the average close to true distribution or not. If the prediction is perfect, then the u-plot looks like the line of unit slope. A common way to test the significant difference between prediction and actual value is Kolmogorove Distance (KD). The KD is the maximum vertical distance between u-plot and the line of unit slope [3, 16].

The mean value functions and calculated parameter values of parametric SRGMs for both datasets has been tabulated in Table 3.

Table 3. Mean value functions and estimated parameters

Model	Mean Value Function	Estimated Parameters
Goel-Okumoto	$m\ (t) = a(1 - e^{-bt})$	$a = 497.282,\ b$=0.0308 (Data Set 1) $a = 557.6004,\ b$=0.0209 (data Set 2)
Delay S Shaped	$m(t) = a(1-(1+bt)e^{-bt})$	$a = 483.039\ b$=0.06866 (Data Set1) $a = 543.2680,\ b$=0.0522 (Data Set 2)
Quasi - renewal Time-delay model based on G-O model	$m(t) = \sum_{k=1}^{n-1}\left(ab - b.m(s_1\sum_{j=0}^{k-2}\alpha^j)\right)\left(s_1\sum_{j=0}^{k-1}\alpha^j - s_1\sum_{j=0}^{k-2}\alpha^j\right) + \left(ab - b.m[s_1\sum_{j=0}^{n-2}\alpha^j]\right)\left(t - s_1\sum_{j=0}^{n-2}\alpha^j\right)$	$a = 490.31,\ b$=0.0187, α=0.93, s_1=45 (Data Set 1) $a = 540.2593,\ b$=0.01780, α=0.94, s_1=47 (Data Set 2)

For the Data Set 1, u-plot has been shown in Figure 4 and the SSE, RMSE, R^2 and KD of u-plot values have been computed for SMN model and compiled in Table 4.

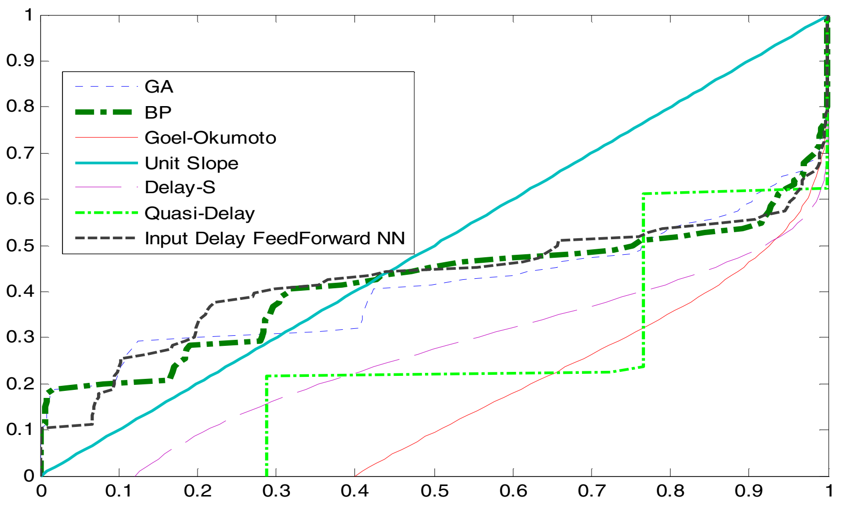


Fig. 4. u-plot of Data Set 1

Table 4. SSE, RMSE, R^2 and KD values for Data Set 1

Models	RMSE	R square	KD Values
Goel-Okumotto	31.6736	0.9484	0.4482
Delay S Shaped	18.6170	0.9822	0.4205
Quasi - renewal Time-delay model based on G-O model	18.7153	0.9820	0.5303
Single Neuron GA	11.5097	0.9932	0.3093
Single Neuron Back P	13.0370	0.9913	0.3674
Input Delay Feed Forward NN	14.9367	0.9885	0.3707

For the Data Set 2, u-plot has been shown in Figure 5 and the SSE, RMSE, R^2 and KD of u-plot values have been computed for SMN model and compiled in Table 5.

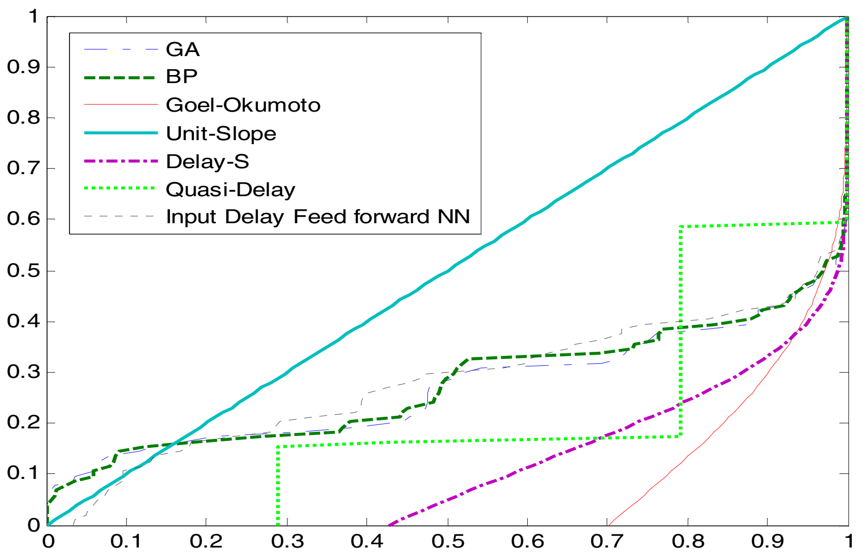


Fig. 5. u-plot of Data Set 2

Table 5. SSE, RMSE, R^2 and KD values for Data Set 2

Models	RMSE	R square	KD Values
Goel-Okumotto	47.8074	0.9167	0.7008
Delay S Shaped	17.3946	0.9890	0.5687
Quasi - renewal Time-delay model based on G-O model	37.6687	0.9483	0.6191
Single Neuron GA	9.3132	0.9968	0.4860
Single Neuron Back P	13.8405	0.9930	0.4888
Input Delay Feed Forward NN	14.4888	0.9923	0.4926

Tables 4 and 5 clearly demonstrate that all the computed values for proposed SMN model are obtained using BP and RCGA are minimum. The computed value for GA is better than BP algorithm also. Hence, it can be concluded that the proposed model perform well compared to model given in [4, 5, 29, 30, 31].

6 Concluding Remark

In this chapter the use of single multiplicative neuron model for prediction of cumulative faults of software has been explored. Back propagation and genetic algorithm have been used to train the network and estimate the model parameters for the prediction of cumulative number of faults during testing. The performance of this model has been studied by using two real data sets. The computed result shows that there is a significant better performance by the proposed SMN model as compared to the existing parametric software reliability models for both training BP algorithm and GA. Genetic algorithm is better than BP. It is interesting to note that the proposed method has advantage in handling any software failure data set since the previous data is required. The significant advantage of the proposed model is that it is assumption free, applicable to any software failure data and gives better prediction. Also, in SMN model as there is no hidden layer, the proposed model is computationally more efficient than other ANN model.

Acknowledgement. Authors acknowledge University Grants Communication (UGC), New Delhi for financial help under the project number F.No.33-115/2007 (SR) and Indian School of Mines, Dhanbad, for providing necessary facilities for this work. The authors are very much thankful to the reviewers for valuable suggestion to improve the paper.

References

[1] Pham, H.: System Software Reliability. Springer, London (2006)

[2] Kapur, P.K., Garg, R.B., Kumar, S.: Contribution to Hardware and Software Reliability. World Scientific, Singapore (1999)

[3] Lyu, M.R.: Handbook of Software Reliability Engineering. IEEE Computer Society Press, McGraw Hill, New York (1996)

[4] Karunanithi, N., Whitley, D., Malaiya, Y.K.: Prediction of software reliability using connectionist models. IEEE Transactions on Software Engineering 18(7), 563–574 (1992), doi:10.1109/32.148475

[5] Karunanithi, N., Whitley, D., Malaiya, Y.K.: Using neural network in reliability prediction. IEEE Software 9(4), 53–59 (1992), doi:10.1109/52.143107

[6] Cai, K.Y., Cai, L., Wang, W.D., Yu, Z.Y., Zhang, D.: On the neural network approach in software reliability modeling. The Journal of Systems and Software 58(1), 47–62 (2001), doi:10.1016/S0164-1212(01)00027-9

[7] Tian, L., Noore, A.: On-line prediction of software reliability using an evolutionary connectionist model. The Journal of Systems and Software 77, 173–180 (2005), doi:10.1016/j.jss.2004.08.023

[8] Tian, L., Noore, A.: Evolutionary neural network modeling for software cumulative failure time prediction. Reliability Engineering and System Safety 87, 45–51 (2005), doi:10.1016/j.ress.2004.03.028

[9] Khoshgoftaar, T.M., Szabo, R.: Using neural network to predict software faults during testing. IEEE Transactions on Reliability 45(3), 456–462 (1996), doi:10.1109/24.537016

[10] Park, J.Y., Lee, S.U., Park, J.: Neural network modeling for software reliability prediction from failure time data. Journal of Electrical Engineering and Information Science 4(4), 533–538 (1999)

[11] Adnan, W.A., Yaakob, M., Anas, R., Tamjis, M.R.: Artificial neural network for software reliability assessment. In: TENCON Proceeding of the Intelligent System and Technologies for the New Millennium, pp. 446–451 (2000), doi:10.1109/TENCON.2000.892307

[12] Aljahdali, S.H., Buragga, K.: Employing four ANNs Paradigms for Software Reliability Prediction: an Analytical Study. International Journal on Artificial Intelligence and Machine Learning (AIML) ICGST 8(2), 1–8 (2008)

[13] Ho, S.L., Xie, M., Goh, T.N.: A study of the connectionist models for software reliability prediction. Computer and Mathematics with Application 46(7), 1037–1045 (2003), doi:10.1016/S0898-1221(03)90117-9

[14] Sherer, S.A.: Software fault prediction. Journal of System and Software 29, 97–105 (1995), doi:10.1016/0164-1212(94)00051

[15] Hu, Q.P., Xie, M., Ng, S.H., Levitin, G.: Robust recurrent neural network modeling for software fault detection and correction prediction. Reliability Engineering and System safety 92, 332–340 (2007), doi:10.1016/j.ress.2006.04.007

[16] Su, Y.S., Huang, C.Y., Chen, Y.S., Chen, J.X.: An artificial Neural-Network-Based Approach to Software Reliability Assessment. In: Proceedings of IEEE Region 10 Conference, Melbourne, Australia (2005), doi:10.1109/TENCON.2005.301242

[17] Zheng, J.: Predicting software reliability with neural network ensembles. Expert System with Applications 36, 2116–2122 (2009), doi:10.1016/j.eswa.2007.12.029

[18] Durbin, R., Rumelhart, D.: Product units: A computationally powerful and biologically plausible extension to back propagation networks. Neural Computation 1, 133–142 (1989), doi:10.1162/neco.1989.1.1.133

[19] Schmitt, M.: On the Complexity of Computing and Learning with Multiplicative Neural Networks. Neural Computation 14, 241–301 (2001), doi:10.1162/08997660252741121

[20] Yadav, R.N., Kalra, P.K., John, J.: Time series prediction with single multiplicative neuron model. Applied Soft Computing 7, 1157–1163 (2007), doi:10.1016/j.asoc.2006.01.003

[21] Koch, C.: Computation and single neuron. Nature 385, 207–210 (1997), doi:10.1038/385207a0

[22] Koch, C.: The role of single neurons in information processing. Nature Neuroscience 3(suppl.), 1171–1177 (2000), doi:10.1038/81444

[23] Zhao, L., Yang, Y.: PSO-based single multiplicative neuron model for time series prediction. Expert Systems with Applications 36, 2805–2812 (2009), doi:10.1016/j.eswa.2008.01.061

[24] Holland, J.H.: Adaptation in Natural and Artificial System. University of Michigan Press, Ann Arbor (1975)

[25] Pham, D.T., Karaboga, D.: Intelligent Optimization Techniques, Genetic Algorithms, Tabu Search, Simulated Annealing and Neural Networks. Springer, New York (2000)

[26] Michalewicz, Z.: Genetic Algorithm+Data Structures=Evolution Programs, 2nd edn. Springer, New York (1994)

[27] Aljahdali, S.H., El-Telbany, M.: Genetic Algorithms for Optimizing Ensemble of Models in Software Reliability Preiction. International Journal on Artificial Intelligence and Machine Learning (AIML) ICGST 8(1), 5–13 (2008)

[28] Benaddy, M., Wakrim, M., Aljahdali, S.: Evolutionary Neural Network Prediction for Cumulative Failure Modeling. In: Proceeding of ACS/IEEE International Conference on Computer System and Application, pp. 179–184 (2009), doi:10.1109/AICCSA.2009.5069322

[29] Goel, A.L., Okumoto, K.: A Time-Dependent Error Detection Rate Model for Software Reliability and Other Performance Measure. IEEE Trans. on Reliability 28(3), 206–211 (1979), doi:10.1109/TR.1979.5220566

[30] Yamada, S., Ohba, M., Osaki, S.: S-shaped Software Reliability Growth Models and their Applications. IEEE Trans. on Reliability 33(4), 289–292 (1984), doi:10.1109/TR.1984.5221826

[31] Hwang, S., Pham, H.: Quasi-Renewal Time-Delay Fault Removal Consideration in Software Reliability Modelling. IEEE Trans. on Systems, Man and Cybernetics-A 39(1), 200–209 (2009), doi:10.1109/TSMCA.2008.2007982

Chapter 7
Numerical Treatment for Painlevé Equation I Using Neural Networks and Stochastic Solvers

Muhammad Asif Zahoor Raja[1,4], Junaid Ali Khan[1,4], Siraj-ul-Islam Ahmad[2], and Ijaz Mansoor Qureshi[3]

[1] Department of Electronic Engineering, International Islamic University Islamabad, Pakistan
{asif.phdee10,Junaid.phdee17}@iiu.edu.pk
[2] Pakistan Institute of Engineering and Applied Sciences, Nilore, 45650 Islamabad Pakistan
[3] Department of Electrical Engineering, Air University, Islamabad Pakistan
[4] Department of Electrical Engineering, COMSATS Institute of Information Technology, Attock, Pakistan
{muhammad.asif,Junaid.Ali}@ciit-attock.edu.pk

Abstract. In this chapter, a new stochastic numerical treatment is presented for solving Painlevé I equation. The mathematical model of the equation is formulated with feed-forward artificial neural networks. Linear combination of the networks defines the unsupervised error for the equation. The error is reduced subject to the availability of appropriate weights of networks. Training of weights is done with genetic algorithm, simulating annealing and pattern search algorithms hybridized with interior point algorithm for rapid local search. The reliability and effectiveness is validated with the help of statistical analysis. Comparison of results is made with standard approximate analytic solvers of the equation. It is found that the proposed results are in a good agreement with their corresponding numerical solutions.

1 Introduction

The Painlevé transcendents are of special interest in the research community due to its nature of singularity. The solution of Painlevé equations arises in both pure [1] and applied mathematics [2], along with theoretical physics [3]. These transcendents have numerous applications in applied sciences and engineering [4]. A number of researchers have investigated the numerical solution of Painlevé I using several methods, which attracts the authors to find the approximate solution of this non-linear initial value problem. We have considered the following form

$$\frac{d^2y(t)}{dt^2} = 6y(t)^2 + t, \tag{1}$$

using the following initial conditions

$$y(0) = 0, \qquad \frac{dy(0)}{dt} = 1. \tag{2}$$

The exact solution of (1) is not known, however the approximate analytic solution has been worked out by various authors. The solution of Painlevé is achieved by the reduction of Korteweg-de Vries (KdV) and cylindrical KdV equation [5]. The state of art analytic solvers like Adomian decomposition method, variation iteration method and Homotopy perturbation method are well known methods to find the solution of these kinds of non-linear problems [6-10]. In above mentioned methods the solution is provided in the form of infinite series. The accuracy of the results of these solvers depends upon number of the terms used to approximate the solution. The existence, uniqueness and convergence of the solution are the criteria for a good numerical method.

In this regard, artificial neural networks are the candidates for approximating the solutions for the same criteria. They are applicable for the problems involving non-linearity along with singularity [11-12]. Recently, neural networks mathematical models are used to solve not only ordinary differential equations but also fractional differential equations [13-16]. This is the motivation to investigate for the solution of Painleve equation with the help of neural networks modeling. The real strength of such model can be seen by the use of modern stochastic solvers for training of their weights.

In this paper, the neural networks supported with computational intelligence algorithms are used to solve Painlevé equation. To the best of author's knowledge, such techniques have yet not applied to solve this equation. The mathematical model of the equation is developed by using the feed-forward artificial neural networks that defines an unsupervised error. The availability of weights to optimize such error is highly stochastic in nature. So, the training of weights is carried out by Genetic algorithm (GA) used as a global optimizer hybridized with interior point algorithm (IPA), which is a tool for rapid local search. The simulating annealing and pattern search techniques hybridized with IPA are also used for learning of weights. The reliability and effectiveness of the schemes are analyzed through statistical analysis. Moreover, the comparison of the results is made with standard analytic solvers.

The organization of the paper is as follows. The neural network mathematical modeling along with the formulation of the fitness function and the learning procedure of the neural networks is introduced in 2nd section. A brief introduction to genetic algorithm, simulating annealing, pattern search and interior point algorithm is revealed in section 3. A detailed application of the designed scheme along with discussion on the results is presented in section 4. The last section concludes the findings along with some directions for future research.

2 Neural Network Mathematical Model

In this section, mathematical model for Painlevé I is developed with feed-forward artificial neural networks (ANN). The neural network architecture uses log-sigmoid as an activation function.

It is well known that feed-forward NN have been extensively used as an universal approximating function, because any continuous function along with its derivative of any order on a compact set can be approximated well by the basic architecture of feed-forward neural networks. In case of this equation, following continuous mapping is employed in neural network methodology for the solution *y(t)*, its first derivative (dy/dt) and second derivative (d^2y/dt^2), respectively [12], [17].

$$\hat{y}(t) = \sum_{i=1}^{m} \alpha_i f(w_i t + \beta_i), \tag{3}$$

$$\frac{d\hat{y}(t)}{dt} = \sum_{i=1}^{m} \alpha_i \frac{d}{dt} f(w_i t + \beta_i), \tag{4}$$

$$\frac{d^2\hat{y}(t)}{dt^2} = \sum_{i=1}^{m} \alpha_i \frac{d^2}{dt^2} f(w_i t + \beta_i), \tag{5}$$

where α_i, w_i and β_i are real-valued bounded adaptive parameters, m is constant giving number of neurons and f being the activation function taken as log sigmoid for hidden layers.

$$f(t) = \frac{1}{1+e^{-t}}. \tag{6}$$

The linear combination of the networks represented by (3), (4) and (5) can arbitrarily model the equation (1). It is named as differential equation neural network (DENN) of the equation and its architecture is shown in Fig. 1.

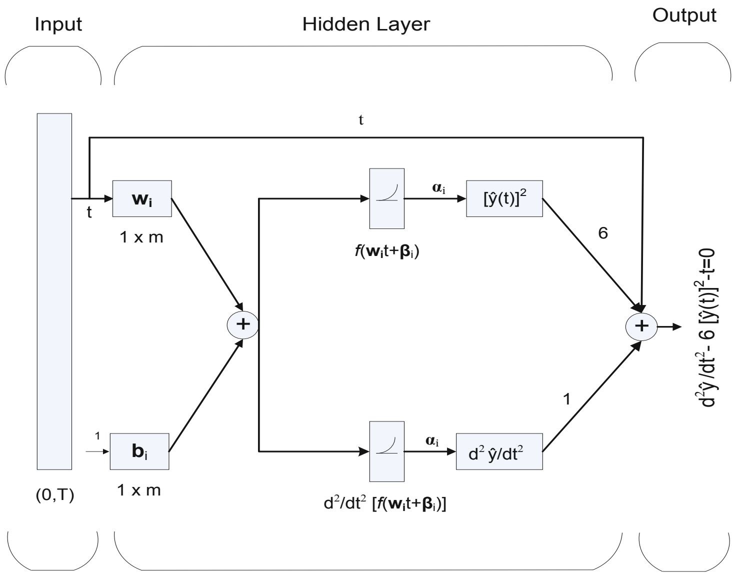


Fig. 1. DENN architecture for Painlevé I

The fitness function e_j is formulated as:

$$e_j = e_1 + e_2\big|_j, \quad j = 1, 2, 3\ldots, \tag{7}$$

where j is the generation number, and e_1 is the error associated with the equation (1) that is represented by (3) to (5) and is given as

$$e_1 = \frac{1}{s}\sum_{i=1}^{s}\left[\frac{d^2}{dt^2}\hat{y}(t_i) - 6\{\hat{y}(t_i)\}^2 + a\hat{y}(t_i)\right]^2 \tag{8}$$

where $T = sh$, S is the total number of equally spaced steps and h defines the size of step, t is taken in the interval *(0,T)*. By increasing the value of S , the accuracy of the modeling is enhanced but at the cost of more computations.

Similarly e_2 is linked with initial conditions and is written as,

$$e_2 = \frac{1}{2}[\{\hat{y}(0)\}^2 + \{\frac{d\hat{y}(0)}{dt} - 1\}^2] \tag{9}$$

It is quite evident that the unknown weights for which the fitness function e_j approaches zero, the solution *y(t)* of the equation can be approximated by the model of ŷ(t) given in (3).

3 Stochastic Solvers

In this section, a brief introduction has been presented for different stochastic solvers used in the article.

Simulating annealing is kind of probabilistic computational method used for local and global optimization problems in applied sciences and engineering. It is a technique inspired from material heating and controlled cooling characteristics. Its aim is to find efficiently and effectively the required objective parameters in fixed amount of time instead of the best solutions. This method is still widely used by research community for optimization since its introduction in 1983 [18-19].

Pattern search (PS) belongs to the numerical optimization methods that do not require the gradient of the problem under consideration. The capabilities of the PS are also on the functions which are not continuous or differentiable. Hooke and Jeeves are the researchers who invented PS and applied it in the statistical problems [20]. Pattern search computes a sequence of points that approach an optimal point. At each step, the algorithm searches a set of points, called a mesh, around the point computed at the previous step of the algorithm. The mesh is formed by adding the current point to a scalar multiple of a set of vectors called a pattern [21]. If the pattern search algorithm finds a point in the mesh that improves the objective function at the current point, the new point becomes the current point at the next step of the algorithm. Pattern search is very handy for the optimization problems as well as parallel computing [22].

Interior point methods also referred to as barrier methods are a certain class of algorithms to solve the linear and nonlinear convex optimization problems. These algorithms have been inspired by Karmarkar's algorithm, developed by Narendra Karmarkar in 1984 for linear programming [23]. IPA is a constraint minimization problem solving method. The basic elements of the method consist of a self-concordant barrier function used to encode the convex set. Contrary to the simplex method, it reaches an optimal solution by traversing the interior of the feasible region [24]. To solve the approximate problem, the algorithm uses either a Newton step via a linear programming or conjugate gradient step using a trust region during the each iteration [25].

Evolutionary computational intelligence based on Genetic algorithm is inspired by natural evolution. It is an optimization mechanism successfully applied for solving various problems. The main aim of the algorithm is to have good solution to the problem in a large search space of candidate solutions. It is well known that GA algorithms are best in controlling the robustness; avoid local minima, free from divergence, viable as compared with other heuristic mathematical solvers [26-27]. The generic flow diagram of evolutionary algorithm is given in Fig. 2.

In this article, our intent is to use GA, SA and PS techniques hybridized with IPA algorithm for learning of weights. MATLAB GA and Direct search tool box is used for GA, PS and SA algorithms. Moreover, the evolutionary algorithm based on GA algorithm is given in the following steps:

Step 1: *Initialized Population*: Randomly generated initial population with bounded real values represented Chromosomes or individuals. Each individual contains as many elements as the number of weights in neural network. Enough spread is made in the initial population for better search space of algorithm.

Step 2: *Initialization*: The parameters setting are initialized before algorithm execution. Set the number of variable, the number of generations, the fitness limit, elite count and value of crossover fraction as 0.80 for reproduction. Set also Migration in both directions. Start generation count, etc.

Step 3: *Fitness Evaluation*: Calculate the fitness for each individual using the expressions like (7), (8) and (9).

Step 4: *Ranking:* Each individual in the populations is ranked on the basis of fitness values.

Step 5: *Termination Criteria:* Terminate the algorithm for predefined fitness value i.e. MSE 10^{-09} is achieved or number of generation is complete. If termination criterion meets, then go to step 8 else continue

Step 6: *Reproduction:* To reproduces the next generation by

Crossover: Call for Heuristic function, Mutation: Call for Adaptive Feasible function, Selection: Call for Stochastic Uniform function, and Elitism.

Repeat the procedure from step 3 to step 6 until total number of generations are complete

Step 7: *Storage:* Store the global best individual of this run. Repeat the step 2 to 6 to have sufficient numbers of independent runs of the algorithm for better statistical analysis.

Step 8: *Refinement:* Interior point algorithm is used for further refinement of results. Take the best individual of GAs as starting point for the algorithm. MATLAB optimization tool box is used for Interior point algorithm. Store also the refined global best individual.

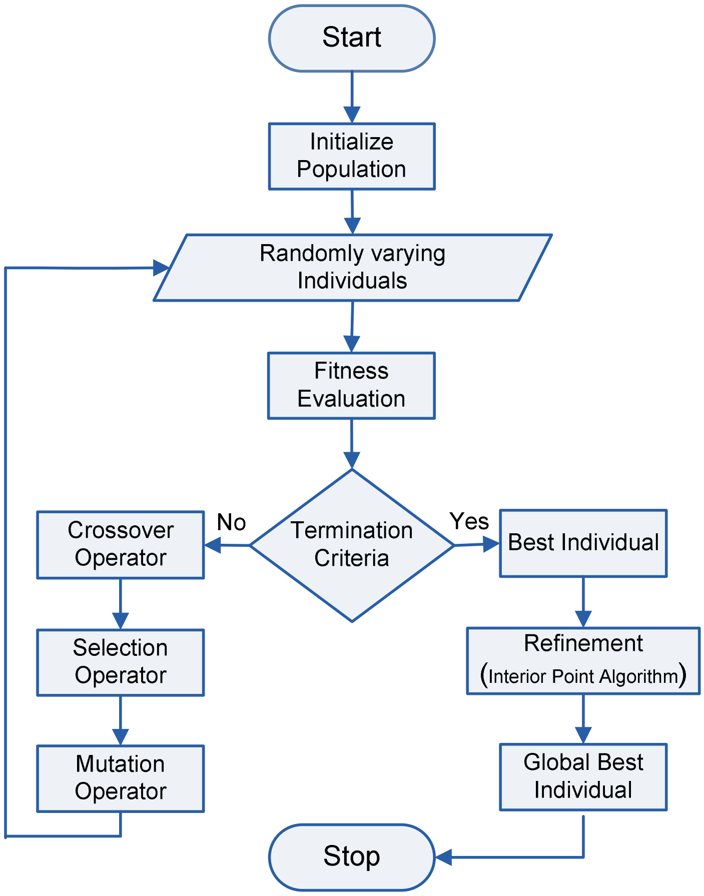


Fig. 2. Evolutionary Computation Algorithm Flowchart

The parameter setting used in this paper for GA and IPA algorithms are given in Table 1, while for SA and PS techniques is shown in Table 2.

Table 1. Parameters Setting for GA and IPA Algorithms

GA		IPA	
Parameters	*Setting*	*Parameters*	*Setting*
Population Creation	Constrain dependant	Start Point	Randomly between (-1,1)
Scaling faction	Rank	Derivative	Approximate by solver
Selection function	Stochastic Uniform	Sub-problem algorithm	IDI factorization
Crossover fraction	0.80	Scaling	Objective and constraints
Crossover function	Heuristic	Maximum Perturbation	0.1
Mutation	Adaptive feasible	Finite Difference types	Forward Differences
Elite count	3	Hessian	BFGS
Initial Penalty	10	Minimum Perturbation	10^{-08}
Penalty factor	100	X-Tolerance	10^{-10}
Migration fraction	0.2	Max Iteration	2000
Migration interval	20	Max. function evolutions	25000

Table 2. Parameters Setting for SA and PS Algorithms

SA		PS	
Parameters	*Setting*	*Parameters*	*Setting*
Start Point	Randomly between (0, 1)	Start Point	Randomly between (0, 1)
Annealing Function	Fast Annealing	Polling order	Consecutive
Reannealing Interval	100	Mesh Accelerator	Off
Temperature Update Function	Exponential	Mesh Rotae/Scale	On
Acceptance Criteria	Simulating annealing acceptance	Poll method	GPS Positive basis 2N
Stall Iteration	15000	Mesh expansion factor	2.0
Initial Temperature	100	Mesh Contraction factor	0.51
Maximum Iteration	30000	Maximum Iteration	3000
Maximum Function Evolutions	90000	Maximum function evolutions	100000

4 Simulation and Results

In this section, results of the detailed simulations have been presented for Painlevé equation I. The exact solution of this nonlinear ordinary differential equation is not known; therefore, its comparative studies are presented only with other stochastic as well as approximate analytic numerical techniques.

Mathematical modeling of the equation is formulated with DENN networks represented by expression (3) and (5) by taking *10* number of neurons, which result into *30* number of unknown weights of the networks. These weights are bounded real numbers between *-10* to *10*. The initial population consists of a set of *240* individuals, which is divided into *8* subpopulations each with *30* individuals. Each individual has *30* elements, which is equivalent to number of unknown parameters of DENN networks. Input of the training set is taken as $t \in (0, 1)$ with a step of *0.1*. Therefore, the fitness function as given in equation (7) in this case can be formulated as

$$e_j = \frac{1}{11}\sum_{i=1}^{11}[\frac{d^2\hat{y}(t_i)}{dt_i^2} - 6\{\hat{y}(t_i)\}^2 - t_i]^2 + \frac{1}{2}[\{\hat{y}(0)\}^2 + \{\frac{d}{dt}\hat{y}(0) - 1\}^2]\Bigg|_j \tag{10}$$

where, j is the number of generations, $\hat{y}$ and $d^2\hat{y}/dt^2$ are networks provided in equations (3), and (5), respectively. The proposed scheme runs iteratively in order to compute the minimum of fitness function with a termination criteria as 1600 number of generations or the value of the fitness function $e_j \leq 10^{-09}$ whichever comes earlier. The best individual found by GA technique is passed to IPA algorithm as a start point for rapid local search. Optimization of the fitness function (10) is also done with SA and PS techniques by using the parameter setting used given in Table 2. One of the best set of weights to DENN learned stochastically by GA hybrid with IPA (GA-IPA) and PS hybrid with IPA (PS-IPA) are given in Table 3, while for SA hybrid with IPA (SA-IPA) and IPA algorithm alone is shown in Fig. 3(a) and Fig. 3(b), respectively. These weights can be used in (3) to obtain the solution of Painlevé I for any input time *t* between *0* and *1*.

Table 3. **W**eights Trained by GA-IPA and PS-IPA Algorithms

	GA-IPA			***PS-IPA***		
i	w_i	α_i	β_i	w_i	α_i	β_i
1	-0.09419252	4.40969456	0.71523375	-1.78933408	-1.12659854	1.16741047
2	-7.74586194	-6.53184804	9.91763408	6.81748687	9.85249048	-8.84014695
3	-0.49586318	1.78392014	2.06442263	-3.59430726	1.04319752	1.88017204
4	-4.72851231	-0.43163723	2.74248908	-0.43443994	0.69086185	2.60000911
5	-0.61856008	-0.15933885	-2.33076773	-2.13391965	-0.74371424	-0.33900757
6	1.70851031	1.82676926	1.58142028	-8.66716665	1.60682702	9.86368403
7	8.68663701	-0.95884483	-6.99811597	-3.28087403	0.81147957	0.35898406
8	8.00958999	1.58562081	-6.57768753	-1.75825624	-1.28078620	-0.05537760
9	-0.16835997	1.61553058	2.12009645	-2.96689160	-1.09785526	2.12656963
10	-2.21614693	-1.13364609	-0.05052634	-2.28235189	-1.12692452	1.18012342

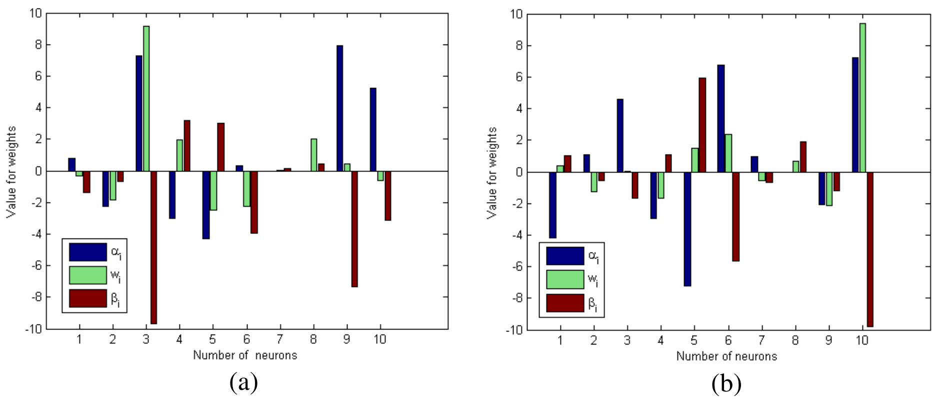


Fig. 3. The weights trained for DENN networks (a) for SA-IPA, (b) for IPA algorithm

Table 4. Summary of the results for Painlevé I

	$\check{\mathbf{y}}(\mathbf{t})$		$\hat{\mathbf{y}}(\mathbf{t})$			
	VIM	*HPM*	*GA-IPA*	*PS-IPA*	*SA-IPA*	*IPA*
0.0	0.00000000	0.00000000	-0.00039196	-0.00009893	-0.00028524	-0.00019076
0.1	0.10021675	0.10021675	0.09980365	0.10011231	0.09991484	0.10001106
0.2	0.20213945	0.20213945	0.20170181	0.20203160	0.20181712	0.20190915
0.3	0.30863075	0.30863075	0.30815596	0.30851505	0.30828301	0.30837833
0.4	0.42398628	0.42398629	0.42345876	0.42385344	0.42359728	0.42370064
0.5	0.55433991	0.55434012	0.55372347	0.55418851	0.55388222	0.55399693
0.6	0.70845966	0.70846209	0.70772733	0.70828524	0.70792054	0.70805274
0.7	0.89922969	0.89924992	0.89831646	0.89901945	0.89855904	0.89872394
0.8	1.14639825	1.14653134	1.14532872	1.14623689	1.14563612	1.14584789
0.9	1.48177895	1.48251774	1.48078306	1.48210078	1.48124971	1.48154291
1.0	1.95942104	1.96303937	1.96099008	1.96263649	1.96147223	1.96190600
1.1	2.67624476	2.69235969	2.68210924	2.68863694	2.68619845	2.68651955
1.2	3.81549745	3.88205938	3.72145787	3.78667552	3.78268911	3.77519443
1.3	5.74150466	5.99957450	4.97894878	5.24591958	5.28519638	5.23208793
1.4	9.21266839	10.15837961	6.15133119	6.78950450	6.97977217	6.80867366
1.5	15.87246747	19.16051447	6.99170361	8.08090883	8.48183210	8.13726415
1.6	29.40316664	40.27336198	7.48178522	8.98117383	9.54431170	9.02949288
1.7	58.27239041	92.49760981	7.72898187	9.54051283	10.17677508	9.53261660
1.8	122.32053949	225.09720711	7.83899494	9.86881508	10.51077040	9.78093116
1.9	268.59971659	563.40155337	7.87872259	10.05879080	10.67133968	9.88817221
2.0	609.39725329	1418.35049652	7.88382437	10.17037746	10.74012436	9.92470932

The solutions obtained from the individuals in Table 3, Fig 3(a) and Fig 3(b) are given in Table 4 for inputs between 0 and 2 with step of 0.1. It also includes the results of 3rd iterate variational iteration method (VIM) and Homotopy perturbation method (HPM) for the same inputs [28-29]. It is further elaborated that VIM method, based on a polynomial of 38 degree with 34 terms, while the solution for HPM method is represented by 20 terms based on a polynomial of 22 degree. It can be seen from the results given in Table 4 that solution provided by our scheme is in good agreement with VIM and HPM methods for inputs between 0 and 1. However, for the values greater than ‘1’ the analytical methods diverge while the results by stochastic solvers are still convergent. It is interesting to see the differences in the results near *t* = *2* are very large while the difference of the results for stochastic solvers is very small.

As the exact solution for Painlevé I is unknown, so the following criteria is made for the comparison of our proposed solution ŷ(t) and approximate analytical solvers y̌(t).

$$\varepsilon(t) = [\frac{d^2 y(t)}{dt^2} - 6\{y(t)\}^2 - t]^2 \tag{11}$$

It means that the solution that provides the minimum of expression (11) will lead to the accurate solution of Painleve I. The results are determined between the time 0 to 2 with a step of 0.1 for stochastic and analytical solvers and is given in the Table 5. It can be found that the results for input less than 0.8, the VIM and HPM techniques are remarkable, while the results by proposed scheme remain consistent with an accuracy of 10^{-06} to 10^{-08} between 0 and 1. It can also be seen that for $t \in (1, 2)$ the error of the given proposed scheme is much less than that of VIM and HPM techniques. It is difficult to compare analytical solvers like VIM and HPM with the stochastic solvers, however, one has to go through the complex mathematical procedures to obtain the results for such analytical solvers, whereas such issues are not dealt with stochastic solvers. Beside this, the advantages of the stochastic schemes are robustness, simplicity of the concept, ease in implementation and broader application domain.

Table 5. Comparison of the results for Painlevé I based on function $\varepsilon(t)$

t	y̌(t)		ŷ(t)					
			DENN trained for (0, 1) & step 0.1				DENN trained for (0, 2) & step 0.2	
	HPM	VIM	GA-IPA	PS-IPA	SA-IPA	IPA	GA-IPA	IPA
0.0	0	0	3.22E-08	3.88E-08	1.79E-08	6.75E-08	2.81E-08	1.04E-08
0.1	1.93E-34	5.28E-24	1.80E-08	1.93E-07	2.62E-08	1.10E-06	2.36E+02	2.29E+02
0.2	4.93E-32	2.29E-17	1.15E-07	3.34E-08	1.66E-07	1.36E-06	2.74E-07	4.21E-08
0.3	5.80E-26	1.84E-13	2.97E-07	4.53E-07	3.40E-08	1.60E-08	8.68E+01	8.92E+01
0.4	1.04E-20	1.16E-10	2.05E-06	3.94E-07	1.72E-06	2.35E-06	5.54E-07	4.89E-08
0.5	1.41E-16	1.88E-08	2.95E-06	4.40E-07	6.30E-06	5.30E-06	8.69E-02	7.73E-02
0.6	3.65E-13	1.32E-06	2.10E-06	3.66E-06	6.60E-06	4.45E-06	6.91E-06	3.11E-07
0.7	3.03E-10	5.37E-05	7.28E-07	3.41E-06	3.03E-06	1.83E-06	1.74E-03	1.07E-03
0.8	1.10E-07	1.51E-03	2.42E-07	9.59E-07	9.57E-07	4.60E-07	1.22E-05	1.19E-07
0.9	2.13E-05	3.32E-02	5.70E-08	8.42E-08	1.62E-07	6.13E-08	1.52E-04	4.90E-05
1.0	2.54E-03	6.18E-01	6.59E-09	2.11E-09	1.03E-08	3.56E-09	7.34E-05	1.17E-06
1.1	2.10E-01	1.04E+01	1.17E+02	3.8.1E+01	4.28E+01	5.04E+01	6.25E-06	3.23E-07
1.2	1.30E+01	1.65E+02	3.64E+03	2.34E+03	1.94E+03	2.25E+03	4.43E-06	2.87E-06
1.3	6.53E+02	2.63E+03	2.53E+04	2.48E+04	2.19E+04	2.32E+04	4.91E-05	6.16E-07
1.4	2.86E+04	4.29E+04	6.97E+04	9.32E+04	9.89E+04	9.38E+04	8.90E-05	1.84E-05
1.5	1.17E+06	7.43E+05	1.09E+05	1.88E+05	2.30E+05	1.98E+05	5.42E-06	3.03E-05
1.6	4.69E+07	1.40E+07	1.31E+05	2.70 E+05	3.50E+05	2.81E+05	2.34E-04	2.63E-05
1.7	1.92E+09	2.91E+08	1.39E+05	3.26 E+05	4.26E+05	3.27E+05	6.28E-04	3.24E-04
1.8	8.11E+10	6.73E+09	1.42E+05	3.60 E+05	4.65E+05	3.48E+05	6.69E-05	6.06E-06
1.9	3.45E+12	1.72E+11	1.43E+05	3.80 E+05	4.82E+05	3.54E+05	6.38E-03	4.71E-03
2.0	1.43E+14	4.77E+12	1.42E+05	3.93 E+05	4.88E+05	3.55E+05	5.67E-06	5.44E-07

Furthermore, the accuracy of the proposed methods can be increased further by training of the weights for the larger intervals. In this regard, DENN networks are optimized with GA-IPA and IPA algorithms for interval (0, 2) with a step of 0.2. The weights obtained by learning are given in Fig.4. These weights are used in expression (3) for finding the solution of the equation in the interval 0 to 2. The results are also given in Table 5. It is quite evident from the table that the error for IPA and GA-IPA are in acceptable range. Moreover it is noted that on the points where we have not performed learning the value of the error is large. Such error can be reduced by the proposed scheme but at the cost of much greater computational budget.

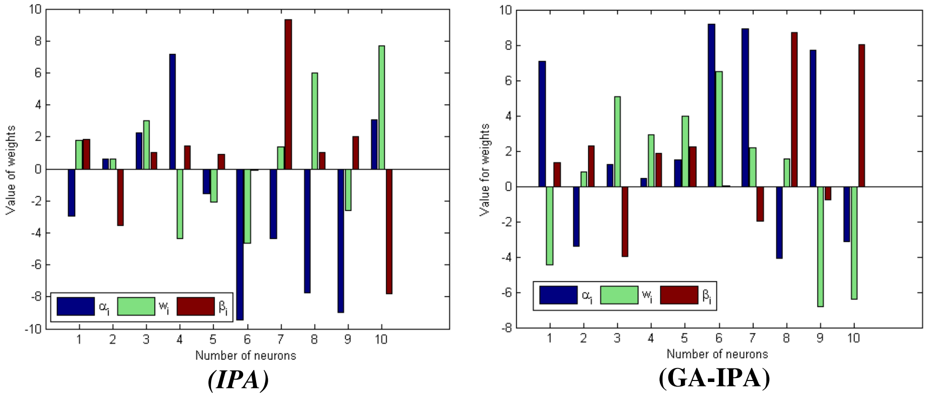


Fig. 4. The weights trained for DENN networks (a) for IPA, (b) for GA-IPA algorithms

In order to see the behavior of the proposed schemes on the intermediate points, we have trained DENN networks using IPA and GA-IPA on interval (0, 2) with a step of 0.1. In this case, the values of weights are restricted between (-100, 100). One of the weights trained stochastically using IPA and GA-IPA are presented in Table 6. The value of the function $\varepsilon(t)$ as given in equation (11) is determined using the values of weights given in Table 6. The results are provided in Table 7 for $t \in (0, 2)$ with a step of 0.05. It can be seen from the table the accuracy of the results increases by using the small step in training of the DENN networks. Comparing the results given in Table 5 and Table 7, it can be inferred that the values of $\varepsilon(t)$ at intermediate points is significantly reduced. For example, the maximum and minimum values of $\varepsilon(t)$ for GA-IPA from Table 5 are $2.36\times10^{+02}$ and 2.81×10^{-08}, respectively, while from Table 7 the maximum and minimum values are reduced to 5.68×10^{-02} and 2.97×10^{-14}, respectively.

The proposed schemes are based on stochastic solvers, so their reliability and effectiveness can only be validated through detailed statistical analysis. In this regard, 75 numbers of independent runs for IPA, SA-IPA, PS-IPA, and GA-IPA algorithms are executed for finding the appropriate DENN weights. The best and worst are defined as the minimum and maximum value of function e_j as given in equation (7), respectively. Moreover, the statistical parameter of mean and standard deviation

(STD) are also estimated to check the scatterness of the results. The results of statistical analysis are provided in Table 8 for inputs *t* between 0 and 1 with a step of 0.2. It can be seen from the table that the best value at given input points is not consistent for a specific algorithm. Whereas, the values for the worst, mean and STD are consistently lowest for GA-IPA algorithm as compared to the values obtained for IPA, SA-IPA, and PS-IPA algorithms. Therefore, the GA-IPA demonstrates its supremacy on other stated stochastic algorithms.

Table 6. DENN Weights Trained by GA-IPA and IPA Algorithms for t ϵ (0, 2) with step 0.1

	GA-IPA			***PS-IPA***		
i	w_i	α_i	β_i	w_i	α_i	β_i
1	1.6277311735	-1.9640565815	-0.8308272114	20.7071003034	-4.3220074424	1.8575418198
2	0.9940224901	31.6451659694	-3.4626214760	-1.5264427244	5.8057402316	1.3458613318
3	-2.1476011252	2.8630336114	4.4785310648	6.2901282731	24.7072473708	-18.5003559282
4	22.2687477933	-5.7347562305	1.7021049359	1.6256389001	5.9421024089	-2.0060440152
5	24.7127339783	23.4462894840	0.5126163015	-1.7082648989	3.8677694367	5.3506636949
6	0.4637975905	-11.1647128580	-3.2030010653	26.2564250904	8.2052029101	0.3111031742
7	-25.0026019350	19.6095358672	-0.4027976287	22.9254315835	2.8896848305	0.5142521509
8	-0.7723237910	4.1733994626	1.8066328239	-2.8327763585	4.4550452365	11.1157520327
9	-6.1282451011	-39.2172826259	18.2566230435	26.0776834920	-8.7031590254	0.1980018290
10	0.0652423477	15.9679030529	2.9883904823	-2.3825613370	-11.6487695882	7.2010539798

Table 7. Comparison of the Results based on value of Function $\varepsilon(t)$

t ϵ (0, 1)			t ϵ (1, 2)		
T	IPA	GA-IPA	t	IPA	GA-IPA
0.00	8.93138905E-15	2.96786048E-14	1.05	6.87284565E-09	2.82837938E-09
0.10	7.64293387E-13	1.14700655E-12	1.10	5.29072203E-11	2.17723047E-09
0.15	8.04058697E-02	5.68451868E-02	1.15	3.27985472E-09	1.67912064E-08
0.20	9.17657716E-11	3.96855328E-12	1.20	8.30889828E-09	2.77909622E-08
0.25	1.34199584E-04	9.26695097E-05	1.25	7.41833106E-09	2.04858379E-08
0.30	9.93722701E-09	4.02880669E-10	1.30	2.24501736E-09	4.04241353E-09
0.35	3.17074099E-06	2.13702657E-06	1.35	6.37618280E-11	2.19347491E-09
0.40	1.89576156E-07	9.27401210E-08	1.40	3.14669944E-09	2.19496228E-08
0.45	2.54817898E-07	2.21201379E-07	1.45	5.53076252E-09	3.89668392E-08
0.50	3.69339842E-07	2.87815947E-07	1.50	2.58086613E-09	2.72008859E-08
0.55	8.90738070E-08	6.72326886E-08	1.55	6.77226162E-11	2.61125061E-09
0.60	2.36862171E-09	2.28745281E-09	1.60	5.65943902E-09	1.06854328E-08
0.65	6.78727097E-08	5.63774307E-08	1.65	1.16398922E-08	4.92622905E-08
0.70	9.49389552E-08	7.87542587E-08	1.70	4.57663966E-09	4.83100461E-08
0.75	5.44449495E-08	4.41008028E-08	1.75	2.45555186E-09	3.28013602E-09
0.80	9.94492350E-09	6.57809884E-09	1.80	3.19837612E-08	3.68076772E-08
0.85	1.18614278E-09	2.54814926E-09	1.85	3.04899533E-08	9.01184546E-08
0.90	1.65841203E-08	2.02236284E-08	1.90	1.34180367E-08	4.73597042E-10
0.95	2.71195877E-08	2.93014949E-08	1.95	3.14913209E-07	4.69653377E-07
1.00	2.06676906E-08	1.84347082E-08	2.00	6.28509811E-10	1.60490096E-10

Table 8. Statistical Analysis for Stochastic Solvers

T	Algorithm	Best e_j	*Worst* e_j	*Mean*		STD	
				$\hat{y}(t)$	e_j	$\hat{y}(t)$	e_j
0.2	IPA	6.95E-09	0.000197	0.199095	1.67E-05	0.005190	2.75E-05
	SA-IPA	3.40E-08	0.000109	0.199647	1.58E-05	0.002284	1.99E-05
	PS-IPA	3.34E-08	0.063693	0.192070	0.001395	0.042241	0.00862
	GA-IPA	4.57E-08	2.19E-05	0.201029	6.36E-06	0.001133	5.80E-06
0.4	IPA	7.53E-08	0.000953	0.420263	3.76E-05	0.006325	0.000118
	SA-IPA	2.85E-09	0.000126	0.420921	2.12E-05	0.002823	2.40E-05
	PS-IPA	3.49E-08	0.055404	0.406569	0.000866	0.082342	0.006446
	GA-IPA	8.43E-08	6.15E-05	0.422626	1.08E-05	0.001381	1.02E-05
0.6	IPA	4.10E-08	0.001425	0.703277	8.77E-05	0.008859	0.000172
	SA-IPA	5.01E-07	0.000327	0.704203	8.87E-05	0.003947	7.12E-05
	PS-IPA	7.49E-07	0.179667	0.681390	0.003056	0.131618	0.021278
	GA-IPA	2.10E-06	0.000159	0.706578	4.06E-05	0.001941	3.85E-05
0.8	IPA	4.60E-07	0.004384	1.137980	0.000129	0.014466	0.000524
	SA-IPA	8.18E-07	0.000261	1.139475	3.82E-05	0.006535	5.08E-05
	PS-IPA	6.95E-08	0.035404	1.104146	0.000603	0.203184	0.004101
	GA-IPA	1.89E-07	0.000661	1.143407	2.12E-05	0.003202	7.62E-05

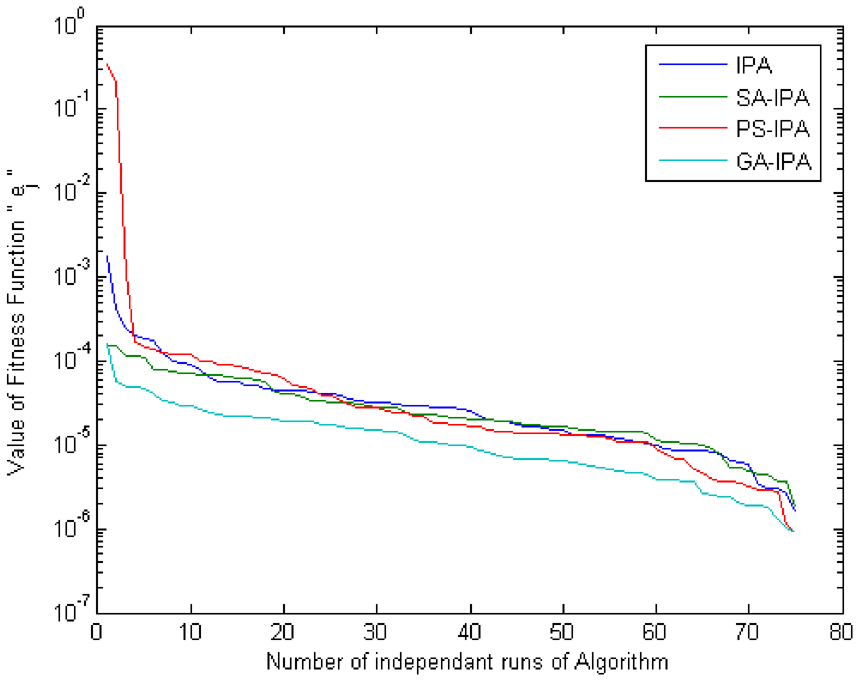


Fig. 5. Comparisons of different stochastic numerical solvers

Moreover, the behavior of the algorithms must be investigated on the basis of fitness achieved on the training interval instead of value for fitness at specific point. The values of the fitness function e_j are determined on interval *(0, 1)* with a step of *0.1* for 75 independent runs of IPA, PS-IPA, SA-IPA and GA-IPA algorithms. The values of fitness function e_j are plotted against independent runs in Fig 5. These runs are arranged by descending order of the fitness values. It can be seen from the figure that for IPA and PS-IPA about 5% to 10% of the independent runs are failed to provide convergence up to a reliable accuracy, while SA-IPA and GA-IPA are 100% consistent in providing the convergent results. Comparing the results obtained from GA-IPA and SA-IPA, it is observed that GA-IPA provides consistently the lowest value of the fitness than that of SA-IPA.

5 Conclusion

On the basis of the simulations and results, it can be concluded that Painlevé I can be solved by the designed computational intelligence algorithm. The DENN networks of the equation trained by GA-IPA algorithm is the best stochastic optimizer as compared to PS-IPA, SA-IPA and IPA algorithms. On the basis of the statistical analysis, it can be inferred that the proposed computational approaches are reliable and effective. For smaller inputs the methods like VIM and HPM are better while for larger inputs the stochastic solvers are much better. In our future work, we intend to use other computational intelligence algorithms to solve other Painlevé transcendents.

References

[1] Bassom, A.P., Clarkson, P.A., Hicks, A.C.: Numerical studies of the fourthPainlevè equation. IMA Journal of Applied Mathematics 50, 167–139 (1993)

[2] He, J.H.: Variational iteration method, a kind of non-linear analytical technique, some examples. Internat. J. Nonlin. Mech. 34(4), 699–708 (1999)

[3] He, J.H.: Some asymptotic methods for strongly nonlinear equation. Int. J. N. Phy. 20(10), 1144–1199 (2006)

[4] Wazwaz, A.M.: A first course in integral equations. WSPC, New Jersey (1997)

[5] Tajiri, M., Kawamoto, S.H.: Reduction of Kdv and cylindrical KdV equationsto Painlevè equation. Journal of the Physical Society of Japan 51, 1678–1681 (1982)

[6] Wazwaz, A.M.: Construction of solitary wave solution and rational solutions for the KdV equation by ADM. Chaos, Solution and Fractals 12, 2283–2293 (2001)

[7] Ghorbani, A., Nadjfi, J.S.: He's homotopy perturbation method for calculating Adomian's polynomials. Int. J. Nonlin. Sci. Num. Simul. 8(2), 229–332 (2007)

[8] Biazar, J., Babolian, E., Islam, R.: Solution of the system of ordinary differential equations by Adomian decomposition method. Applied Math. Comput. 147(3), 713–719 (2004)

[9] He, J.H.: Variational approach to the Thomas-Fermi equation. Appl. Math. Comput. 143, 533–535 (2003)

[10] Golbabai, A., Keramati, B.: Solution of non-linear Fredholm integralequations of the first kind using modified homotopy perturbation. Chaos, Solution and Fractals 5, 2316–2321 (2009)

[11] Aarts, L.P., Veer, P.V.D.: Neural Network Method for solving the partial Differential Equations. Neural Processing Letter 14, 261–271 (2001)

[12] Rarisi, D.R., et al.: Solving differential equations with unsupervised neural networks. J. Chemical Engineering and Processing 42, 715–721 (2003)

[13] Raja, M.A.Z., Khan, J.A., Qureshi, I.M.: A new Stochastic Approach for Solution of Riccati Differential Equation of Fractional Order. Ann. Math. Artif. Intell. 60(3-4), 229–250 (2010), doi:10.1007/s10472-010-9222-x

[14] Raja, M.A.Z., Khan, J.A., Qureshi, I.M.: Solution of fractional order system of Bagley-Torvik equation using Evolutionary computational intelligence. Mathematical Problems in Engineering, 1–18 (2011), doi:10.1155/2011

[15] Zahoor Raja, M.A., Khan, J.A., Qureshi, I.M.: Evolutionary Computational Intelligence in Solving the Fractional Differential Equations. In: Nguyen, N.T., Le, M.T., Świątek, J. (eds.) ACIIDS 2010, Part I. LNCS (LNAI), vol. 5990, pp. 231–240. Springer, Heidelberg (2010)

[16] Raja, M.A.Z., Khan, J.A., Qureshi, I.M.: Swarm Intelligent optimized neural networks for solving fractional differential equations. International Journal of Innovative Computing, Information and Control 7(11), 6301–6318

[17] Khan, J.A., Raja, M.A.Z., Qureshi, I.M.: Stochastic Computational Approach for Complex Non-linear Ordinary Differential Equations. Chin. Phys. Lett. 28(2), 020206 (2011), doi:10.1088/0256-307X

[18] Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by Simulated Annealing. Science, New Series 220(4598), 671–680 (1983)

[19] Kroumov, V., Yu, J., Shibayama, K.: 3D Path Planning for mobile robots using simulated annealing neural network. International Journal of Innovative Computing, Information and Control 6(7), 2885–2899 (2007)

[20] Hooke, R., Jeeves, T.A.: Direct search solution of numerical and statistical problems. Journal of the Association for Computing Machinery (ACM) 8(2), 212–229 (1961)

[21] Dolan, E.D., Lewis, R.M., Torczon, V.J.: On the local convergence of pattern search. SIAM Journal on Optimization 14(2), 567–583 (2003)

[22] Torczon, V.J.: On the convergence of pattern search algorithms. SIAM Journal on Optimization 7(1), 1–25 (1997)

[23] Karmarkar, N.: A New Polynomial Time Algorithm for Linear Programming. Combinatoric 4(4), 373–395 (1984)

[24] Wright, S.: Primal-Dual Interior-Point Methods. SIAM, Philadelphia (1997) ISBN 0-89871-382-X

[25] Wright, M.H.: The interior-point revolution in optimization: history, recent developments, and lasting consequences. Bull. Amer. Math. Soc. (N.S) 42, 39–56 (2005)

[26] García, P., Mingo, J.D., Carro, P.L., Valdovinos, A.: Efficient Feedforward Linearization Technique Using Genetic Algorithms for OFDM Systems. Journal on Advances in Signal Processing 1(354030), 10 (2010), doi:10.1155/2010/354030

[27] Reeves, C.R., Rowe, J.E.: Genetic algorithms principles and perspective: A guide to GA Theory. Kluwer Academic Publishers, Norwell (2003)

[28] Hesameddini, E., Peyrovi, A.: The use of variational iteration method and Homotopy perturbation method for Painlevé equation I. Applied Mathematics Sciences 3, 1861–1871 (2009)

[29] Behzadi, S.S.: Convergence of Iterative Methods for Solving Painlevé Equation. Applied Mathematical Science 4(30), 1489–1507 (2010)

Chapter 8
An Investigation into the Adaptive Capacity of Recurrent Neural Networks

N.H. Siddique[1] and B.P. Amavasai[2]

[1] School of Computing and Intelligent Systems, University of Ulster,
Northland Road, Londonderry, BT48 7JL
nh.siddique@ulster.ac.uk
http://www.infm.ulst.ac.uk/~siddique

[2] Reading Innovation Centre, Procter and Gamble, 460 Basingstoke Road,
Reading RG2 0QE, UK
amavasai.b@pg.com

Abstract. The characteristic feature of any intelligent system is its ability to appropriately adjust its behaviour in response to a change of the environment and/or change in the structure of the system. Such a feature or behaviour of a system is termed as “adaptive behaviour”, which is a measure of the learning ability of a system. Feed-forward neural networks have commonly been used to model such behaviours. However the weights of feed-forward neural networks remain static once strained and so can hardly be categorised as adaptive. On the contrary, recurrent networks have the capability to exhibit dynamic behaviour. In general, the feedback connections in a recurrent network are made after the non-linear activation function. In this chapter we investigated network architectures with different feedback connections made before and after the non-linear activation function to observe adaptive capability of these networks. Backpropagation training algorithms are applied to these networks with a minimum number of recurrent neurons at which adaptive behaviour is attainable. Three benchmark problems are chosen to investigate the performances on learning ability of the proposed architectures and results are presented and analysed.

Keywords: Recurrent network, Learning capacity, Adaptive behaviour, Back propagation algorithm.

1 Introduction

All forms of multilayer perceptron networks (MLP), radial basis functions networks (RBF), generalised regression networks (GRN) and probabilistic networks (PN) are classified as feedforward networks. They are typically used as static networks in such varied applications like identification, control, prediction, speech generation and pattern recognition [1]-[7]. Research interest in recurrent neural networks (RNN) has grown over the years because of their capability of exhibiting dynamic behaviour and computational power [8]. RNNs have empirically shown the ability to perform inference in problems as diverse as grammar induction, demonstrated potential for applications such as time series prediction, process modelling, and process control [9]-[14]. These networks can be trained, depending on their architecture and weights, as oscillators, as associative memories and also as in finite automata. Generally, training of this class of neural networks has been difficult due to the excessive time required to converge [15], [16]. Stability issues of training in recurrent neural networks are addressed in [17], [18]. Some researchers have suggested improvements that can be made to well known algorithms, which can improve the training of recurrent networks. Practical problems during the learning process of recurrent networks are mainly due to the presence of local minima in the cost function [19]. The adaptation of recurrent neural networks with fixed weights has been investigated by many researchers. It has been demonstrated that dynamic networks can learn without changing their synaptic weights [20]-[24]. Bengio et al. showed that gradient-based learning algorithms face an increasingly difficult problem as the duration of the dependencies to be captured increases [25]. De Jesus et al. demonstrated that spurious minima are introduced into the error surface due to characteristics in the input sequence that make the training more difficult for gradient descent algorithms in recurrent network [15]. Other early results on recurrent training algorithms have been reported in [26]-[28].

In this chapter we investigated the RNN architectures with different feedback connections, mainly, before and after the application of the non-linear activation function to observe the adaptive capability of these networks with varying weights for the feed forward connections and fixed weights for the recurrent connections. Therefore, traditional backpropagation (BP) training algorithms can be applied to networks with minimum number of recurrent neurons at which adaptive behaviour is attainable. Three different benchmark problems were chosen to verify the performances of the proposed architectures of the recurrent neural networks.

The rest of the chapter is organised as follows. Section 2 presents an overview of different RNN architectures; the proposed architectures and their block diagrams are described in section 3. The training algorithm for the proposed RNN is presented in section 4. The benchmark problems, experimentation and analysis of results are presented in section 5. Finally the chapter concludes with some remarks in section 6.

2 Overview of Recurrent Architectures

In the past few decades several recurrent neural network models have been proposed [9], [10], [14], [29]-[36]. These recurrent architectures can be classified into three broad categories where feedback connections are taken from three locations in the feedback loop: synapse feedback, feedback from the neuron output before non-linearity and output feedback after non-linearity proposed by different researchers [9], [10], [13], [14], [37]. All of these architectures reported in the literature are summarised according to their feedback connectivity in Table 1.

Table 1. Summary of architectures

Architecture	Synapse feedback	Output feedback	
		before nonlinearity $f(.)$	after nonlinearity $f(.)$
Hopfield			yes
Elman (uses context units)	-	-	yes
Jordan (uses context units)	-	-	yes
William-Zipser (allows any neuron)	-	-	yes
Frasconi-Gori-Soda	-	yes	yes
De Vries-Principe	yes	-	-
Poddar- Unnikrishnan	-	-	yes
Back-Tsoi	yes	-	-

The structure of the original Jordan network has three layers, with the main feedback connections taken from the output layer to the context layer. It has been theoretically shown that the original Jordan network [32] is not capable of representing arbitrary dynamic systems. However, by adding the feedback connections from the hidden layer to the context layer, (similar to the case of the Elman network [9]) a modified Jordan network is obtained. The modified Jordan network can be trained using the standard BP algorithm to model different dynamic systems. The values of the feedback connection weights have to be fixed by the user if the standard BP algorithm is employed.

The Elman architecture differs from the above in that it uses an extra layer of context neurons to copy hidden layer outputs and after delaying these values for one time unit, feeds them back as additional inputs to hidden layer neurons [9]. The idea of introducing self-feedback connections for the context units was borrowed from Jordan [32]. The standard BP algorithm is employed to Elman networks.

The William-Zipser architecture allows any neuron in the network to be connected to any other neuron in the network [14]. The William-Zipser architecture typically suffers from a lack of stability, i.e., for a given set of initial values, activations of linear output units may grow without limit and it has a slow convergence [13]. Frasconi et al. experimented with a slightly different architecture where they introduced local activation feedback and output feedback taken from hidden layer neurons after it has passed through the non-linearity and weighted by a constant value [10]. Their architecture is represented by the following equation:

$$y(t) = f\left(k_1 y(t-1) + \sum_{i=1}^{n} w_i x_i + b \right) \tag{1}$$

Where x_i, $i = 1,2,...,n$ are the inputs, w_i is the weight, b is the bias, $y(t-1)$ is the output delayed by one time unit and k_1 is the weighting factor of the output.

The Back-Tsoi architecture is different from Frasconi et al. architecture in that the feedback is taken at the synapse output [13]. They introduced a synapse with a linear transfer function instead of a synapse with a constant weight. Poddar-Unikrishnan [33] introduced a memory neuron which remembers the past output values. This architecture has a feedback transfer function with one pole only. A critical review of the various feedback architectures that appeared in the literature can be found in [13].

The Hopfield network consists of a set of neurons and a corresponding set of unit delays, where the output of each neuron is fed back to each of the other neurons except itself via the delay units [38]. The number of feedback connections is equal to the number of neurons. It consists of *n* neurons. The discrete Hopfield network is described in discrete time as

$$y_i(t) = \Gamma\left(\sum_{j=1}^{n} w_{ij} y_j(t-1) + x_i - b_i\right) \quad for \quad j \neq i, i = 1,2,\cdots,n \tag{2}$$

Where $\Gamma()$ is the activation function defined as $\mathrm{sgn}(.)$, x_i is the external input to *i*th neuron and b_i is the bias. The feedback input to the *i*th neuron is equal to the weighted sum of neuron outputs y_j, where $j = 1,2,\cdots,n$. The matrix of synaptic weights W in the Hopfield model is an $n \times n$ symmetric matrix i.e. $w_{ij} = w_{ji}$ and the diagonal elements are zero i.e. $w_{ij} = 0$ for $\forall i = j$ and defined as

$$W = \begin{bmatrix} 0 & w_{12} & \cdots & w_{1n} \\ w_{21} & 0 & \cdots & w_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ w_{n1} & w_{n2} & \cdots & 0 \end{bmatrix} \tag{3}$$

Hence no connection exists from any neuron back to itself. Though the difference equation in (2) is highly coupled and nonlinear, the solution of the Hopfield model is stable when the weight matrix in (3) is symmetric and the biases are zero. This yields a non-increasing energy function for Hopfield network defined as

$$H(y) = -\frac{1}{2}\sum_i \sum_j w_{ij} y_i y_j \tag{4}$$

3 Proposed Network Architecture

The architecture considered in this study is a combination of Elman and Frasconi-Gori-Soda architecture with three layers of neurons in that it does not use any context layer and has a single unit time delay. Two types of feedback connections are investigated in this study: (i) output feedback before the activation function (non-linearity) and (ii) output feedback after the activation function (non-linearity). The two architectures can be described mathematically in the following equations. The recurrent architecture with feedback connections before nonlinearity is described by the equations (5)-(7).

$$y_k(t) = f\left(\left(\sum_{j=1}^{m} w_{kj} y_j^h(t)\right) + \theta_k\right), \ k = 1,2,\cdots,l \tag{5}$$

$$y_j^h(t) = f\left(net_j(t)\right) \tag{6}$$

$$net_j(t) = \left(\sum_{i=1}^{n} w_{ji} x_i + \theta_j\right) + \left(\sum_{j=1}^{m} w_{jj} K_s net_j(t-1)\right) \tag{7}$$

The architecture with feedback connections after nonlinearity can be described by the equations (5)-(6) and (8).

$$net_j(t) = \left(\sum_{i=1}^{n} w_{ji} x_i + \theta_j\right) + \left(\sum_{j=1}^{m} w_{jj} K_s y_j^h(t-1)\right) \tag{8}$$

where $i = 1,2,3,\cdots,n$, $j = 1,2,3,\cdots,m$ and $k = 1,2,3,\cdots,l$. n, m and l represent the maximum number of neurons in the input, hidden and output layer respectively. $y_k = \left[y_1, y_2, y_3, \cdots, y_l\right]^T$ are the outputs of the output-layer, $y_j^h(t) = \left[y_1^h(t), y_2^h(t), y_3^h(t), \cdots, y_m^h(t)\right]$ are the outputs of the hidden neurons after the nonlinear activation function, $net_j(t) = \left[net_1(t), net_2(t), net_3(t), \cdots, net_m(t)\right]^T$ are the outputs of the hidden neurons, $x_i = \left[x_1, x_2, x_3, \cdots, x_n\right]^T$ are the inputs, w_{kj} is the weight matrix of $l \times m$ dimension representing connectivity from hidden layer neurons to output layer neurons, w_{ji} is the weight matrix of $m \times n$ dimension representing connectivity from input layer to hidden layer neurons, $\theta_k = \left[\theta_{k1}, \theta_{k2}, \theta_{k3}, \cdots, \theta_{kl}\right]^T$ and $\theta_j = \left[\theta_{j1}, \theta_{j2}, \theta_{j3}, \cdots, \theta_{jm}\right]^T$ are biases of the output layer and the hidden layer neurons respectively. w_{jj} is the weight matrix of $m \times m$ dimension represents the recurrent connectivity between neurons in the hidden layer. The scaling factor K_s scales the feedback variables to neurons. The matrix w_{jj} can take the form of a full, symmetric, upper diagonal, lower diagonal, diagonal or a null matrix, which represents different connectivity between hidden-layer neurons. The two architectures proposed in this section, described by the equations (5)-(8), can be written in the following form, which will help understanding the block structure of the two networks.

$$y(t) = F_k\left(net_k(t)\right)I_k \tag{9}$$

$$net_k(t) = A_k y^h(t) + \Theta_k \tag{10}$$

$$y^h(t) = F\left(net_j(t)\right)I_j \tag{11}$$

$$net_j(t) = \left(A_j x + \Theta_j\right) + B\left(K_s net_j(t-1)\right) \tag{12}$$

$$net_j(t) = \left(A_j x + \Theta_j\right) + B\left(K_s y^h(t-1)\right) \tag{13}$$

The equations (9)-(11) and (12) represent the recurrent architecture with feedback connections before non-linearity and the equations (9)-(11) and (13) represent the recurrent architecture with feedback connections after non-linearity. $A_j = w_{ji}$, $A_k = w_{kj}$ and $B = w_{jj}$ are connection matrices, $\Theta_j = [\theta_1, \theta_2, \theta_3, \cdots, \theta_m]^T$ and $\Theta_k = [\theta_1, \theta_2, \theta_3, \cdots, \theta_l]^T$ are vectors of biases. $I_j = [1 \quad 1 \quad \cdots \quad 1]^T$ and $I_k = [1 \quad 1 \quad \cdots \quad 1]^T$ are vectors of $m \times 1$ and $l \times 1$ dimensions respectively. F_j and F_k are diagonal matrices of non-linear activation functions of hidden layer and output layer defined as

$$F_j = \begin{bmatrix} f_1(.) & 0 & \cdots & 0 \\ 0 & f_2(.) & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & f_m(.) \end{bmatrix} \tag{14}$$

$$F_k = \begin{bmatrix} f_1(.) & 0 & \cdots & 0 \\ 0 & f_2(.) & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & f_l(.) \end{bmatrix} \tag{15}$$

Thus, the equations (9)-(13) along with the definitions of the non-linear activation matrices defined by equations (14)-(15) can be described by the block notation form of the networks shown in Figures (1) and (2). While z^{-1} represents one unit of time delay element in the feedback path resulting in a recurrent network. The block notation, first used by Santini et al. [39] derived from Narendra's [2], are developed for the two network architectures to provide a clearer representation of the feedforward and feedback connectivity.

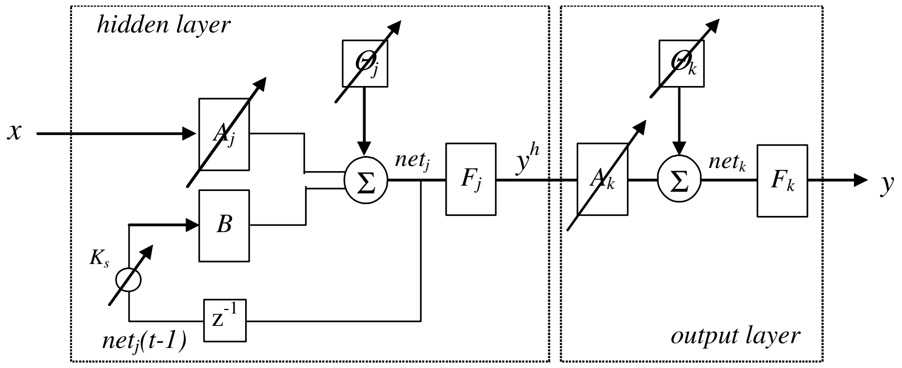


Fig. 1. Block diagram of the recurrent network with feedback before activation function

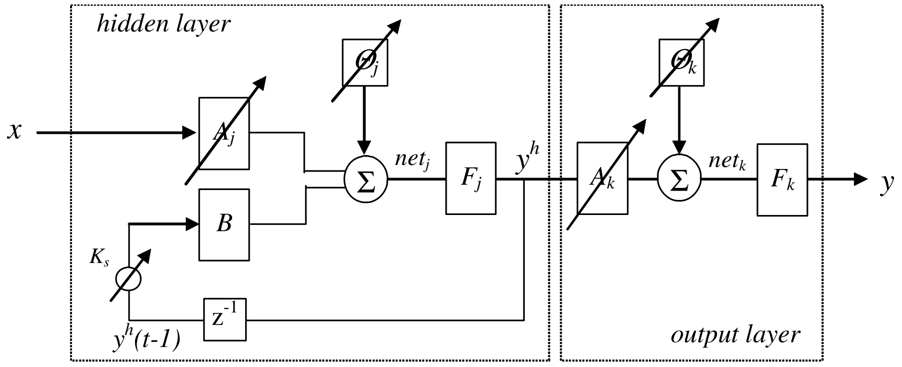


Fig. 2. Block diagram of the recurrent network with feedback after activation function

$B = w_{jj}$ in Figure 1 and Figure 2 represents the feedback connection matrix describing the feedback connectivity between hidden neurons and can take various forms depending on the connectivity described by the equations (16)-(26). The feedback connection matrix can be a full-matrix shown in equation (16), where the elements $\{b_{11}, b_{12}, \cdots, b_{mm}\} \in \Re$ are real values. The full-matrix implies that all neurons in the hidden layer are connected to each other in all possible ways, which yields a connection matrix (B_F) of the form

$$B_F = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1m} \\ b_{21} & b_{22} & \cdots & b_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m1} & b_{m2} & \cdots & b_{mm} \end{bmatrix} \tag{16}$$

When the connectivity matrix is a full matrix (B_F), it can be seen as a special case of the Williams-Zipser architecture and its performance is well-known for its slow convergence [14].

The connectivity matrix between hidden neurons can be symmetric, which means that the outputs are fed back to other neurons except themselves. Such feedback connectivity can be described by the matrix (B_S) given by equation (22) with diagonal elements zero.

$$B_S = \begin{bmatrix} 0 & b_{12} & \cdots & b_{1m} \\ b_{21} & 0 & \cdots & b_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m1} & b_{m2} & \cdots & 0 \end{bmatrix} \tag{17}$$

The feedback connection matrix can be an upper diagonal matrix (B_U) shown in equation (18) where neurons are connected to the next neuron in the layer. For example, neuron 1 is connected to neurons 2 to *n*, neuron 2 is connected to 3 to *n*, and neuron *n* does not have any connections.

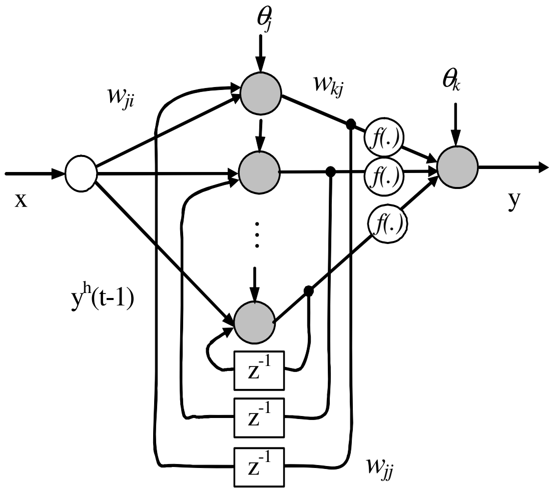
$$B_U = \begin{bmatrix} 0 & b_{12} & \cdots & b_{1m} \\ 0 & 0 & \cdots & b_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix} \tag{18}$$

The connectivity matrix can be a lower diagonal matrix (B_L) as shown in equation (19). The connections are just reversed to connections in matrix (B_U).

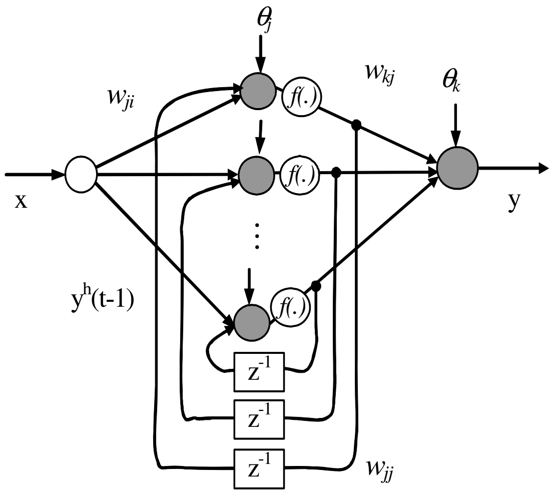
$$B_L = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ b_{21} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ b_{m1} & b_{m2} & \cdots & 0 \end{bmatrix} \tag{19}$$

The outputs of the hidden neurons can be fed back only to the neuron itself. In this case the weight matrix will be a diagonal matrix (B_D) described by equation (20). The connectivity is shown in Figure 3. The connectivity before non-linearity is shown in Figure 3(a) and after non-linearity is shown in the Figure 3(b).

$$B_D = \begin{bmatrix} b_{11} & 0 & \cdots & 0 \\ 0 & b_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & b_{mm} \end{bmatrix} \tag{20}$$



(a) Feedback connection before nonlinearity.



(b) Feedback connection after nonlinearity.

Fig. 3. Feedback connections to neurons themselves

If there are no feedback connections, it implies a feedforward network and yields a null matrix (B_N) in equation (21). In this case, the network is simply a feedforward network.

$$B_N = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix} \tag{21}$$

The symmetric matrix B_S , upper diagonal B_U and lower diagonal matrix B_L cases are abandoned for computational simplicity. The diagonal matrix B_D is chosen for detailed investigation in this chapter.

4 Training Algorithm of the RNN

Generally the training of RNNs have been difficult due to the time dependencies present in the architecture and difficulties in gradient descent algorithms. The most common error (criterion) function E for the training of dynamic neural networks is given by

$$E = \frac{1}{2}\sum_{t=0}^{T}\left(y_d(t) - y(t)\right)^2 \tag{22}$$

$$\Delta A_j = -\eta \frac{\partial E}{\partial A_j} \tag{23}$$

$$\Delta A_k = -\eta \frac{\partial E}{\partial A_k} \tag{24}$$

Where T is the maximum training time at which error function E approaches very close to zero i.e. for $T \to \infty$, $E \to 0$. η is a fixed positive learning rate. A varying feedback connection weights may produce better results but for simplicity feedback connection weights (B) are considered fixed in the both cases. Therefore, ∂B is constant and hence the learning of the weights does not occur. Since the weights of the feedback connections are considered fixed, the weight update rule for B is

$$\Delta B = -\eta \frac{\partial E}{\partial B} = 0 \tag{25}$$

From the derivations above, the recurrent network can be considered as a feedforward network and by using the weight-update rules in (23)-(25) the network can be trained with standard BP algorithm.

To apply the BP procedure to RNNs, the ordered list of dependencies for the recurrent topologies needs to be adjusted. The present value of the activation of the state denoted as $y(t)$ depends on the previous value $y(t-1)$. The sensitivity of the present state also depends on the previous sensitivities.

5 Experimentation and Analysis of Results

The recurrent network architectures and training procedures described in Section 3 were trained using the BP algorithm to demonstrate the capabilities of the networks. Three well-known benchmark problems were chosen for this task:

(i) Two-input XOR,
(ii) Three-input XOR, and
(iii) Shape eight

The XOR problem has historically been considered a good test of a network model and its associated learning algorithm. There are many reasons for this choice. Firstly, the XOR problem is one of the simplest problems, which is not linearly separable and complex enough for BP algorithm to be trapped in local minima without reaching the global optimum. Secondly, there has been a significant number of research work which claimed that the XOR problem exhibits local minima, a view that is widely accepted in neural network literature [40]-[42]. The configuration of the network and the truth table for the two-input XOR are shown in Figure 4 and Table 2 respectively. Only the symbolic representations of the networks are produced here.

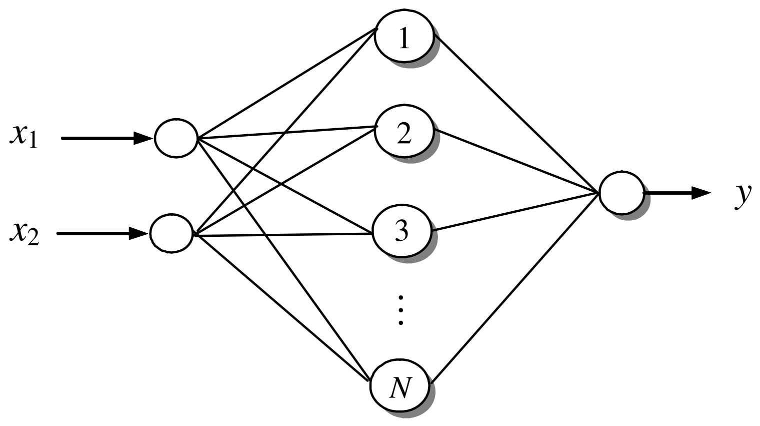


Fig. 4. Neural network for two-input XOR problem

Table 2. Truth-table for two-input XOR

inputs		output
x_1	x_2	y
0	0	0
1	0	1
0	1	1
1	1	0

The parity problem can be considered as a generalised XOR. The generalised XOR problem is good test for computer simulation, because we can systematically vary the mismatch between the degrees of freedom for the problem (number of weights in an unconstrained network). The configurations of the network and truth tables for the three-input generalised XOR is shown in Figure 5 and Table 3.

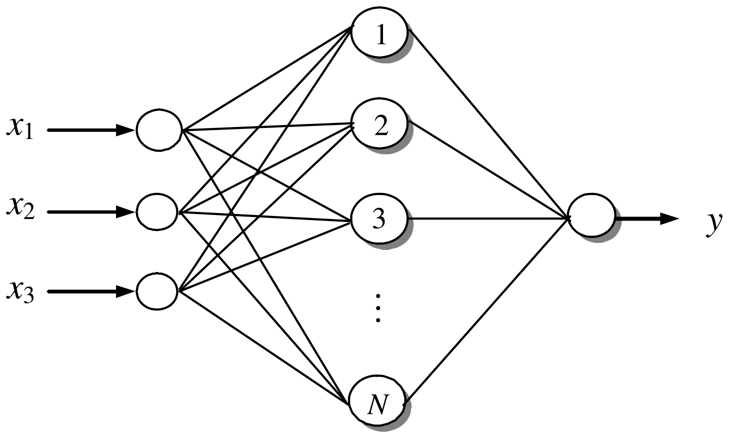


Fig. 5. Neural network for three-input generalised XOR problem

Table 3. Truth-table for three-input generalised XOR

inputs			output
x_1	x_2	x_3	y
0	0	0	0
1	0	0	1
0	1	0	1
1	1	0	0
0	0	1	1
1	0	1	0
0	1	1	0
1	1	1	1

A continuous target trajectory like shape eight is also used as benchmark problem by many researchers [43], [44]. The analytical equations of the corresponding target trajectory of figure eight are given by equations (26)-(27).

$$r1(t) = 0.5 * (1 + 0.9 * \sin(2 * t)) \tag{26}$$

$$r2(t) = 0.5 * (1 + 0.9 * \sin(t)) \tag{27}$$

where $t \in \Re$ and chosen between $[1,8]$. The network configuration and the target trajectory are shown in Figure 6 and Figure 7 respectively.

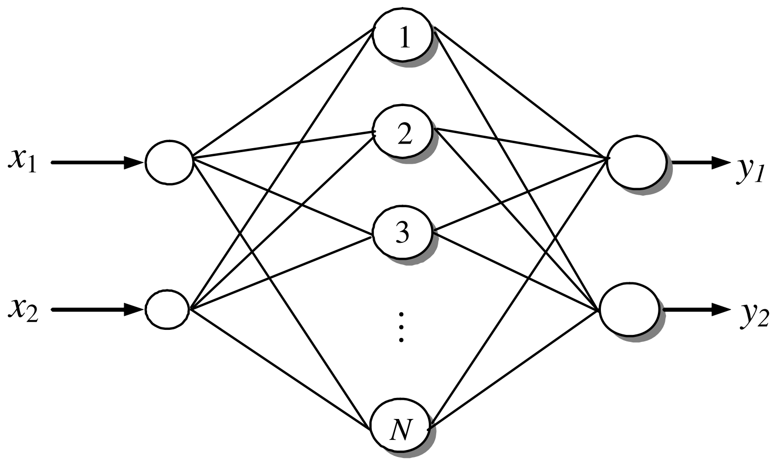


Fig. 6. Neural network for figure eight problem

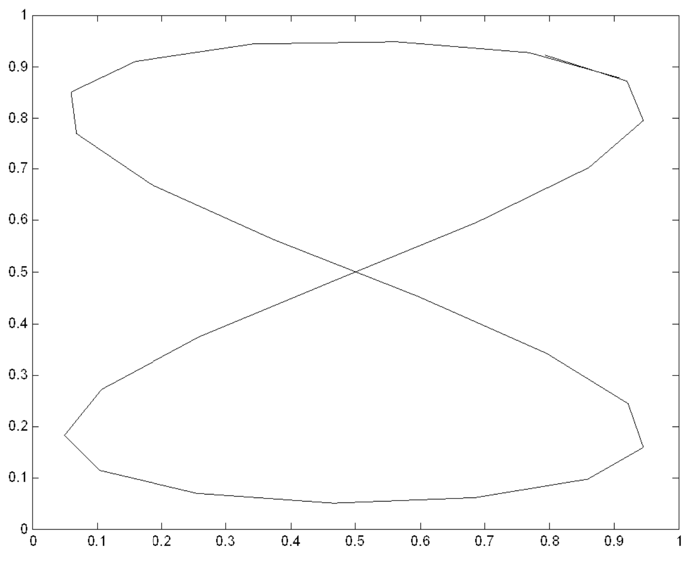


Fig. 7. The shape eight problem

In this study, sigmoidal activation functions were used for the hidden layer and linear activation functions were used for the output layer. Experiments on different architectures with 1 to 18 hidden neurons were investigated on the three benchmark problems. For each problem an error goal of 0.009 and a maximum iteration of 25000 epochs were set. Training parameters such as learning rate, momentum, and acceleration were different for each problem. 11 patterns were chosen for two-input XOR (2XOR) and three-input XOR (3XOR) problem. 27 patterns were used for the figure eight problem, which is found to be the minimum for representing a reasonably smooth curve. For even smoother curve representations, more training time will be required, and so it not considered in this study. The initial weights of the network are sensitive to the convergence of the learning profile and hence found by trial and error for each case of the three benchmark problems.

For the two-input XOR with feedback after non-linearity, a 2-12R-1 RNN achieved an error goal of 0.1955 at 25000 epochs. For the case of feedback before non-linearity, a 2-4R-1 RNN achieved an error goal of 0.0517 at 25000 epochs. For the latter case, full feedback caused instability of the network output and hence a scaling factor was introduced. The RNN was found to be stable for a scaling factor $0 < K_s < 1$ and produced the best result at a value of $K_s = 0.2$. In both cases, learning rate, momentum, and acceleration were kept constant, as shown in Tables 4 and 5. The weight matrices A_j and A_k were initialised within the range (-0.5, +0.5) and yielded better result than if initialised between [0, 1]. The learning profiles for two-input XOR problem are shown in Figure 8. It has been reported by many researchers that the existence of local minima depends on both the problem and the architecture of the network being investigated. Rumelhart et al. found local minima to be very rare [45]. In a study by Hamey, it is shown that a feedforward neural network for the XOR problem has no local minima, which provides a valuable refinement of theories of the occurrence of local minima [46]. It is clearly evident from Figure 8 that the performance of 2-4R-1 RNN with feedback before non-linearity is better than that of 2-12R-1 RNN with feedback after non-linearity. Although feedback before

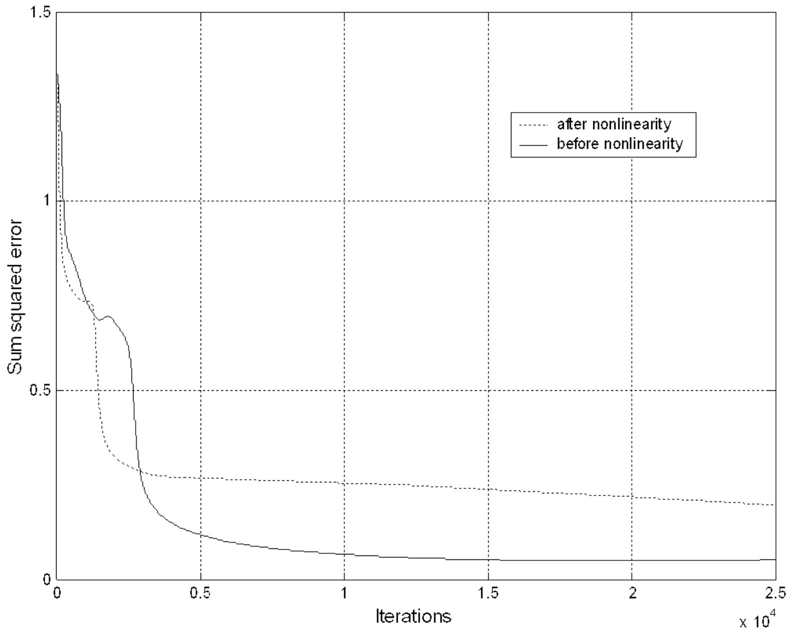


Fig. 8. Learning profile for two-input XOR problem

nonlinearity shows a small local minimum basin at a very early stage of learning, it exhibits a smooth learning curve with a much smaller sum squared error. In the experimentations, we also investigated cases with different odd and even number combinations of hidden neurons. It was found that odd numbers of hidden neurons produced poorer results in the case of two-input XOR.

For the three-input generalised XOR with feedback after non-linearity, a 3-6R-1 RNN achieved an error goal of 0.0089 at 8792 epochs. The weight matrices A_j and A_k were initialised within the range of [-0.5, 0.5]. For the case of feedback before non-linearity, a 3-4R-1 RNN achieved an error goal of 0.00899 at 8080 epochs with a scaling factor of $K_s = 0.2$ In this case, weight matrices A_j and A_k initialised within range of [0, 1] produced better result than within the range of [-0.5, 0.5]. Training parameters were exactly the same in the both cases. The learning profiles for three-input XOR problem are shown in Figure 9. The learning profile for feedback after non-linearity shows two local minima. Training could be easily trapped in those local minima for wrong choice of parameters whereas the learning profile for feedback before non-linearity shows smoother training and much better performance with a smaller architecture with 4 hidden neurons. Investigations further showed that an even number of hidden neurons produce better result than that of odd numbers.

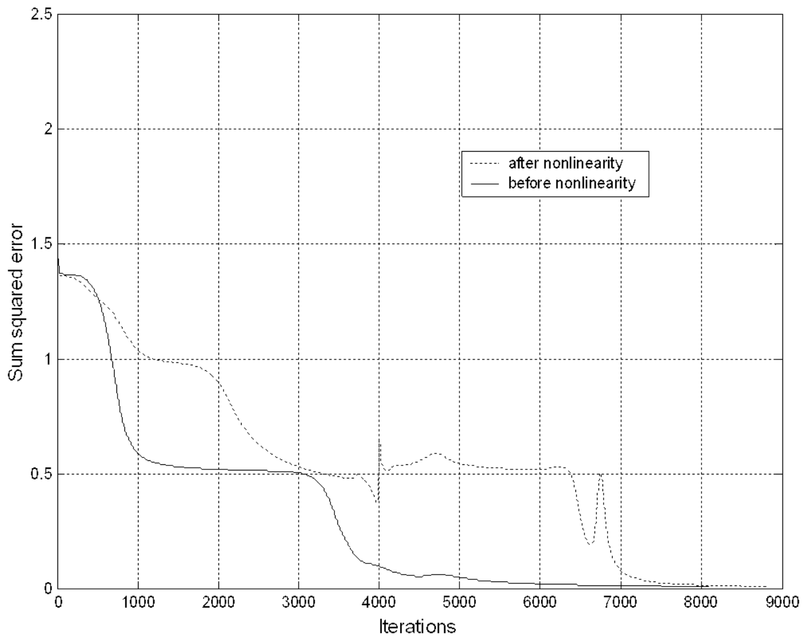


Fig. 9. Learning profile for three-input XOR problem

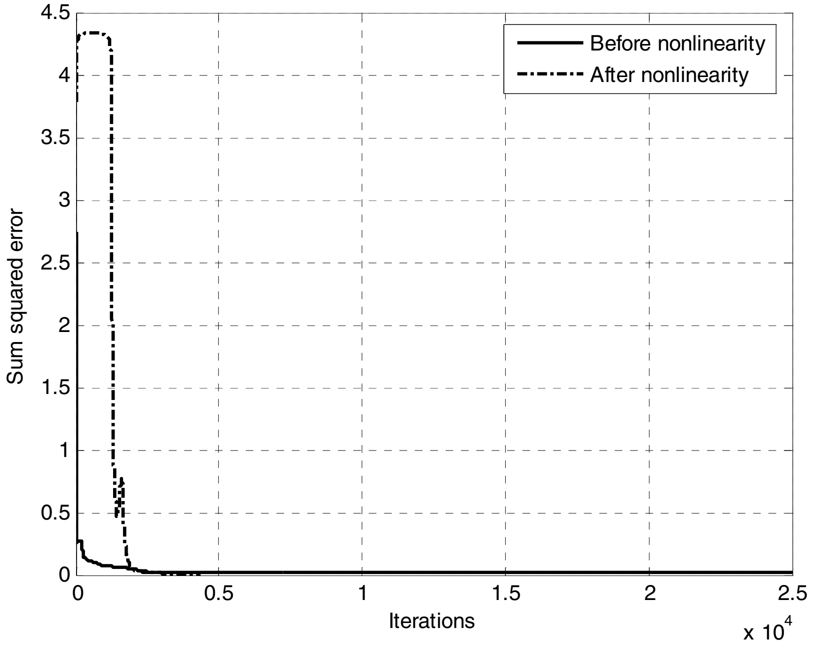


Fig. 10. Learning profile for figure 8 problem

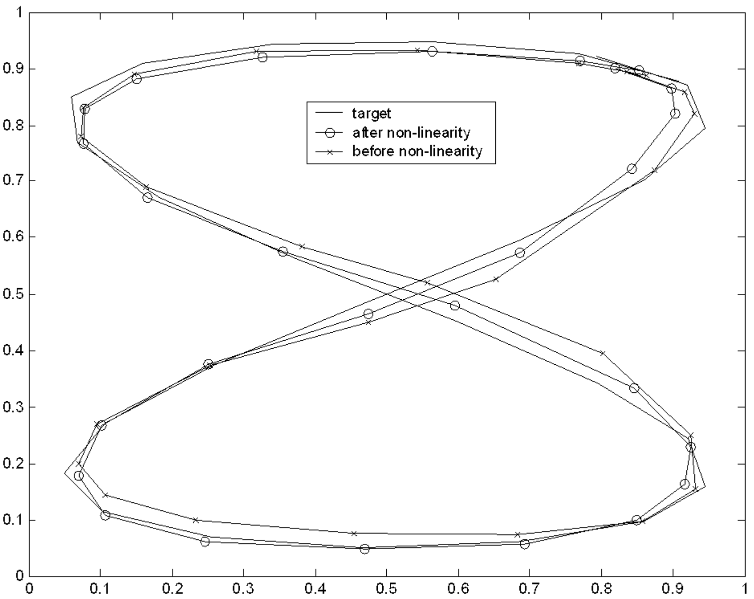


Fig. 11. Trajectory tracking for figure 8

For the figure-eight problem with feedback after non-linearity, a 2-7R-2 RNN achieved an error goal of 0.00899 at 3339 epochs. For the case of feedback before non-linearity, a 2-4R-2 RNN achieved an error goal of 0.01562 at 25000 epochs with a scaling factor of $K_s = 0.2$. Training parameters were kept same in the both cases. The weight matrices A_j and A_k were initialised within the range of [-1, 1]. The learning profiles for the shape-eight problem are shown in Figure 10. The learning profile for feedback after non-linearity shows local minima. Training could be easily trapped for incorrect choice of parameters where as the learning profile for feedback before non-linearity shows smoother training and much better performance with a smaller number of hidden neurons, i.e. 4 hidden neurons. Investigations show that odd number of hidden neurons produced better results in the case of feedback after non-linearity and even number produced better result in the case of feedback before non-linearity. It is found that such architectures of RNN can follow a continuous trajectory as shown in Figure 11.

A summary of network training parameters used for the three benchmark problems are shown in Table 4 and Table 5.

Table 4. Feedback after non-linearity

	Learning rate	Momentum	Acceleration	Hidden neuron	K_s	Epochs	Error
2XOR	.04	.8	-0.1	12	1	25000	.1955
3XOR	.04	.8	-0.1	6	1	8792	.0089
Fig8	1.2	.7	-0.7	7	1	3339	.0089

Table 5. Feedback before non-linearity

	Learning rate	Momentum	Acceleration	Hidden neuron	K_s	Epochs	Error
2XOR	.04	.8	-0.1	4	.2	25000	.0517
3XOR	.04	.8	-0.1	4	.2	8080	.0089
Fig8	1.2	.7	-0.7	4	.2	25000	.0156

As can be seen from Table 5 that feedback connection before non-linearity is more stable than that of feedback connection after non-linearity, require less hidden layer neurons, and achieve better error goal within fewer epochs.

6 Conclusion

The adaptive capacity of the RNN was investigated on two different neural architectures. The three benchmark problems have been used to investigate the performance of the proposed RNN architectures. Improved performances have been demonstrated by RNN architectures with feedback connections before the application of the non-linear function when compared with RNN architectures with feedback connection applied after the non-linear function. The experiments have shown that the RNN can be trained with a reduced risk of the stability, which is mainly caused by recurrent connection after non-linearity. The results from this investigation will thus be useful in designing future applications that incorporates RNNs.

References

[1] Parlos, A.G., Parthasarathy, S., Atiya, A.F.: Neuro-Predictive Process Control using On-line Controller Adaptation. IEEE Transaction on Control Systems Technology 9(5), 741–755 (2001)

[2] Narendra, K.S., Parthsarathy, K.: Identification and control of dynamical systems using Neural Network. IEEE Transaction on Neural Networks 1(1), 4–27 (1993)

[3] Narendra, K.S., Mukhopadhyay, S.: Adaptive Control using Neural Networks and Approximate Models. IEEE Transactions on Neural Networks 8, 475–485 (1997)

[4] Powell, M.J.D.: Radial Basis Functions for Multivariable Interpolation: A review. IMA Conference Algorithm for Approximation of Functions and Data, RMCS Shrivenham (1985)

[5] Specht, D.F.: Probabilistic Neural Networks. Neural Networks, International Neural Network Society 3, 109–118 (1990)

[6] Specht, D.F.: A Generalized Regression Neural Network. IEEE Transactions on Neural Networks 2(6), 568–576 (1991)

[7] Millan, J.R., Mourino, J., Franze, M., Cincotti, F., Varsta, M., Heikkonen, J., Babiloni, F.: A Local Neural Classifier for the Recognition of EEG Patterns Associated to Mental Tasks. IEEE Transaction on Neural Networks 13(3), 678–686 (2002)

[8] Feldkamp, L.A., Prokhorov, D.V., Feldkamp, T.M.: Simple and Conditioned Adaptive Behaviour from Kalman Filter Trained Recurrent Networks. Neural Networks 16, 683–689 (2003)

[9] Elman, J.L.: Finding Structure in Time. Cognitive Science 14, 179–211 (1990)

[10] Frasconi, P., Gori, M., Soda, G.: Local feedback multilayered networks. Neural Computing 4, 120–130 (1992)

[11] Patan, K.: Stability Analysis and the Stabilization of a Class of Discrete-Time Dynamic Neural Networks. IEEE Transaction on Neural Networks 18(3), 660–673 (2007)

[12] Spiegel, R., Suret, M., Le Pelley, M.E., McLaren, I.P.L.: Analysing State Dynamics in a Recurrent Neural Network. In: IEEE International Conference on Neural Networks, pp. 834–839 (2002)

[13] Tsoi, A.C., Back, A.D.: Locally Recurrent Globally Feedforward Networks: A Critical Review of Architectures. IEEE Transaction on Neural Networks 5(2), 229–239 (1994)

[14] Williams, R.J., Zipser, D.: A Learning Algorithm for Continually Running Fully Recurrent Networks. Neural Computing 1, 270–280 (1989)

[15] De Jesus, O., Horn, J.M., Hagan, M.T.: Analysis of Recurrent Network Training and Suggestions for Improvements. In: IEEE International Joint Conference on Neural Networks, pp. 2632–2637 (2001)

[16] Towntey, S., Ilchmann, A., Weib, M.G., Mcclements, W., Ruiz, A.C., Owens, D.H., Pratzel-Wolters, D.: Existence and Learning of Oscillation in Recurrent Neural Networks. IEEE Transactions on Neural Networks 11(1), 205–214 (2000)

[17] Suykens, J.A.K., De Moor, B., Vandewalle, J.: Robust Local Stability of Multilayer Recurrent Neural Networks. IEEE Transactions on Neural Networks 11(1), 222–229 (2000)

[18] Xia, Y., Wang, J.: Global Exponential Stability of Recurrent Networks for Solving Optimisation and Related Problems. IEEE Transactions on Neural Networks 11(4), 1017–1022 (2000)
[19] Bianchini, M., Gori, M., Maggini, M.: On the Problem of Local Minima in Recurrent Neural Networks. IEEE Transactions on Neural Networks 5(2), 167–177 (1994)
[20] Feldkamp, L., Puskorius, G., Moore, P.: Adaptation from Fixed Weight Dynamic Networks. In: Proceedings of IEEE International Conference on Neural Networks, pp. 155–160 (1996)
[21] Younger, A.S., Conwell, P.R., Cotter, N.E.: Fixed-Weight on-line Learning. IEEE Transaction on Neural Networks 10(2), 272–283 (1999)
[22] Prokhorov, D.V., Feldkamp, L.A., Tyukin, I.Y.: Adaptive Bahaviour with Fixed Weights in RNN: An Overview. In: IEEE International Joint Conference on Neural Networks, pp. 2018–2022 (2002)
[23] Younger, A.S., Hochreiter, S., Conwell, P.R.: Meta Learning with Backpropagation. In: IEEE International Conference on Neural Networks, pp. 2001–2006 (2001)
[24] Lo, J.T., Bassu, D.: Adaptive vs. Accommodative Neural Networks for Adaptive System Identification. In: IEEE International Joint Conference on Neural Networks, pp. 1279–1284 (2001)
[25] Bengio, Y., Simard, P., Frasconi, P.: Learning Long-Term Dependencies with Gradient Descent is Difficult. IEEE Transactions on Neural Networks 5(2), 157–166 (1994)
[26] Atiya, A.F., Parlos, A.G.: New Result on Recurrent Network Training: Unifying the Algorithms and Accelerating Convergence. IEEE Transactions on Neural Networks 11(3), 697–709 (2000)
[27] Campolucci, P., Uncini, A., Piazza, F., Rao, B.D.: On-Line Learning Algorithms for Locally Recurrent Neural Networks. IEEE Transactions on Neural Networks 10(2), 253–271 (1999)
[28] Leistritz, L., Galicki, M., Witte, H., Kochs, E.: InitialState Training Procedure Improves Dynamic Recurrent Networks with Time-Dependent Weights. IEEE Transaction on Neural Networks 12(6), 1513–1518 (2001)
[29] Back, A., Tsoi, A.: FIR and IIR synapses, a new neural network architecture for time series modelling. Neural Computation 3(3), 375–385 (1991)
[30] Billings, S.A., Jamaluddin, H.B., Chen, S.: Properties of neural networks with applications to modelling non-linear dynamical systems. International Journal of Control 55(1), 193–224 (1992)
[31] Arai, K., Nakano, R.: Stable Behaviour in a Recurrent Neural Network for a FiniteState Machine. Neural Networks 13, 667–680 (2000)
[32] Jordan, M.I.: Attractor dynamics and parallelism in a connectionist sequential machines. In: Proceedings of the 8th Conference of the Cognitive Science Society, pp. 531–546. Erlbaum (1986)
[33] Poddar, P., Unnikrishnan, K.P.: Non-linear prediction of speech signals using memory neuron networks. In: Juang, B.H., Kung, S.Y., Kammedts, C.A. (eds.) Proceedings of the 1991 IEEE Workshop on Neural Networks for Signal Processing, pp. 1–10 (1991)
[34] Gers, F.A., Schmidhuber, J.: LSTM Recurrent Networks Learn Simple Context-Free and Context-Sensitive Languages. IEEE Transactions on Neural Networks 12(6), 1333–1340 (2001)
[35] De Vries, B., Principe, J.C.: The gamma model – A new neural network for temporal processing. Neural Networks 5, 565–576 (1992)
[36] Chalup, S.K., Blair, A.D.: Incremental Training of First Order Recurrent Neural Networks to Predict a Context-Sensitive Language. Neural Networks 16, 955–972 (2003)
[37] Lin, T., Horne, B.G., Giles, C.L.: How embedded memory in recurrent neural network architectures helps learning long-term temporal dependencies. Neural Networks 11, 861–868 (1998)

[38] Hopfield, J.: Neural networks and physical systems with emergent collective computational abilities. Proceedings National Academy of Sciences, USA 79, 2554–2558 (1982)

[39] Santini, S., Del Bimbo, A., Jain, R.: Block-Structured Recurrent Neural Networks. Neural Networks 8(1), 135–147 (1995)

[40] Cetin, B.C., Burdick, J.W., Barhen, J.: Global descent replaces gradient descent to avoid local minima problem in learning with artificial neural networks. In: Proceedings of of the IEEE International Conference on Neural Networks, Piscataway, NJ, USA, vol. 2, pp. 836–842 (1993)

[41] Dayhoff, J.E.: The exclusive-OR: A classic problem. In: Neural Network Architectures: An Introduction, pp. 76–79. Van Nostrand Reinhold, New York (1990)

[42] Gori, M., Tesi, A.: On the problem of local minima in backpropagation. IEEE Transactions on Pattern Analysis and Machine Intelligence 14, 76–85 (1992)

[43] Galiki, M., Leistritz, L., Witte, H.: Learning Continuous Trajectories in Recurrent Neural Networks with Time-Dependent Weights. IEEE Transaction on Neural Networks 10(4), 741–756 (1999)

[44] Leistritz, L., Galicki, M., Witte, H., Kochs, E.: Training Trajectories by Continuous Recurrent Multilayer Networks. In: IEEE International Joint Conference on Neural Networks, pp. 283–291 (2002)

[45] Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning internal representations by error propagation. In: Parallel Distributed Processing, pp. 318–362. MIT Press, Cambridge (1986)

[46] Hamey, L.G.C.: XOR has no local minima: A case study in neural network error surface analysis. Neural Networks 11, 669–681 (1998)

Chapter 9
An Extended Approach of a Two-Stage Evolutionary Algorithm in Artificial Neural Networks for Multiclassification Tasks

Antonio J. Tallón-Ballesteros[1], César Hervás-Martínez[2], and Pedro A. Gutiérrez[2]

[1] Department of Languages and Computer Systems,
University of Seville,
Reina Mercedes Avenue, Seville, 41012 Spain
[2] Department of Computer Science and Numerical Analysis,
University of Córdoba,
Campus of Rabanales, Albert Einstein Building, Córdoba, 14071 Spain
atallon@us.es

Abstract. This chapter considers a recent algorithm to add broader diversity at the beginning of the evolutionary process and extends it to sigmoidal neural networks. A simultaneous evolution of architectures and weights is performed with a two-stage evolutionary algorithm. The methodology operates with two initial populations, each one containing individuals with different topologies which are evolved for a small number of generations, selecting the half best individuals from each population and combining them to constitute a single population. At this point, the whole evolutionary cycle is applied to the new population. This idea was previously proposed by us for product unit neural networks, and we now extend to sigmoidal neural networks. The experimentation has been carried out on twelve data sets from the UCI repository and two complex real-world problems which differ in their number of instances, features and classes. The results have been contrasted with nonparametric statistical tests and show that our proposal significantly improves the test accuracy of the models with respect to the obtained ones with a standard methodology based on a single population. Moreover, the new proposal is much more efficient than other methods developed previously by us.

Keywords: Artificial neural networks, Sigmoidal units, Product units, Evolutionary algorithms, Classification, Population diversity.

1 Introduction

The diversity issue is very important to avoid a premature convergence of evolutionary algorithms (EAs) and to reach solutions with good quality. At the beginning of the algorithm is suitable a diverse population and a more condensed one at the end of the search [1]. In this way, the algorithm combines the two key ideas exploration and exploitation. The matter of generating various populations is related with the diversity throughout the evolutionary process and has been also discussed previously (for instance in [2]). The number of populations may vary and there is not a common accepted value. A common view of the evolutionary cycle is that diversity enhances the performance of a population by providing more chances for evolution. A homogeneous population offers no advantage for improvement as the entire population is focused in a particular portion of the search space. High diversity does not imply better genetic algorithm performance; this is closely related to the question of exploration versus exploitation, but enforcing diversity in the early phases of evolution ensures a broad exploration of the search space [3].

Briefly, the main approach considered in this chapter diversifies the architecture of the neural network (NN) for classification problems at the beginning of the evolutionary process. Our previous methodology [4], which operates with evolutionary artificial neural networks (EANNs) based on product units [5] in a two-stage EA (TSEA) is extended in the current chapter in order to consider EANNs with sigmoidal units. Our objective is to compare the improved methodology -based on two stages- to the standard one -composed of a single stage with a population- employing sigmoidal units in both cases and also to the previous TSEA methodology that have been employed to date by us for product unit neural networks (PUNNs). A computational cost analysis is performed to determine the efficiency of the standard and improved methodologies that have been applied first time both for sigmoidal neural networks in the current chapter. Also, an efficiency report between product and sigmoidal units applying the improved methodology is presented.

The chapter is organized as follows: Sect. 2 describes some concepts about evolutionary sigmoidal and product unit neural networks, the base EA and TSEA; Sect. 3 introduces our proposal; Sect. 4 details the experimentation; then Sect. 5 shows and analyzes the results obtained; finally, Sect. 6 states the concluding remarks.

2 Methodology

2.1 Evolutionary Artificial Neural Networks Based on Sigmoidal and Product Units

EANNs offer a platform to optimize network performance and architecture simultaneously. Miller et al. [6] proposed that evolutionary computation was a very good candidate to search the space of architectures. Since then, many methods have been developed to evolve artificial neural networks (ANNs), for instance, [7, 8]. Next, we describe EANNs based on sigmoidal and product units. The methodology employed here uses an EA as a tool for simultaneous learning the architecture and weights of the ANN. Other authors have also considered this idea [9, 10]. In previous works we have trained PUNNs [4, 11], but in the current chapter we experiment, moreover, with sigmoidal units.

Fig. 1 shows the scheme of two ANN models for a bi-classification problem with sigmoidal (at part a)) and product units (at part b)). Each one is a *k:m:1* three-layer architecture, that is, *k* nodes in the input layer, *m* ones and a bias one in the hidden layer and one node in the output layer.

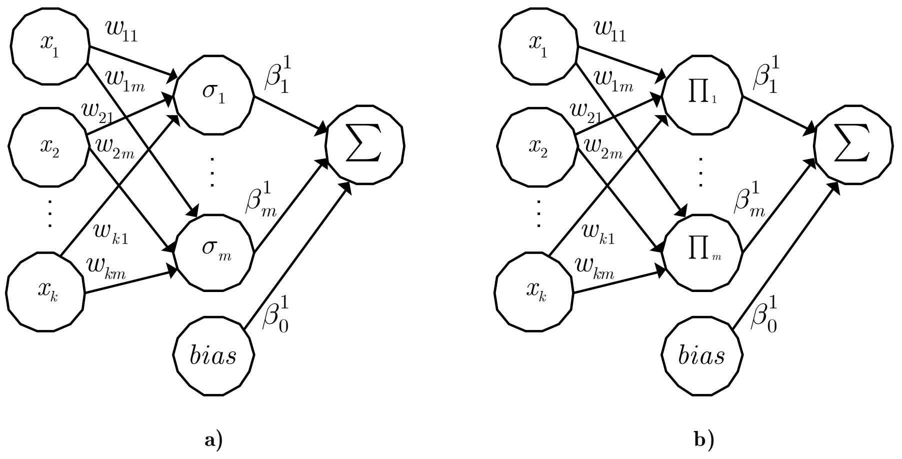


Fig. 1. Scheme of two ANN models with sigmoidal (a) and product (b) units for a bi-classification problem

The transfer function of each node in the hidden and output layers is the identity function. Thus, the functional model obtained by each of the nodes in the output layer with *J* classes is given by Eq. 1.

$$f(x_1, x_2, ..., x_k) = \beta_0^l + \sum_{j=1}^{m} \beta_j^l B_j(\boldsymbol{x}, w_j) \qquad l = 1, 2, ..., J \tag{1}$$

where B_j follows different expressions depending on the type of unit:

- Product unit:

$$B_j(\boldsymbol{x}, w_j) = \prod_{i=1}^{k} x_i^{w_{ji}} \tag{2}$$

- Sigmoidal unit:

$$B_j(\boldsymbol{x}, w_j) = \frac{1}{1 + \exp(-w_{j0} - \sum_{i=1}^{k} w_{ji} x_i)} \tag{3}$$

2.2 Two-Stage Evolutionary Algorithm

We have used a base EA to design the structure and learn the weights of ANN as in [11]. The followed base algorithm in the current chapter, that has also several common points with the EA described in [12], is exactly equal to those described in our previous work [4] for classification tasks. However, it has been generalized to any type of ANNs. We have followed the same guidelines given in our previous work; a detailed explanation can be read in Sect. 2.2 of [4].

```
Program: Two-Stage Evolutionary Algorithm
Data: Training set
Input parameters: gen, neu
Output: Best ANN model
1:                                  // **First Stage**
2:  $t \leftarrow 0$
3:                                  // Population $P_1$
4:  $P_1(t) \leftarrow \{ind_1, \ldots, ind_{10000}\}$   // Individuals of $P_1$ have *neu* nodes in the hidden layer
5:  $f_1 (P_1(t) \{ind_1, \ldots, ind_{10000}\}) \leftarrow$ *fitness* $(P_1(t) \{ind_1, \ldots, ind_{10000}\})$ // Calculate fitness
6:  $P_1(t) \leftarrow P_1(t) \{ind_1, \ldots, ind_{10000}\}$   // Sort individuals
7:  $P_1(t) \leftarrow P_1(t) \{ind_1, \ldots ind_{1000}\}$   // Retain the 1000 best ones
8:                                  // Population $P_2$
9:  $P_2(t) \leftarrow \{ind_1, \ldots, ind_{10000}\}$   // Individuals of $P_2$ have *neu+1* nodes in the hidden layer
10: $f_2 (P_2(t) \{ind_1, \ldots, ind_{10000}\}) \leftarrow$ *fitness* $(P_2(t) \{ind_1, \ldots, ind_{10000}\})$ // Calculate fitness
11: $P_2(t) \leftarrow P_2(t) \{ind_1, \ldots, ind_{10000}\}$   // Sort individuals
12: $P_2(t) \leftarrow P_2(t) \{ind_1, \ldots ind_{1000}\}$   // Retain the 1000 best ones
13: **for** each $P_i$   // Evolution of populations $P_1$ and $P_2$ until 0.1**gen* generations
14:   current_generation $\leftarrow 0$
15:   $t \leftarrow 0$
16:     **while** current_generation < 0.1*gen not met **do**
17:        $P_i(t) \{ind_{901}, \ldots ind_{1000}\} \leftarrow P_i(t) \{ind_1, \ldots, ind_{100}\}$ // Best 10% replace the worst 10%
18:        $P_i(t+1) \leftarrow P_i(t) \{ind_1, \ldots, ind_{900}\}$
19:        $P_i(t+1) \leftarrow$ *pm* $(P_i(t+1) \{ind_1, \ldots, ind_{90}\})$   // Parametric mutation (10% $P_i (t+1)$)
20:        $P_i(t+1) \leftarrow$ *sm* $(P_i(t+1) \{ind_{91}, \ldots, ind_{900}\})$   // Structural mutation (90% $P_i (t+1)$)
21:        $f_i(P_i(t+1) \{ind_1, .. ind_{900}\}) \leftarrow$ *fitness* $(P_i(t+1) \{ind_1, \ldots, ind_{900}\})$   // Evaluate
22:        $P_i(t+1) \leftarrow P_i(t+1) (ind_1, \ldots, ind_{900}) \bigcup P_i(t)\{ind_{901}, \ldots, ind_{1000}\}$
23:        $P_i(t+1) \leftarrow P_i(t+1)\{ind_1, \ldots, ind_{1000}\}$   // Sort individuals
24:        current_generation $\leftarrow$ current_generation + 1
25:        $t \leftarrow t + 1$
26:     **end while**
27: **end for**
28: $P(0) \leftarrow P_1\{ind_1, \ldots, ind_{500}\} \bigcup P_2\{ind_1, \ldots, ind_{500}\}$   // Individuals of P has [neu, neu+1]
29:                                                   // nodes in the hidden layer
30: $P(0) \leftarrow P(0) \{ind_1, \ldots, ind_{1000}\}$   // Sort individuals by fitness: $ind_i > ind_{i+1}$
31:                                  // **Second Stage**
32:                                  // Input: gen, neu+1
33:   $t \leftarrow 0$
34:   **while** stop criterion not met **do**   // main loop
35:       $P(t) \{ind_{901}, \ldots ind_{1000}\} \leftarrow P(t) \{ind_1, \ldots, ind_{100}\}$ // Best 10% replace the worst 10%
36:       $P(t+1) \leftarrow P(t) \{ind_1, \ldots, ind_{900}\}$
37:       $P(t+1) \leftarrow$ *pm* $(P(t+1) \{ind_1, \ldots, ind_{90}\})$   // Parametric mutation (10% $P(t+1)$)
38:       $P(t+1) \leftarrow$ *sm* $(P(t+1) \{ind_{91}, \ldots, ind_{900}\})$   // Structural mutation (90% $P(t+1)$)
39:       $f(P(t+1) \{ind_1, .. ind_{900}\}) \leftarrow$ *fitness* $(P(t+1) \{ind_1, \ldots, ind_{900}\})$   // Evaluate
40:       $P(t+1) \leftarrow P(t+1) (ind_1, \ldots, ind_{900}) \bigcup P(t) \{ind_{901}, \ldots, ind_{1000}\}$
41:       $P(t+1) \leftarrow P(t+1)\{ind_1, \ldots, ind_{1000}\}$   // Sort individuals
42:       last_generation $\leftarrow t$
43:       $t \leftarrow t+1$
44:  **end while**
45:  **return** *best* (P(last_generation) $\{ind_1\}$)
```

Fig. 2. Pseudo-code of the TSEA for classification

A specialization of the previous algorithm called TSEA has been introduced in [4] for product units. Basically, the two-stage EA operates with two initial populations, each of one containing individuals with different topologies, evolving them for a small number of generations, selecting the best individuals from each population in the same proportion and combining them to constitute a single population. At this point, the whole evolutionary cycle is applied to the new population. The pseudo-code of the TSEA appears in Fig. 2. The individuals are subjected to the operations of replication and mutation, thus the algorithm falls into the class of evolutionary programming. We do not use the crossover operator due to the permutation problem [7].

3 Proposal Description

The current chapter presents an extended methodology called TSEASig (Two-Stage Evolutionary Algorithm for neural networks with *Sig*moidal units) that is derived from TSEA [4]. The number of neurons in the input layer is equal to the number of variables in the problem; a hidden layer with a number of nodes that depends on the data set to be classified; and the number of nodes in the output layer equal to the number of classes minus one because a softmax-type probabilistic approach has been used. The first stage consists of creating two populations, each one with individuals that present different maximum number of nodes, *neu* and *neu+1*, in the hidden layer, evolving them with equal settings of the remaining parameters of the EA for a small number of generations, *0.1*gen*, selecting the half best individuals from each population and unifying them to constitute a single population. In the second stage, the full evolutionary process is applied to the population. The initial short training improves random individuals and let to explore possible promising areas in two directions, since there are two different populations. After that, individuals with different topologies coexist and the more adapted ones will remain. The parameters are defined as *gen,* the maximum number of generations; and *neu,* the maximum number of nodes in the hidden layer. In the recently cited work, we proposed a TSEA for EANNs with product units. TSEASig is an extension of TSEA that operates with NNs based on sigmoidal units. Fig. 3 depicts the structure of TSEASig.

EDD (experimental design distribution) methodology has been launched by us in [13]. It distributes some parameters of the network topology or of the base EA over some computing nodes. An initial configuration, called base configuration, is defined and it is modified with a new value for one parameter in each of the nodes. The idea is to run the base EA with different configurations. Up to date, EDD has been applied with NNs based on product units. Now, we try out it first time with sigmoidal units and the resulting methodology is named EDDSig, another original contribution of this chapter.

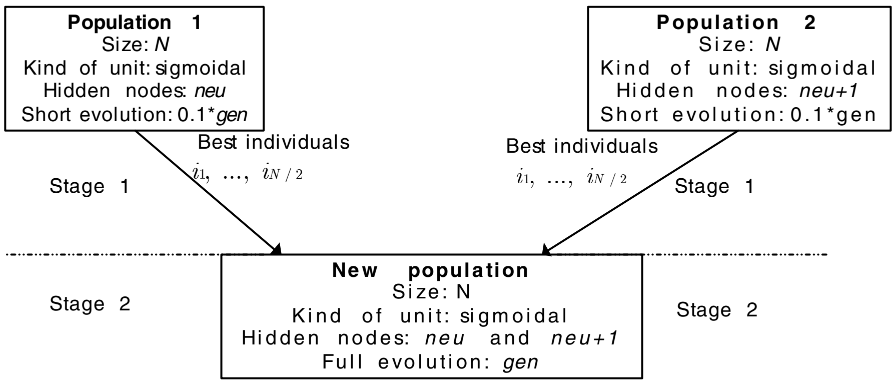


Fig. 3. TSEASig structure

Table 1 presents the description of the TSEA, EDDSig and TSEASig configurations. The *neu* and *gen* and parameters take the values indicated as input to the program. α_2 is associated with the residual of the updating expression of the output-layer weights and the initial value is fixed. For EDDSig, we have considered two configurations that can be compared to the equivalent one of TSEASig. Related to TSEA, we have taken into account the configuration 1*. It is noticeable that TSEA works with PUNNs and the remaining methodologies with sigmoidal units.

Table 1. Description of the TSEA, EDDSig and TSEASig configurations

Methodology	Configuration	Num. of neurons	Num. of neurons in each population	Max. Num. of generations	Max. Num. of generations in each population	α_2
TSEA	1*	-	neu and neu+1	-	0.1*gen	1
EDDSig	1S	neu	-	gen	-	0.5
EDDSig	2S	neu+1	-	gen	-	0.5
TSEASig	1S*	-	neu and neu+1	-	0.1*gen	0.5

4 Experimentation

4.1 Data Sets and Validation Technique

Table 2 summarizes the data sets employed. Most of them are publicly available at the UCI repository [14] and the last two concern complex real-world problems. The following fourteen ones have been used: Statlog (*Australian* credit approval), *Balance* scale, breast *Cancer* Wisconsin, *Heart* disease (Cleveland), *HeartY, Hepatitis*, *Horse* colic, Thyroid disease (allhypo, *Hypothyroid*), Thyroid disease (*Newthyroid*), *Pima*

Indians diabetes, *Waveform* database generator (version 2) and *Yeast* regarding the UCI data sets, and *BTX* and *Listeria* monocytogenes as complex real-world problems. BTX is a multi-class classification environment problem for different types of drinking waters [15]. *Listeria* monocytogenes is a bi-class problem in predictive microbiology [16].

All nominal variables have been converted to binary ones. Also, the missing values have been replaced in the case of nominal variables by the mode or, when concerning continuous variables, by the mean, considering the full data set. The experimental design uses the cross validation technique called stratified hold-out that consists of splitting the data into two sets: training and test set, maintaining the class distribution of the samples in each set approximately equal as in the original data set. Their sizes are approximately *3N/4* and *N/4*, being *N* the number of patterns in the problem. The proportions do not match in BTX [15] and Listeria [17] because the data is prearranged in two sets due to their specific features.

Table 2. Summary of the data sets used

Data set	Total Patterns	Training Patterns	Test Patterns	Features	Inputs	Classes
Australian	690	517	173	14	51	2
Balance	625	469	156	4	4	3
Cancer	699	525	174	10	9	2
Heart	303	227	76	13	26	2
HeartY	270	202	68	13	13	2
Hepatitis	155	117	38	19	19	2
Horse	368	276	92	27	83	2
Hypothyroid	3772	2829	943	29	29	4
Newthyroid	215	161	54	5	5	3
Pima	768	576	192	8	8	2
Waveform	5000	3750	1250	40	40	3
Yeast	1484	1112	372	8	8	10
BTX	63	42	21	3	3	7
Listeria	539	305	234	4	4	2

4.2 Common Parameters of the Different Methodologies and Specific Parameters Depending on the Dataset

Table 3 introduces the common parameters for all datasets related to TSEASig, EDDSig and TSEA methodologies. All values are according with our previous works [4, 18]. In the case of TSEA, since PUNNs are used, data is normalized to avoid input values near to 0, which can produce very large values of the outputs for negative exponents.

Table 3. Common parameters for TSEASig, EDDSig and TSEA

Parameter/Feature	**Value**	
	TSEASig/EDDSig	TSEA
Population size (N)	1000	1000
gen-without-improving	20	20
Interval for the terms/exponents w_{ji}	[-5, 5]	[-5, 5]
Interval for the coefficients β_j^l	[-10, 10]	[-5, 5]
Initial value of α_1	0.5	0.5
Initial value of α_2	0.5	1
Normalization of the input data	[0.1, 0.9]	[1, 2]
Number of nodes in node addition and node deletion operators	[1, 2]	[1, 2]

The values of the *neu* and *gen* parameters depend on the data set and are shown in Table 4. The decision about the number of neurons is a very difficult task in the scope of NNs, but determining the optimal values is a challenge. With respect to the number of generations, we have defined three kinds of values: small (100-150), medium (300), large (500) and very large (1000). Again, the optimal number is unknown; however the algorithm has a stop criterion to avoid evolving up to the maximum number of generations if there is no improvement. We have given values of our choice to the two parameters depending on the complexity of the data set (number of

Table 4. Values of TSEA/EDDSig/TSEASig parameters depending on the data set

Data set	**TSEA**		**EDDSig**		**TSEASig**	
	Num. of neurons in each population (*neu* and *neu+1*)	**Max. Num. of generations in each population**	**Num. of neurons (*neu*)**	**Max. Num. of generations (*gen*)**	**Num. of neurons in each population (*neu* and *neu*+1)**	**Max. Num. of generations in each population**
Australian	4 and 5	100	4	100	4 and 5	100
Balance	5 and 6	150	4	300	4 and 5	300
Cancer	2 and 3	100	2	100	2 and 3	100
Heart	3 and 4	300	3	300	3 and 4	300
HeartY	4 and 5	100	4	100	4 and 5	100
Hepatitis	3 and 4	100	3	100	3 and 4	100
Horse	4 and 5	300	4	300	4 and 5	300
Hypothyroid	3 and 4	500	3	500	3 and 4	500
Newthyroid	3 and 4	300	3	300	3 and 4	300
Pima	3 and 4	120	3	100	3 and 4	100
Waveform	3 and 4	500	3	500	3 and 4	500
Yeast	11 and 12	1000	11	1000	11 and 12	1000
BTX	5 and 6	500	5	500	5 and 6	500
Listeria	4 and 5	300	4	300	4 and 5	300

classes, inputs, instances,…). Other times the values are based on previous works [4, 18]. EDDSig and TSEASig values are in concordance to compare the performance of both methodologies. Sometimes, the values differ between methodologies. The initial tests with sigmoidal units for Balance dataset sheds light on that a high number of neurons provokes overfitting.

4.3 Nonparametric Statistical Analysis

We follow the recommendations pointed out by J. Demšar [19] to perform nonparametric statistical tests to determine the statistical significance of the differences in rank observed for each method with all data sets. There are two methods, Friedman and Iman-Davenport tests. The former test is based in χ_F^2 statistic; the null hypothesis states that all algorithms perform equal. The latter test is based of F_F which is a better statistic, derived from χ_F^2. F_F is distributed according to the F-distribution with $(k-1)$ and $(k-1)(N-1)$ degrees of freedom with k algorithms and N datasets. If the null-hypothesis is rejected, we can proceed with a post-hoc test. Nemenyi test has been performed to compare all classifiers to each other. The critical difference (CD) can be computed from critical values, k and N. The considered significance levels have been 0.05 for Iman-Davenport test, and 0.05 and 0.10 for the post-hoc method.

5 Results

First of all, this section presents the results obtained related to the Correct Classification Ratio (CCR) in the test set with TSEA, EDDSig and TSEASig methodologies. After that, a nonparametric statistical analysis compares all of them. Next, an analysis of the computational cost is performed. Finally, we report a summary of the results obtained with a good number of classifiers, from the scope of NNs or classical/modern machine learning.

5.1 Results Applying TSEA, EDDSig and TSEASig

The results obtained by applying TSEA [4] are presented, along with those obtained with EDDSig and TSEASig. In the case of EDDSig two configurations have been considered (1S and 2S). In TSEA, there were two configurations 1* and 2*; however, only one (1*) is considered to make a fair comparison with the remaining methodologies. In TSEASig, the single configuration is 1S*. EDDSig configurations are equivalent to 1S*.

Table 5 shows the mean and standard deviation (SD) of the CCR and the topology for each data set and configuration for a total of 30 runs or iterations. By rows, the best result appears in boldface. The value obtained with TSEASig is in italics if it is better than the two values related to EDDSig. The descriptive analysis of the data reveals that the TSEA methodology obtains best results for eight data sets and TSEASig six times. Usually, TSEASig has lower SD than EDDSig and it expresses more homogeneous results of the former methodology.

Table 5. Results obtained in fourteen data sets with the different configurations related to TSEA, EDDSig and TSEASig methodologies

Data set	Methodologies							
	TSEA		EDDSig				TSEASig	
	1*		1S		2S		1S*	
	Mean±SD	Top.	Mean±SD	Top.	Mean±SD	Top.	Mean±SD	Top.
Australian	**88.11±1.56**	51:[4,5]:1	87.12±1.45	51:4:1	87.46±1.18	51:5:1	86.55±1.43	51:[4,5]:1
Balance	**96.20±1.06**	4:[5,6]:2	94.14±1.87	4:4:2	95.08±1.27	4:5:2	93.93±2.21	4:[4,5]:2
Cancer	**98.74±0.61**	9:[2,3]:1	98.39±0.74	9: 2:1	98.69±0.54	9:3:1	98.31±0.69	9:[2,3]:1
Heart	83.68±2.57	26:[3,4]:1	82.89±2.46	26:3:1	83.02±2.59	26:4:1	***83.85±2.65***	26:[3,4]:1
HeartY	**84.01±3.05**	13:[4,5]:1	83.28±2.92	13:4:1	83.18±2.61	13:5:1	*83.87±2.66*	13:[4,5]:1
Hepatitis	85.26±4,34	19:[3,4]:1	86.22±4.77	19:3:1	85.78±4.12	19:4:1	**86.75±3.41**	19:[3,4]:1
Horse	85.50±2.97	83:[4,5]:1	85.21±2.98	83:4:1	85.40±2.69	83:5:1	***88.04±2.19***	83:[4,5]:1
Hypothyroid	**95.37±0.40**	29:[3,4]:3	94.69±0.39	29:3:3	94.94±0.35	29:4:3	*95.15±0.18*	29:[3,4]:3
Newthyroid	**94.81±0.89**	5:[3,4]:3	94.07±0.75	5:3:3	94.25±1.01	5:4:3	*94.38±0.59*	5:[3,4]:3
Pima	78.63±1.33	8:[3,4]:1	77.62±1.62	8:3:1	78.14±1.60	8:4:1	***78.76±1.39***	8:[3,4]:1
Waveform	84.46±0.92	40:[3,4]:2	85.35±1.45	40:3:2	85.96±1.22	40:4:2	***86.58±1.18***	40:[3,4]:2
Yeast	60.05±1.10	8:[11,12]:9	58.96±1.27	8:11:9	59.24±1.20	8:12:9	***60.33±0.61***	8:[11,12]:9
BTX	**79.68±7.39**	3:[5,6]:6	72.85±7.41	3:5:6	73.01±4.89	3:6:6	*76.34±5.65*	3:[5,6]:6
Listeria	**86.54±1.67**	4:[4,5]:1	85.06±1.13	4:4:1	85.43±0.97	4:5:1	*85.75±0.66*	4:[4,5]:1

5.1.1 Statistical Analysis

Now, we compare TSEA, EDDSig and TSEASig methodologies by means of nonparametric statistical tests. To determine whether there are significant differences we apply an Iman-Davenport test. Since two configurations were run for EDDSig, now we consider the best value of the two mean ones reported in Table 5. The average ranks of the different methodologies are 2.64, 1.78 and 1.57 respectively for EDDSig, TSEASig and TSEA. According to Iman-Davenport test results, since the statistic $F_F = 6.16$ is higher than the critical value $F(2,26) = 3.37$ at $\alpha = 0.05$ the null-hypothesis is rejected. Therefore, we proceed with post-hoc Nemenyi test. The performance of two classifiers is significantly different if the corresponding average ranks differ by at least the CD. Table 6 shows the Nemenyi test results where the ranking difference between each different pair and the detected significant difference level have been indicated for more clarity. In a single row, the CD (at $\alpha = 0.05$ and $\alpha = 0.10$) is shown.

Table 6. Pairwise comparisons of the TSEA, EDDSig and TSEASig methodologies by means of a Nemenyi test

	EDDSig	TSEASig	TSEA
EDDSig		0.86°	1.07 *
TSEASig			0.21

$CD_{(\alpha = 0.05)} = 0.89;\ CD_{(\alpha = 0.10)} = 0.78$

Each filled cell contains the ranking difference between the methods in the row and the column. Also, it is specified if the former method outperforms the latter one at a significance level of 0.05 (*) or 0.10 (°)

An analysis based upon the results Nemenyi test allow us to state the following. There are significant differences between the TSEASig and EDDSig at $\alpha = 0.10$. Consequently, TSEASig is better than EDDSig. In other words, regarding to ANNs with sigmoidal units, the methodology based in the two-stage algorithm outperforms the standard one based in EDD. Comparing TSEA and TSEASig there are not significant differences. It means that the new approach is competitive with respect to TSEA. We can conclude that the methodology based on the two-stage algorithm is suitable both for sigmoidal and product units. Finally, TSEA is significantly better than EDDSig at $\alpha = 0.05$.

5.1.2 Analysis of Computational Cost

The comparison between TSEA, EDDSig and TSEASig methodologies is completed by means of a computational cost analysis. Experiments have been run in a desktop computer with an Intel Core 2 Quad processor at 2.4GHz and 2GB RAM of physical memory. The acceleration rate of the *i* method with respect to *j* method is given by Eq. 4.

$$Acceleration\_Rate(i, j) = \frac{time(j)}{time(i)} \tag{4}$$

Table 7 reports the time results concerning to the computational cost per iteration measured in seconds (s). The first column specifies the data set name. From second to fifth columns are showed the elapsed time of an iteration with each configuration of the different methodologies. Two last columns depicted the accelerate rates for TSEASig regards to EDDSig and TSEA. Last row contains the average of the values in the column. Since two configurations of EDDSig are equivalent to one of

Table 7. Computational cost and acceleration rates of TSEA, EDDSig and TSEASig

Data set	**Computational cost (s)**				**Acceleration rate**	
	Methodologies				(TSEASig, EDDSig)	(TSEASig, TSEA)
	TSEA	EDDSig		TSEASig		
	1*	1S	2S	1S*		
Australian	303	126	155	191	1.47	1.59
Balance	287	317	370	382	1.80	0.75
Cancer	98	40	51	61	1.49	1.61
Heart	207	114	133	134	1.84	1.54
HeartY	62	34	37	51	1.39	1.22
Hepatitis	36	19	22	25	1.64	1.44
Horse	817	304	378	344	1.98	2.38
Hypothyroid	6503	4525	6090	6694	1.59	0.97
Newthyroid	122	83	101	116	1.59	1.05
Pima	105	63	73	88	1.55	1.19
Waveform	9213	5217	6412	7155	1.63	1.29
Yeast	49320	21983	30349	28538	1.83	1.73
BTX	256	158	174	190	1.75	1.35
Listeria	231	168	198	216	1.69	1.07
Average	4825.71	2367.93	3181.64	3156.07	1.66	1.37

TSEASig, the acceleration rate (TSEASig, EDDSig) is calculated as the sum of the times of 1S and 2S divided by the time of 1S*.

Having a look at the Table 7, we conclude that regarding to sigmoidal units, TSEASig is 1.66 times faster than EDDSig. In the comparison between TSEASig and TSEA, the former is 1.37 times faster than the latter. The empirical times give notice that the proposed methodology, TSEASig, is much more efficient than the previous methodology, TSEA. Moreover TSEASig is faster than EDDSig. The efficiency measures are based on the computation time of an iteration of the whole population throughout the EA running. The speed depends on the own evolutionary process and the number of mathematical operations that are involved to calculate the NN output.

5.2 Results Obtained with a Good Number of Classifiers

Now, a general review is made about the results obtained with another kind of NNs and other machine learning algorithms. In the literature, a huge amount of tests has been carried out with some of the data sets here considered. Our purpose is to view some of the methods that have been tested with some of the data sets dealt with in the current chapter.

Related to NNs, we have reported TSEASig, TSEA, the traditional MLP model [20] with a learning Back-Propagation method (BP); the RBF model [21] with a normalized Gaussian, HMOEN_L2 [22], SONG [23] and CC-EBFNN [24]. As classical or modern machine learning algorithms have been included: C4.5, k-nearest neighbours (k-NN) -with the best accuracy for *k* in {1, 3, 5, 7, 9}-, PART and SVM [25]. Since, MLP, RBF, C4.5, k-NN, PART and SVM are implemented in Weka tool [26], we have conduced the experiments. The parameters have been set to the default values with the exceptions that we describe; for BP were the following: learning rate $\eta = 0.3$, momentum $\alpha = 0.2$ and the number of epochs was adjusted in each data set. To determine the learning and the momentum we try out a grid search algorithm with values in the range [0, 1] in 0.1 steps. Regarding the topology of the models in the case of MLP and RBF we have considered the default one.

Table 8 includes the results of all classifiers, averaged for 30 runs in non-deterministic algorithms, with each of the data set; the best ones in boldface and in italics the second best ones, as well as the averages of the methods run by us with all data sets. From a purely descriptive analysis of the results, it can be concluded that SVM and RBF obtain the best result for three data sets, TSEASig, MLP, C4.5 and PART for two data sets, and TSEA and k-NN once. There is not one method that performs really well with all data sets; depending on the data set, the best classifier belongs to either the neural networks approach or to the classical/modern machine learning. Furthermore, the TSEA method achieves the highest mean accuracy ($\overline{CCR} = 85.79$), followed by the TSEASig ($\overline{CCR} = 85.61$) and RBF ($\overline{CCR} = 84.42$). The previous statements point out that methodology based on two stages is proper with product or sigmoidal units.

Table 8. Summary of the results in fourteen data sets comparing TSEASig to other methods related to neural networks or classical/modern machine learning approaches

Method	Australian	Balance	Cancer	Heart	HeartY	Hepatitis	Horse
TSEASig	86.55	*93.93*	98.31	83.85	83.87	86.75	*88.04*
TSEA	*88.11*	**96.20**	*98.74*	83.68	84.01	85.26	85.50
HMOEN_L2	-	-	96.30	-	-	80.30	-
MLP	84.10	93.78	97.81	*84.82*	**84.29**	84.73	**88.51**
RBF	75.84	88.27	97.20	**86.75**	83.79	*89.30*	80.47
SONG	-	87.80	97.40	-	-	-	-
CC-EBFNN	-	-	96.67	82.45	-	-	-
C4.5	86.71	83.33	97.13	75.00	*84.21*	84.21	*88.04*
k-NN	85.55	91.67	**98.85**	82.89	83.70	86.84	*88.04*
PART	84.97	85.26	97.13	80.26	82.12	81.58	85.87
SVM	**88.44**	88.46	98.28	82.89	80.37	**89.47**	*88.04*

Method	Hypothyroid	Newthyroid	Pima	Waveform	Yeast	BTX	Listeria
TSEASig	95.15	94.38	**78.76**	86.58	**60.33**	76.34	85.75
TSEA	95.37	94.81	*78.63*	84.46	60.05	*79.68*	*86.54*
HMOEN_L2	-	-	78.50	-	-	-	-
MLP	94.39	97.08	75.94	84.85	*60.11*	54.12	84.49
RBF	92.83	**98.27**	77.34	*87.29*	59.83	**80.95**	83.70
SONG	-	*97.20*	76.40	-	-	-	-
CC-EBFNN	-	-	76.04	-	-	-	-
C4.5	**99.15**	96.30	74.48	76.40	54.84	**80.95**	85.93
k-NN	94.06	94.44	75.00	81.12	48.39	76.19	85.93
PART	*98.83*	92.59	74.48	78.16	56.72	**80.95**	**86.67**
SVM	93.85	88.89	78.13	**88.80**	55.91	61.90	80.74

$\overline{CCR}\ (TSEASig) = 85.61$; $\overline{CCR}\ (TSEA) = 85.79$; $\overline{CCR}\ (MLP) = 83.50$;

$\overline{CCR}\ (RBF) = 84.42$

$\overline{CCR}\ (C4.5) = 83.22$; $\overline{CCR}\ (k-NN) = 83.76$; $\overline{CCR}\ (PART) = 83.26$;

$\overline{CCR}\ (SVM) = 83.16$

6 Conclusions

This chapter aims to tackle multi-classification problems using evolutionary artificial neural networks based on sigmoidal units. We have extended to sigmoidal units a previous methodology for neural networks based on product units based with an EA divided in two phases. The new approach has been called TSEASig. Our basic assumption is that it is convenient to employ a methodology based on a population with more diverse models in terms network architectures and this produces an improvement in efficiency and accuracy.

The TSEASig methodology is applied to solve fourteen classification problems, twelve from the UCI repository and two real-world problems, with a great deal of variety in the number of instances, features and classes. The test results confirm that our approach obtains promising results, achieving a high classification rate level in the data sets at a lower computational cost than EDDSig.

A comparison between TSEASig, EDDSig and TSEA has been carried out by means of nonparametric tests. The test results reveal that there are significant differences between TSEASig and EDDSig. However, significant differences are not present between TSEASig and TSEA; this fact indicates that the methodology in two stages is also suitable for sigmoidal units. According to the above results, our new learning methodology of neural networks, TSEASig, based on sigmoidal units is competitive in accuracy and more efficient than the remaining methodologies. The empirical times give notice that TSEASig is 1.37 times faster than TSEA.

We have also summarized the results obtained with other kinds of neural networks and classical/modern machine learning algorithms. From the analysis of the results we can observe the good performance of our new approach.

Acknowledgments. This chapter has been partially subsidized by TIN2007-68084-C02-02 and TIN2008-06681-C06-03 projects of the Spanish Inter-Ministerial Commission of Science and Technology (MICYT), FEDER funds and the P08-TIC-3745 project of the "Junta de Andalucía" (Spain).

References

1. Maaranen, H., Miettinen, K., Mäkelä, M.M.: Quasi-random initial population for genetic algorithms. Computers & Math. with Appl. 47, 1885–1895 (2004)
2. Wang, L., Zheng, D.Z., Tang, F.: An improved evolutionary programming for optimization. In: Proc of the 4th world Congress on Intelligent Control and Automation, vol. 3, pp. 1769–1773. IEEE, Shanghai (2002)
3. Amor, H.B., Rettinger, A.: Intelligent exploration for genetic algorithms: using self-organizing maps in evolutionary computation. In: Proc. of the 2005 Conference on Genetic and Evolutionary Computation, GECCO 2005, pp. 1531–1538. ACM, Washington DC (2005)
4. Tallón-Ballesteros, A.J., Hervás-Martínez, C.: A two-stage algorithm in evolutionary product unit neural networks for classification. Expert Syst. with Appl. 38(1), 743–754 (2011)
5. Durbin, R., Rumelhart, D.: Products units: a computationally powerful and biologically plausible extension to back-propagation networks. Neural Comp. 1(1), 133–142 (1989)
6. Miller, G.F., Toddm, P.M., Hegde, S.U.: Designing neural networks using genetic algorithms. In: Proc. of the 3rd International Conference on Genetic Algorithms, ICGA 1989, pp. 379–384. Morgan Kaufmann, George Mason University, Fairfax, Virginia, USA (1989)
7. Yao, X.: Evolving artificial neural networks. Proc. of the IEEE 87(9), 1423–1447 (1999)
8. Yao, X., Liu, Y.: A new evolutionary system for evolving artificial neural networks. IEEE Trans. on Neural Netw. 8(3), 694–713 (1997)
9. Yao, X., Liu, Y.: Making use of population information in evolutionary artificial neural networks. IEEE Trans. on Syst., Man and Cybernetics, Part B: Cybernetics 28(3), 417–425 (1998)
10. Azzini, A., Tettamanzi, A.G.B.: A new genetic approach for neural network design, vol. 82, pp. 289–323. Springer, Heidelberg (2008)
11. Martínez-Estudillo, F.J., Hervás-Martínez, C., Gutiérrez, P.A., Martínez-Estudillo, A.C.: Evolutionaryproduct-unit neural networksclassifiers. Neurocomputing 72(1-3), 548–561 (2008)

12. Martínez-Estudillo, A.C., Martínez-Estudillo, F.J., Hervás-Martínez, C., García-Pedrajas, N.: Evolutionary product unit based neural networks for regression. Neural Netw. 19, 477–486 (2006)
13. Tallón-Ballesteros, A.J., Gutiérrez-Peña, P.A., Hervás-Martínez, C.: Distribution of the search of evolutionary product unit neural networks for classification. In: Proc. of the IADIS Internacional Conference on Applied Computing, AC 2007, pp. 266–273. IADIS, Salamanca (2007)
14. Asuncion, A., Newman, D.J.: UCI Machine Learning Repository. School of Information and Computer Science. University of California, Irvine (2007), `http://www.ics.uci.edu/~mlearn/MLRepository.html`
15. Hervás, C., Silva, M., Gutiérrez, P.A., Serrano, A.: Multilogistic regression by evolutionary neural network as a classification tool to discriminate highly overlapping signals: Qualitative investigation of volatile organic compounds in polluted waters by using headspace-mass spectrometric analysis. Chemom. and Intell. Lab. Syst. 92, 179–185 (2008)
16. Beuchat, L.R.: Listeria monocytogenes: incidence on vegetables. Food Control 7(4-5), 223–228 (1996)
17. Valero, A., Hervás, C., García-Gimeno, R.M., Zurera, G.: Product unit neural network models for predicting the growth limits of Listeria monocytogenes. Food Microbiol. 24(5), 452–464 (2007)
18. Gutiérrez, P.A., Hervás, C., Lozano, M.: Designing multilayer perceptrons using a guided saw-tooth evolutionary programming algorithm. Soft Computing 14, 599–613 (2010)
19. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. 7, 1–30 (2006)
20. Bishop, C.M.: Neural networks for pattern recognition. Oxford University Press, Oxford (2004)
21. Howlett, R.J., Jain, L.C.: Radial Basis Function Networks 1: Recent Developments in Theory and Applications. Springer, Heidelberg (2001)
22. Goh, C.K., Teoh, E.J., Tan, K.C.: Hybrid multi objective evolutionary design for artificial neural networks. IEEE Trans. on Neural Netw. 19(9), 1531–1548 (2008)
23. Inoue, H., Narihisa, H.: Self-organizing neural grove and its applications. In: Proc. of the International Joint Conference on Neural Networks, IJCNN 2005, vol. 2, pp. 1205–1210. IEEE, Montreal (2005)
24. Tian, J., Li, M., Chen, F.: A hybrid classification algorithm based on coevolutionary EBFNN and domain covering method. Neural Comput. & Applic. 18, 293–308 (2009)
25. Vapnik, V.: The Nature of Statistical Learning Theory. Springer, New York (1995)
26. Witten, I.H., Frank, E.: Data Mining: Practical machine learning tools and techniques with Java implementations. Morgan Kaufmann, San Francisco (2005)

Author Index