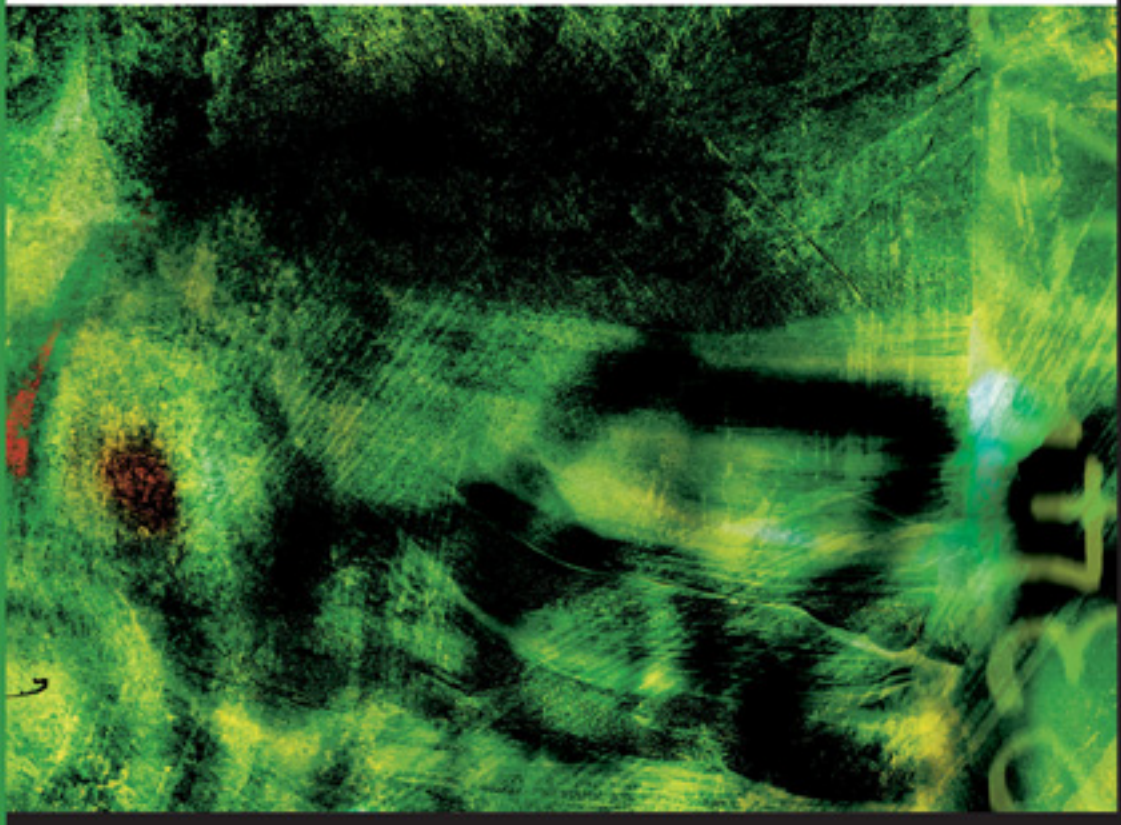
Juan R. Rabunal and Julian Dorado

Artificial Neural Networks in Real-Life Applications



Artificial Neural Networks in Real-Life Applications

Juan R. Rabuñal
University of A Coruña, Spain

Julián Dorado
University of A Coruña, Spain

IDEA GROUP PUBLISHING
Hershey • London • Melbourne • Singapore

Acquisitions Editor: Michelle Potter
Development Editor: Kristin Roth
Senior Managing Editor: Amanda Appicello
Managing Editor: Jennifer Neidig
Copy Editor: Amanda O'Brien
Typesetter: Jennifer Neidig
Cover Design: Lisa Tosheff
Printed at: Yurchak Printing Inc.

Published in the United States of America by
Idea Group Publishing (an imprint of Idea Group Inc.)
701 E. Chocolate Avenue
Hershey PA 17033
Tel: 717-533-8845
Fax: 717-533-8661
E-mail: cust@idea-group.com
Web site: http://www.idea-group.com

and in the United Kingdom by
Idea Group Publishing (an imprint of Idea Group Inc.)
3 Henrietta Street
Covent Garden
London WC2E 8LU
Tel: 44 20 7240 0856
Fax: 44 20 7379 0609
Web site: http://www.eurospanonline.com

Library of Congress Cataloging-in-Publication Data

Artificial neural networks in real-life applications / Juan Ramon Rabunal and Julian Dorrado, editors.
p. cm.
Summary: "This book offers an outlook of the most recent works at the field of the Artificial Neural Networks (ANN), including theoretical developments and applications of systems using intelligent characteristics for adaptability"--Provided by publisher.
Includes bibliographical references and index.
ISBN 1-59140-902-0 (hardcover) -- ISBN 1-59140-903-9 (softcover) -- ISBN 1-59140-904-7 (ebook)
1. Neural networks (Computer science) I. Rabunal, Juan Ramon, 1973- . II. Dorrado, Julian, 1970- .
QA76.87.A78 2006
006.3'2--dc22
2005020637

British Cataloguing in Publication Data
A Cataloguing in Publication record for this book is available from the British Library.

Artificial Neural Networks in Real-Life Applications

Table of Contents

Section III: Data Mining

Section IV: Civil Engineering

Section V: Financial Analysis

Section VI: Other Applications

Preface

Evolution and Development

Throughout the past, human beings have been concerned with how to acquire tools that might increase their potentialities, not only regarding the physical or intellectual aspect but also the metaphysical one.

At the physical aspect, the use of wheels, levers, or cams, among others, finally reached the point of elaborating hominids and automats that in their most sophisticated creations consisted of animated statues that generally reproduced daily movements. Heron of Alexandria constructed some artificial actors which represented the Trojan War, where the idea of automats reached a high level of development as it was established that: (a) the mechanisms would act depending on the internal structure; (b) the action comes from an accurate organisation of motor forces, both natural and artificial; (c) the mobile ones are the most improved, since they are able to move completely. Ultimately, they are only the expression of the unavoidable human wish to increase their possibilities in all the aspects of their lives. In this line, some of the most remarkable creations include "The Dove" by Archytas de Tarente, Archimedes' "Syracuse Defensive Mechanisms" (developed to face the Roman fleet), "The Mechanical Lion" by Leonardo Da Vinci, the clock creations of the Droz brothers at the Cathedrals of Prague and Munich, and "The Transverse Flute Player" by Vaucanson. "The Madzel Chess Automaton" by Hungary's Von Kempelen was able to play chess with the best players of its time and impressed Empress Maria Theresa of Austria. Edgar Allan Poe built a logical test trying to prove that this automaton was not authentic, but failed as he considered that the machine was not able to change its strategy as the game went on (Elgozy, 1985; Poe, 1894).

At the metaphysical aspect, the creations along time also have been numerous. The main concern in this case was "ex nihilo," the idea of a motionless-based creation of beings similar to humans that might act as substitutes to humans during the performance of the most tedious, dangerous, or unpleasant tasks. The Hephaistos (God of the Forge) androids were the first known reference to creation of artificial intelligence.

As Tetis told her son Achilles during their visit to the workshop of the god, "They were made of solid gold and they had understanding in their mind." In the modern age, "The Golem" by Loew, XVI century Prague Rabbi (Meyrink, 1972; Wiener, 1964), "The Universal Robots" by Rossum (Capek, 1923), and "Frankenstein" (Shelley, 1818) should be highlighted as well.

But what is really interesting is the third of the mentioned aspects: the attempt to reproduce and promote the intellect. Multiple mechanical devices, specifically the abacus, were designed in order to improve the capability of calculation. In the Middle Ages, the Majorcan Ramón Llul developed the Ars Magna, a logical method that exhaustively and systematically tested all the possible combinations. Later, in the Modern Age, some of the most noticeable devices are "The Pascal Machines" and the works of several authors such as Leibnitz, Freege, or Boole. Ada Lovelance, Charles Babbage's co-worker at the analytic machine, established "The Lovelance Regime," where she states that "machines only can do those things we know how to tell them to do, so their mission is helping to supply or to obtain what is already known.". Other important contributions of the second half of 20th century in this field include "The Logical Theoretical" by Bewel, "The General Problem Solver" by Shaw, Newell, and Simon, the program for draughts play by Samuel, and the developments of the first computers by Zuse and Sreyers (Samuel, 1963; Erns, 1969).

The appearance of computers and computer software is the key point in the real development of certain characteristics of intelligent beings such as the capabilities of memory or calculus, although most of these characteristics still are merely outlined when replicated in artificial systems. In this way, and despite the high rhythm of advances during the last decades, we are still too far from artificially reproducing something that is so inherent to human beings, such as creativity, criticism capability (including self-criticism), conscience, adaptation capability, learning capability, or common sense, among others.

Artificial intelligence (AI) is an area of multidisciplinary science that comes mainly from cybernetics and deals with the deeper study of the possibility — from a multidisciplinary, but overall engineering, viewpoint — of creating artificial beings. Its initial point was Babbage's wish for his machine to be able to "think, learn, and create" so that the capability for performing these actions might increase in a coextensive way with the problems that human beings deal with (Newel & Simon, 1972). AI — whose name is attributed to John McCarthy from the Dormouth College group of the summer of 1956 — is divided into two branches known as symbolic and connectionist, depending on whether they respectively try to simulate or to emulate the human brain in intelligent artificial beings. Such beings are understood as those who present a behaviour that, when performed by a biological being, might be considered as intelligent (McCorduck, 1979; McCarthy, 1958).

The main precursor of connectionist systems from their biological fundaments was from Spanish Nobel Award-winning Dr. Santiago Ramón y Cajal who, together with Sherringon, Williams y Pavlov, tried to approach the information processes of the brain by means of an experimental exploration and also described the first connectionist system with the statement: "When two brain procedures are active at the same time or consecutively, one tends to propagate its excitation to the other" (Ramón y Cajal, 1967; Ramón y Cajal, 1989).

In the dawn of cybernetics, and within that field, three papers published in 1943 constituted the initiation of the connectionist systems (Wiener, 1985). The first of these works was written by McCulloch and Pitts. Apart from revealing how machines could use such concepts as logic or abstraction, they proposed a model for an artificial neuron, named after them. This model, together with the learning systems, represented the foundations of connectionist systems. Most of the mentioned systems derive from the Hebb Rule, which postulates that a connection between neurons is reinforced every time that this connection is used (McCulloch & Pitts, 1943).

The second work was by Rosemblueth, Wiener, and Bigelow, who suggested several ways of providing the machines with goals and intentions (Rosemblueth, Wiener, & Bigelow, 1943). In the last work, Craik proposed the use of models and analogies by the machines for the resolution of problems, which established that the machines have certain capabilities of abstraction (Craik, 1943).

These three contributions were added to some others: "The Computer and the Brain" by Von Neumann;, "The Turing Machine" by Turing — a theory that preceded actual computers; and "The Perceptron" by Rosemblatt — the first machine with adaptable behaviour able to recognise patterns and provide a learning system where stimulus and answers are associated by means of the action of inputs (Turing, 1943; Von Nuemann, 1958).

During the second half of the 20th century, numerous authors made important contributions to the development of these types of intelligent systems. Some of the most remarkable are Anderson, who made the first approaches to the Associative Lineal Memory, Fukushima, Minsky, Grossberg, Uttley, Amari, McClelland, Rumelhart, Edelman, and Hopfield. They contribute with different cell models, architectures, and learning algorithms, each representing the basis for the most biological AI systems, which eventually resulted in the most potent and efficient ones (Raphael, 1975; Minsky, 1986; Minsky & Papert, 1968; Rumelhart & McClelland, 1986).

These systems are quite interesting due, not only to their ability for both learning automatically and working with inaccurate information or with failures in their components, but also because of their similarities with the neurophysiologic brain models, so that the advances in both disciplines might be exchanged for their reinforcement, indicating a clear symbiosis between them.

Present and Future Challenges

All these studies and investigations have achieved spectacular results, although they are still far from the daily performance of biological systems. Besides, during the last decades, the expectation for these type of systems has broadened due to the miniaturisation of computers coupled with the increment of their capacities for calculus and information storage. In this way, more complex systems are being progressively implemented in order to perform already demanded functions as well as those that will be coming soon and are unforeseen.

The efforts made so far represent two sides: On the one hand, they are the basis for all the advances achieved up to this moment in order to reinforce or reproduce the charac-

teristics that define the intelligent beings; on the other hand, they also reflect the poor — although spectacular — advances achieved with regards to the creation of truly intelligent artificial beings. While the connectionist systems are the most advanced ones in the field of emulation of biological intelligent systems, certain restrictions are present. These limitations are mainly referred to the need to reduce the time for training and to optimise the architecture — or network topology — as well as to the lack of explanation for their behaviour and to the approach to more complex problems. For the two first restrictions, there is a new technique based on genetics, known as genetic algorithms (GA) (Holland, 1975), proposed by Holland and developed until genetic programming in the last decade by Koza (1992) among others. These techniques have proved to be useful for the extraction of new knowledge from the system, using the data mining process.

The two other restrictions might be palliated by incoming solutions such as those suggested with the incorporation of artificial glia cells to the Artificial Neural Networks (ANN). This adventurous proposal is currently being elaborated by our research group of La Coruña University, co-working at the neuroscience aspects with Professors Araque and Buño, of the Santiago Ramón y Cajal Scientific Research Institute.

It seems necessary to look again toward nature, such as it was done when the wider steps were taken along this track, looking for new guides and new information for the search of solutions. And the nature, as it has been mentioned, contributes again with solutions.

Technology also tries to provide solutions. In this line, it is intended to integrate different disciplines under a common label: MNBIC (Micro and Nanotechnologies, Biotechnology, Information Technologies, and Cognitive Technologies) Convergent Technologies. The MNBIC promise to be a revolution at the scientific, technologic, and socioeconomic fields because they contribute to help make possible the construction of hybrid systems: biological and artificial.

Some of their possibilities consist on the use of micro or nano elements that might be introduced into biological systems in order to substitute dysfunctional parts of it, whereas biological particles might be inserted into artificial systems for performing certain functions. According to a recent report of the U.S. National Science Foundation, "The convergence of micro and nanoscience, biotechnology, information technology, and cognitive science (MNBIC) offers immense opportunities for the improvement of human abilities, social outcomes, the nation's productivity, and its quality of life. It also represents a major new frontier in research and development. MNBIC convergence is a broad, cross-cutting, emerging, and timely opportunity of interest to individuals, society, and humanity in the long term."

There is a scientific agreement with regards to the fact that the most complex part for being integrated with the rest of the convergent technologies is the one that represents the cognitive science. The part that has to do with technologies of knowledge has a best level of integration through models of knowledge engineering. It is remarkable that the interaction of the connectionist branch with other disciplines such as the GAs and the introduction of other elements, representing the cells of the glial system, are different from neurons.

Book Organization

This book is organized into six sections with 16 chapters. A brief revision of each chapter is presented as follows:

Section I presents recent advances in the study of biological neurons and also shows how these advances can be used for developing new computational models of ANNs.

- Chapter I shows a study that incorporates, into the connectionist systems, new elements that emulate cells of the glial system. The proposed connectionist systems are known as artificial neuroglial networks (ANGN).
- Chapter II expands artificial neural networks to artificial neuroglial networks in which glial cells are considered.

New techniques such as connectionist techniques are preferred in cases like the time series analysis, which has been an area of active investigation in statistics for a long time, but has not achieved the expected results in numerous occasions. Section II shows the application of ANNs to predict temporal series.

- Chapter III shows a hybrid evolutionary computation with artificial neural network combination for time series prediction. This strategy was evaluated with 10 time series and compared with other methods.
- Chapter IV presents the use of artificial neural networks and evolutionary techniques for time series forecasting with a multilevel system to adjust the ANN architecture.

In the world of databases the knowledge discovery (a technique known as data mining) has been a very useful tool for many different purposes and tried with many different techniques. Section III describes different ANNs-based strategies for knowledge search and its extraction from stored data.

- Chapter V describes genetic algorithm-based evolutionary techniques for automatically constructing intelligent neural systems. This system is applied in laboratory tests and to a real-world problem: breast cancer diagnosis.
- Chapter VI shows a technique that makes the extraction of the knowledge held by previously trained artificial neural networks possible. Special emphasis is placed on recurrent neural networks.
- Chapter VII shows several approaches in order to determine what should be the most relevant subset of variables for the performance of a classification task. The solution proposed is applied and tested on a practical case in the field of analytical chemistry, for the classification of apple beverages.

The advances in the field of artificial intelligence keep having strong influence over the area of civil engineering. New methods and algorithms are emerging that enable civil engineers to use computing in different ways. Section IV shows two applications of ANNs to this field. The first one is referred to the hydrology area and the second one to the building area.

- Chapter VIII describes the application of artificial neural networks and evolutionary computation for modeling the effect of rain on the run-off flow in a typical urban basin.
- Chapter IX makes predictions of the consistency of concrete by means of the use of artificial neuronal networks

The applications at the economical field, mainly for prediction tasks, are obviously quite important, since financial analysis is one of the areas of research where new techniques, as connectionist systems, are continuously applied. Section V shows both applications of ANNs to predict tasks in this field; one of them is for bond-rating prediction, and the other for credit-rating prediction:

- Chapter X shows an application of soft computing techniques on a high dimensional problem: bond-rating prediction. Dimensionality reduction, variable reduction, hybrid networks, normal fuzzy, and ANN are applied in order to solve this problem.
- Chapter XI provides an example of how task elements for the construction of an ANN can be automated by means of an evolutionary algorithm, in a credit rating prediction.

Finally, section VI shows several applications of ANNs to really new areas, demonstrating the interest of different science investigators in facing real-world problems.

As a small sample of the areas where ANNs are used, this section presents applications for music creation (Chapter XII), exploitation of fishery resources (Chapter XIII), cost minimisation in production schedule setting (Chapter XIV), techniques of intruder detection (Chapter XV), and an astronomy application for stellar images (Chapter XVI).

- Chapter XII explains the complex relationship between music and artificial neural networks, highlighting topics such as music composition or representation of musical language.
- Chapter XIII approaches the foundations of a new support system for fisheries, based on connectionist techniques, digital image treatment, and fuzzy logic.
- Chapter XIV proposes an artificial neural network model for obtaining a control strategy. This strategy is expected to be comparable to the application of cost estimation and calculation methods.

- Chapter XV shows a novel hybrid method for the integration of rough set theory, genetic algorithms, and an artificial neural network. The goal is to develop an intrusion detection system.
- Finally, Chapter XVI describes a hybrid approach to the unattended classification of low-resolution optical spectra of stars by means of integrating several artificial intelligence techniques.

Relevance and Conclusions

As can be observed, this book tries to offer an outlook of the most recent works in the field of the connectionist AI. They include not only theoretical developments of new models for constitutive elements of connectionist systems, but also applications of these systems using intelligent characteristics for adaptability, automatic learning, classification, prediction, and even artistic creation.

All this being said, we consider this book a rich and adventurous, but well-based, proposal that will contribute to solving old problems of knowledge-based systems and opening new interrogations which, without doubt, will make the investigations advance through this field.

This is not a book of final words or definitive solutions, rather it contributes new and imaginative viewpoints, as well as small — or big — advances in the search of solutions for achieving truly intelligent artificial systems.

Prof. Alejandro Pazos

Department of Information and Communications Technologies

University of A Coruña, Spain

2005

References

Berry, A. (1983). *La máquina superinteligente.* Madrid: Alianza Editorial.

Capek, K. (1923). *R.U.R. (Rossum's Universal Robots).* Garden City, NY: Doubleday, Page and Co.

Craik, K. J. W. (1943). *The nature of explanation*. Cambridge: Cambridge University Press.

Elgozy, G. (1985). *Origines de l´informatique.* Paris: Technique de L´Ingenieur Press.

Ernst, G. W., & Newell, A. (1969). *GPS: A case study in generality and problem solving.* New York: Academic Press.

Holland, J. H. (1975). *Adaptation in natural and artificial systems.* Ann Arbor: The University of Michigan Press.

Koza, J. (1992). *Genetic programming. On the programming of computers by means of natural selection*. Cambridge, MA: MIT Press.

McCarthy, J. (1958). Programs with common sense. *Proceedings of the Teddington Conference on the Mechanisation of Thought Processes*. London: H.M. Stationery.

McCorduck, P. (1979). *Machines who think.* San Francisco: W.M. Freeman and Co.

McCulloch W., & Pitts, W. (1943). A logical calculus of ideas imminent in nervous activity. In *Bull. of Mathematical Biophysics*. Colorado Springs: The Dentan Printing Co.

Meyrink, A. (1972). *El Golem*. Barcelona: Tusquet editores, S.A.

Minsky, M. (1986). *Society of mind.* New York: Simon & Schuster.

Minsky, M., & Papert, S. (1968). *Perceptrons*. Cambridge, MA: MIT Press.

Newell, A., & Simon, H. A. (1972). *Human problem solving.* NJ: Prentice Hall.

Poe, E. A. (1894). *The works of Edgar Alan Poe*. New York: The Colonial Company.

Ramón y Cajal, S. (1967). *The structure and connexions of nervous system*. Nobel Lectures: Physiology or Medicine: Ed. Elsevier Science Publishing Co.

Ramón y Cajal, S. (1989). *Textura del sistema nervioso del hombre y de los vertebrados*. Madrid, Spain: Ed Alianza.

Raphael, B. (1975). *The thinking computer.* San Francisco: W.H. Freeman.

Rosemblueth, A., Wiener, N., & Bigelow, J. (1943). *Behaviour, purpose and teleology. Philosophy of science.* Boston: Harvard Medical School Press.

Rumelhart, D. E., & McClelland, J. L. (1986). *Parallel distributed processing*. Cambridge, MA: MIT Press.

Samuel, A. L. (1963). *Some studies in machine learning using the game of checkers*. New York: McGraw Hill.

Shelley, M. (1818). *Frankenstein, or the modern Prometheus.* London: Lackington, Allen and Co.

Turing, A. (1943). *Computing machinery and intelligence*. Cambridge, MA: MIT Press.

Von Neumann, J. (1958). *The computer and the brain.* New Haven, CT: Yale University Press.

Wiener, N. (1964). *God and Golem.* Cambridge, MA: MIT Press.

Wiener, N. (1985). *Cibernética o el control y comunicaciones en los animales y las máquinas*. Barcelona, Spain: Ed. Busquets.

Acknowledgments

The editors would like to acknowledge the help of all the people involved with the collation and review process of the book, without whose support the project could not have been satisfactorily completed. A further special note of thanks also goes to all the staff at Idea Group Inc., whose contributions throughout the whole process, from the inception of the initial idea to the final publication, have been invaluable; In particular, to Jan Travers, Michele Rossi, and Kristin Roth, who continuously prodded us via e-mail to keep the project on schedule, and to Mehdi Khosrow-Pour, whose enthusiasm motivated us to initially accept his invitation to take on this project.

Most of the authors of the included chapters also served as referees for articles written by other authors. Our acknowledgement goes to all those who provided constructive and comprehensive reviews.

In closing, we wish to thank all of the authors for their insights and excellent contributions to this book. We also want to thank the resources and support of the staff of RNASA-LAB (Artificial Neural Network and Adaptive Systems Laboratory) as well as the TIC Department (Department of Information and Communications Technologies) and the CITEEC (Centre of Technological Innovations in Construction and Civil Engineering). All of them included at the University of A Coruña.

Finally, Juan R. Rabuñal wants to thank his wife María Rodríguez, his son Diego, and his family for their love and patience. Julián Dorado wants to thank his girlfriend Nieves Pedreira and his family for their love and support throughout this project.

Section I

Biological Modelization

Chapter I

Neuroglial Behaviour in Computer Science

Ana B. Porto, University of A Coruña, Spain

Alejandro Pazos, University of A Coruña, Spain

Abstract

This chapter presents a study that incorporates into the connectionist systems new elements that emulate cells of the glial system. More concretely, we have considered a determined category of glial cells known as astrocytes, which are believed to be directly implicated in the brain's information processing. Computational models have helped to provide a better understanding of the causes and factors that are involved in the specific functioning of particular brain circuits. The present work will use these new insights to progress in the field of computing sciences and artificial intelligence. The proposed connectionist systems are called artificial neuroglial networks (ANGN).

Introduction

The analysis of the computational models developed up to the present day show that the artificial neural networks (ANN) have certain limits as information processing paradigms. We believe that these limitations may be due to the fact that the existing models neither

reflect certain behaviours of the neurons nor consider the participation of elements that are not artificial neurons. Since the ANN pretend to emulate the brain, researchers have tried to represent in them the importance the neurons have in the nervous system (NS). However, during the last decades, research has advanced remarkably in the field of neuroscience, and increasingly complex neural circuits, as well as the glial system (GS), are being observed closely. The importance of the functions of the GS leads researchers to think that their participation in the processing of information in the NS is much more relevant than previously assumed. In that case, it may be useful to integrate into the artificial models other elements that are not neurons. These assisting elements, which until now have not been considered in the artificial models, would be in charge of specific tasks, such as the acceleration of the impulse transmission, the establishment of the best transmission routes, the choice of the elements that constitute a specific circuit, the "heuristic" processing of the information (warning the other circuits not to intervene in the processing of certain information), and so forth.

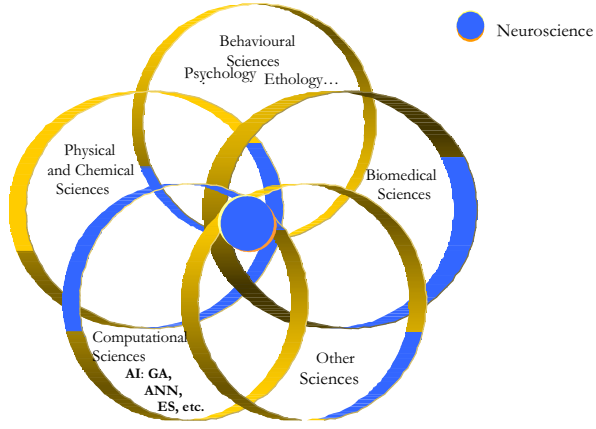
Neuroscience and Connectionist Systems

In order to create ANN that emulate the brain and its tremendous potentiality, we must know and thoroughly understand its structure and functioning; unfortunately, and in spite of numerous discoveries in the course of the last decades, the NS remains a mystery, as Cajal (1904) already predicted a century ago.

Many studies on specialised knowledge fields led to the NS. In biology, for instance, we can study the different forms of animal life and its astounding diversity without realizing that all these shapes depend on a corresponding diversity in NS. The study of the behavioural models of animals in their natural habitat, whose most renowned researcher Lorenz (1986) created hundreds of behavioural models that can be implanted into computers, is known as ethology, and the interrelation of these models and the nervous mechanism is called neuroethology. As such, the study of biological behaviour from a computational point of view could be called "computational neuroethology" or "computoneuroethology". In general psychology, relevant studies from the perspective of computational neuroethology will raise many questions on the mechanisms in the brain which determine human behaviour and abilities. Recently, neuroscientists have disposed of a wide array of new techniques and methodologies that proceeded from the fields of cellular and molecular biology and genetics. These research fields have contributed significantly to the understanding of the NS and the cellular, molecular, and genetic mechanisms that control the nervous cells; they also constitute the first step toward the processing and storage of the NS's information.

It is commonly known that many fields of the learning process imply the NS. Neuroscience can therefore be seen as the intersection of a wide range of overlapping interest spheres. It is a relatively new field that reflects the fact that, until recently, many of the disciplines that compose it had not advanced sufficiently to be intersected in a significant manner: behavioural sciences (psychology, ethology, etc.), physical and chemical sciences, biomedical sciences, artificial intelligence, and computational sciences.

Figure 1. Science fields that contribute to neuroscience



In neuroscience, the study of the NS of vertebrates is increasingly compelled to take into account various elements and points of view. Until a few decades ago, these studies were mainly focused on the analysis of the neurons, but now that the relevance of other cellular types such as the glial cells is being reconsidered, it becomes obvious that the focus must be widened and the research orientation renewed.

Astrocytes: Functions in Information Processing

Since the late 1980s, the application of innovative and carefully developed cellular and physiological techniques (such as patch-clamp, fluorescent ion-sensible images, confocal microscopy, and molecular biology) to glial studies has defied the classic idea that astrocytes merely provide a structural and trophic support to neurons and suggests that these elements play more active roles in the physiology of the central nervous system (CNS).

New discoveries are now unveiling that the glia is intimately linked to the active control of neural activity and takes part in the regulation of synaptic neurotransmission. We know that the astrocytes have very important metabolic, structural, and homeostatic functions, and that they play a critical role in the development and the physiology of the CNS, involved as they are in key aspects of the neural function, such as trophic support (Cajal, 1911), neural survival and differentiation (Raff et al., 1993), neural guidance (Kuwada, 1986; Rakic, 1990), external growth of neurites (LeRoux & Reh, 1994) and

synaptic efficiency (Mauch et al., 2001; Pfrieger & Barres, 1997). Astrocytes also contribute to the brain's homeostasis by regulating local ion concentrations (Largo, Cuevas, Somjen, Martin del Rio, & Herreras, 1996) and neuroactive substances (Mennerick & Zorumski, 1994; Largo et al., 1996). Some of these aspects will be briefly described hereafter, but we can already affirm that they are very interesting from the point of view of the connectionist systems (CS), because they directly affect the topology, number, and specificity of its elements and layers.

Rackic and Kimelberg have shown that neurons usually migrate from one place to another by means of a type of scaffold or safety route, linked to the prolongations of the immature glial cells that afterwards disappear and transform into astrocytes (Rakic, 1978; Kimelberg, 1983). The traditional functions of neural support, maintenance, and isolation that are usually attributed to the glia must therefore be completed with the functions of growth "guide" and the possible regeneration of neurons. Also, the astrocytes take care of the detoxification of products of the cerebral metabolism, which contain a high concentration of glutamine-synthetase enzymes, carbon anhidrasis, and potassium-dependent ATP-ase — elements that contribute to maintain a narrow homeostasis of ammoniac, hydrogenion-CO2, and potassium in the extracellular cerebral environment.

The astrocytes also carry out active missions in the cerebral physiology. They play a decisive role in the metabolism of the neurotransmitters glutamate and gamma-amino butyric acid (GABA), for instance, which are both caught by the astrocyte of the synaptic fissure and metabolised to form glutamine, an amino acid that subsequently helps to synthesise new neurotransmitters. Noremberg, Hertz, and Schousboe (1988) demonstrated that the enzyme that is responsible for the synthesis of glutamine is found exclusively in the astrocytes, which are responsible for the adequate presence of an element that is crucial for the transfer of information between the neurons.

On the other hand, astrocytes are cells in which glucogene can accumulate as a stock and a source of glucosis and used when needed. Glucogenolysis (liberation of glucose) is induced by different neurotransmitters such as noradrenaline and the vasointestinal peptid, substances for which the membrane of the astrocytes has receptors whose internal mechanism is not yet well understood. They also maintain the osmotic balance of the brain by reacting in case of metabolical aggressions like ischemia, increasing rapidly in size or increasing the size of their mitochondria (Smith-Thier, 1975).

When the NS is damaged, the astrocytes can cleanse and repair, together with the microglial cells. To this effect, they undergo a series of morphological and functional transformations, acquire proliferative qualities and become reactive astrocytes, which form a glial scar around the injured area, isolate it from the rest of the nervous tissue, and hereby repair the information process between the neurons.

Another important function of the astrocytes is the "spatial buffering of potassium". Kuffler and his research team discovered that the astrocytes remove the surplus of potassium that is generated by the neural activity in the extracellular space. This function eliminates the noise that could be caused by the presence of the potassium and is therefore important for the information transfer.

Given this variety in functions, it is not surprising that alterations in the astrocytes cause large numbers of pathologies in the NS. In some neurological alterations, there are

obvious anomalies in the astrocytes, whereas in other cases, these anomalies precede those of the neurons. Famous examples are epilepsy, Parkinson's, multiple sclerosis, and certain psychiatric alterations (Kimelberg, 1989).

Whereas until very recently stem cells had only been detected in the spinal marrow, the umbilical cord, and in foetal tissue, in 2004, Sanai, Tramontin, Quiñones, Barbaro, and Gupta discovered the existence of stem cells in the adult human brain (Sanai et al., 2004). They located a band of stem cells that could potentially be used for the regeneration of damaged brain tissue and shed new light on the most common type of brain tumour. Inside a brain cavity filled with brain fluids, the subventricular area, they discovered a layer of astrocytes that, cultivated in vitro, can convert themselves into neurons, which may mean that the astrocytes can regenerate themselves and produce various types of brain cells. Even though their capacity to renew the neurons does not seem to work in vivo, they obviously have great potential and must be further analysed to decypher the mechanisms that control them.

Many receptors and second messengers also are being discovered in the astrocytes, and some studies indicate that they have receptors for various neurotransmitters; even though the function of these receptors is not completely clear, their presence leads us to believe that the astrocytes respond to the changing conditions of the brain with a versatility that may be similar to that of the neurons and even superior.

Communication Between Astrocytes and Neurons: New Concept of Synapse

The astrocytes liberate chemical transmitters, and, more particularly, the increase in calcium that takes place in their interior when they are excited (Verkhratsky, Orkand, & Kettenmann, 1998) leads toward the release of glutamate, the most abundantly present excitatory neurotransmittor of the brain. At present, the functions of the liberation of chemical gliotransmittors are not entirely defined, but it is already clear that the stimulation that elevates the astrocytic calcium, indicating the activation of these cells, releases the glutamate. This glutamate release could lead to the modulation of the transmission in local synapses (Haydon & Araque, 2002) and has indeed been considered in the present research, since we have tried to modulate the synapses produced between the artificial neurons of a network through the presence and performance of elements that represent astrocytes in that network.

In recent years, abundant evidence has suggested the existence of bidirectional communication between astrocytes and neurons, and the important active role of the astrocytes in the NS's physiology (Araque, Carmignoto, & Haydon, 2001; Perea & Araque, 2002). This evidence has led to the proposal of a new concept in synaptic physiology, the tripartite synapse, which consists of three functional elements: the presynaptic and postsynaptic elements and the surrounding astrocytes (Araque, Púrpura, Sanzgiri, & Haydon, 1999). The communication between these three elements has highly complex characteristics, which seem to reflect more reliably the complexity of the information processing between the elements of the NS (Martin & Araque, 2005).

So there is no question about the existence of communication between astrocytes and neurons (Perea & Araque, 2002). In order to understand the motives of this reciprocated signaling, we must know the differences and similarities that exist between their properties. Only a decade ago, it would have been absurd to suggest that these two cell types have very similar functions; now we realise that the similarities are striking from the perspective of chemical signaling. Both cell types receive chemical inputs that have an impact on the ionotropic and metabotropic receptors. Following this integration, both cell types send signals to their neighbours through the release of chemical transmittors. Both the neuron-to-neuron signaling and the neuron-to-astrocyte signaling show plastic properties that depend on the activity (Pasti, Volterra, Pozzan, & Carmignoto, 1997). The main difference between astrocytes and neurons is that many neurons extend their axons over large distances and conduct action potentials of short duration at high speed, whereas the astrocytes do not exhibit any electric excitability but conduct calcium spikes of long duration (tens of seconds) over short distances and at low speed. The fast signaling and the input/output functions in the central NS that require speed seem to belong to the neural domain. But what happens with slower events, such as the induction of memories, and other abstract processes such as thought processes? Does the signaling between astrocytes contribute to their control? As long as there is no answer to these questions, research must continue; the present work offers new ways to advance through the use of artificial intelligence techniques.

We already know that astrocytes are much more prominent in the more advanced species. Table 1 shows the filogenetic comparison elaborated by Haydon (2001).

For the lower species on the filogenetic scale, which survive perfectly with a minimal amount of glial cells, the reciprocate signaling between glia and neurons does not seem to be very important.

However, the synaptic activity increases the astrocytic calcium, the gliotransmission (transmittor release dependant on calcium from the astrocytes) modulates the synapse and may improve the synaptic transmission in the hypocampus in the long term. This means that the glial cells are clearly implied in the signaling of the NS. The release of transmittors by the astrocytes could modulate the neural function and change the threshold for various events; for instance, by releasing glutamate locally, the astrocytes would modulate the threshold for synaptic plasticity and neural excitability (Martin & Araque, 2005). Combining this with their potential to provoke the spatial synchronisation of up to 140,000 synapses each, the astrocytes could add a new layer of information processing and biochemical integration that helps to establish at least some of the differences between the capacities of the NSs of humans, rats, fruit flies, and nemathods.

There is obviously no doubt concerning the high conduction speed of the electric impulse through the neurons. The propagation of this high-speed action potential is essential to control our behaviour and ensure our survival. It is not so clear, however, whether high-speed conduction is necessary and exclusive for many of the intellectual and plastic processes of the NS. Researchers believe that the propagation of the signal in the glial cells at speeds six times slower than the action potential may be sufficiently fast to contribute to many of the plastic and intellectual processes of the NS (Haydon & Araque, 2002).

Antecedents

Introduction

Since its early beginnings, artificial intelligence has been focused on improvements in the wide field of computer sciences, and has contributed considerably to the research in various scientific and technical areas. This work particularly considers the use of the computational modeling technique in the field of artificial intelligence.

There are two types of computational models in the present study context: The first type is based on an axiomisation of the known structures of the biological systems and the subsequent study of the provoked behaviour. Researchers usually apply this work method; the second type, mainly used by engineers, consists in axiomising or specifying a behaviour and afterwards trying to build structures that execute it.

McCulloch and Pitts (1943), mentioned at the beginning of this chapter, and other authors such as Wiener (1985) and Von Neumann (1958), in their studies on cybernetics and their theory on automats, were the first to tackle the problem of the integration of biological processes with engineering methods. McCulloch and Pitts (1943) proposed the artificial neuron model that now carries their name: a binary device with two states and a fixed threshold that receives excitatory connections or synapses, all with the same value and inhibitors of global action. They simplified the structure and functioning of the brain neurons, considering them devices with m inputs, one single output, and only two possible states: active or inactive. In this initial stage, a network of artificial neurons was a collection of McCulloch and Pitts neurons, all with the same time scales, in which the outputs of some neurons were connected to the inputs of others. Some of the proposals of McCulloch and Pitts have been maintained since 1943 without modifications, and others have evolved, but all the mathematical formalisations on the ANN that were elaborated after them have used biological systems as a starting point for the study of biological neural networks, without pretending to be exact models. The recent revival of the ANN is to a great extent due to the presentation of certain models that are strongly inspired by biologists (Hopfield, 1989).

Table 1. Filogenetic comparison of glia in various species

Species	Proportion glia:neuron
Nemathods	<1
Rodents	1:1
Human brain	~50:1

Artificial Neural Networks

Computers that are able to carry out 100 million operations in floating point per second are nevertheless unable to understand the meaning of visual shapes, or to distinguish between various types of objects. Sequential computation systems are successful in resolving mathematical or scientific problems; in creating, manipulating, and maintaining databases; in electronic communications; in the processing of texts, graphics, and auto-editing; and even in making control functions for electric household devices more efficient and user friendly; but they are virtually illiterate in interpreting the world.

It is this difficulty, typical for computing systems based on Von Neumann's sequential system philosophy (Neumann, 1956), which has pushed generations of researchers to focus on the development of new information processing systems, the ANN or CS, which solve daily problems the way the human brain does. This biological organ has various characteristics that are highly desirable for any digital processing system: It is robust and fault tolerant, neurons die every day without affecting its functioning; it is flexible since it adjusts to new environments through "Socratic" learning (i.e., through examples), and as such does not necessarily require programming; it can manage diffuse information (inconsistent or with noise); it is highly parallel and therefore efficient (effective in time); and it is small, compact, and consumes little energy. The human brain is indeed a "computer" that is able to interpret imprecise information from the senses at a considerable pace. It can discern a whisper in a noisy room, recognize a face in a dark alley, and read between the lines. And most surprisingly, it learns to create the internal representations that make these abilities possible without explicit instructions of any kind.

The ANN or CS emulate the biological neural networks in that they do not require the programming of tasks but generalise and learn from experience. Current ANN are composed by a set of very simple processing elements (PE) that emulate the biological neurons and by a certain number of connections between them. They do not execute instructions, respond in parallel to the presented inputs, and can function correctly even though a PE or a connection stops functioning or the information has a certain noise level. It is therefore a fault and noise tolerant system, able to learn through a training process that modifies the values associated to the PE connections to adjust the output offered by the system in response to the inputs. The result is not stored in a memory position; it is the state of the network for which a balance is reached. The knowledge and power of an artificial neural network does not reside in its instructions but in its topology (position of the PE and the connections between them), in the values of the connections (weights) between the PE, and the functions that define its elements and learning mechanisms.

The CS offer an alternative to classic computation for problems of the real world that use natural knowledge (which may be uncertain, imprecise, inconsistent, and incomplete) and for which the development of a conventional programme that covers all the possibilities and eventualities is unthinkable or at least very laborious and expensive. In Pazos (1991) we find several examples of successful applications of CS: image and

voice processing, pattern recognition, adaptive interfaces for man/machine systems, prediction, control and optimisation, signals filtering, and so forth.

Different ANN Types

Since the early beginnings of ANN, researchers have developed a rather large number of ANN types and implementations from the concept of simple PE, that is, the copy of the natural neuron and its massive interconnections. Even though all these types are similar where neurons and connections are concerned, they vary significantly in topology, dynamics, feed, and functions. There also have been, and there continue to be, many advances and varieties in the field of learning algorithms. Some present new learning types, while others offer minor adjustments in already existing algorithms in order to reach the necessary speed and computational complexity.

On the one hand, the presence of such a large amount of possibilities is an advantage that allows the experimentation of various networks and training types; on the other hand, it presents at least two doubts. First, how do we know which is the best option to solve a determined problem? Mathematically speaking, it is impossible to know that the final choice is indeed the best. Second, would it not be better to wait for future improvements that will substantially contribute to solving the problems of ANN, instead of tackling them with the tools that are available today?

Nevertheless, it remains true that all the design possibilities, for the architecture as well as for the training process of an ANN, are basically oriented toward minimising the error level or reducing the system's learning time. As such, it is in the optimisation process of a mechanism, in this case the ANN, that we must find the solution for the many parameters of the elements and the connections between them.

Considering what has been said about possible future improvements that optimise an ANN with respect to minimal error and minimal training time, our models will be the brain circuits, in which the participation of elements of the GS is crucial to process the information. In order to design the integration of these elements into the ANN and elaborate a learning method for the resulting ANGN that allows us to check whether there is an improvement in these systems, we have analysed the main existing training methods that will be used for the elaboration. We have analysed non-supervised and supervised training methods, and other methods that use or combine some of their characteristics and complete the analysis: training by reinforcement, hybrid training, and evolutionary training.

Some Observed Limitations

Several experiments with ANN have shown the existence of conflicts between the functioning of the CS and biological neuron networks, due to the use of methods that did not reflect reality. For instance, in the case of a multilayer perceptron, which is a simple CS, the synaptic connections between the EP have weights that can be excitatory or

inhibitory, whereas in the natural NS, the neurons seem to represent these functions, not the connections; recent research (Perea & Araque, 2002) indicates that the cells of the GS, more concretely the astrocytes, also play an important role.

Another limitation concerns the learning algorithm known as "backpropagation", which implies that the change of the connections value requires the backwards transmission of the error signal in the ANN. It was traditionally assumed that this behaviour was impossible in a natural neuron, which, according to the "dynamic polarisation" theory of Cajal (1904), is unable to efficiently transmit information inversely through the axon until reaching the cellular soma; new research, however, has discovered that neurons can send information to presynaptic neurons under certain conditions, either by means of existing mechanisms in the dendrites or else through various interventions of glial cells such as astrocytes.

If the learning is supervised, it implies the existence of an "instructor", which in the context of the brain means a set of neurons that behave differently from the rest in order to guide the process. At present, the existence of this type of neuron is biologically indemonstrable, but the GS seems to be strongly implied in this orientation and may be the element that configures an instructor that until now had not been considered.

These differences between the backpropagation models and the natural model are not very important in themselves. The design of artificial models did not pretend to obtain a perfect copy of the natural model but a series of behaviours whose final functioning approached it as much as possible. Nevertheless, a close similarity between both is indispensable to improve the output and increase the complexity of the ANN and may result in more "intelligent" behaviours. It is in this context that the present study analyses to what extent the latest discoveries in neuroscience (Araque et al., 2001; Perea & Araque, 2002) contribute to these networks: discoveries that proceed from cerebral activity in areas that are believed to be involved in the learning and processing of information (Porto, 2004).

Finally, we must remember that the innovation of the existing ANN models toward the development of new architectures is conditioned by the need to integrate the new parameters in the learning algorithms so that they can adjust their values. New parameters that provide the PE models of the ANN with new functionalities are harder to come by than optimisations of the most frequently used algorithms that increase the output of the calculations and basically work on the computational side of the algorithm. The present study will analyse the integration of new elements in the existing networks. This approach will not excessively complicate the training process, because we apply a hybrid training method that combines the supervised and unsupervised training and whose functioning will be explained in detail further on.

In our opinion, ANN are still in a phase of development and possibly even in their initial phase. Their real potential is far from being reached, or even suspected.

Artificial Neuroglial Networks

Introduction

Many researchers have used the current potential of computers and the efficiency of computational models to elaborate "biological" computational models and reach a better understanding of the structure and behaviour of both pyramidal neurons, which are believed to be involved in learning and memory processes (LeRay, Fernández, Porto, Fuenzalida, & Buño, 2004) and astrocytes (Porto, 2004; Perea & Araque, 2002). These models have provided a better understanding of the causes and factors that are involved in the specific functioning of biological circuits. The present work will use these new insights to progress in the field of computing sciences and more concretely artificial intelligence.

We propose ANGN that include both artificial neurons and processing control elements that represent the astrocytes, and whose functioning follows the steps that were successfully applied in the construction and use of CS: design, training, testing, and execution.

Also, since the computational studies of the learning with ANN are beginning to converge toward evolutionary computation methods (Dorado, 1999), we will combine the optimisation in the modification of the weights (according to the results of the biological models) with the use of genetic algorithms (GA) in order to find the best solution for a given problem. This evolutionary technique was found to be very efficient in the training phase of the CS (Rabuñal, 1998) because it helps to adapt the CS to the optimal solution according to the inputs that enter the system and the outputs that must be produced by the system. This adaptation phenomenon takes place in the brain thanks to the plasticity of its elements and may be partly controlled by the GS; it is for this reason that we consider the GA as a part of the "artificial glia". The result of this combination is a hybrid learning method that is presented in the following sections and compared with other methods.

In this theoretic study, the design of the ANGN is oriented toward classification problems that are solved by means of simple networks (i.e., multilayer networks), although future research may lead to the design of models in more complex networks. It seems a logical approach to start the design of these new models with simple ANN, and to orientate the latest discoveries on astrocytes and pyramidal neurons in information processing toward their use in classification networks, since the control of the reinforcement or weakening of the connections in the brain is related to the adaptation or plasticity of the connections, which lead to the generation of activation ways. This process could therefore improve the classification of the patterns and their recognition by the ANGN.

The objectives of this study are the following: Analyse the modulation possibilities of the artificial synaptic activity that have not been considered so far; propose a methodology that applies these possibilities to the CS, in totally connected feedforward multilayer networks, without backpropagation and lateral connections, and conceived to solve simple classification and patterns recognition problems.

Analysis of Models and Hypotheses on Astrocytes

We know that glutamate released in the extracellular space by an astrocyte or a presynaptic neuron can affect another astrocyte, another presynaptic neuron, or a postsynaptic neuron. If the glutamate that reaches a postsynaptic neuron proceeds directly from a presynaptic neuron, the action potential (AP) takes place more rapidly and end more or less soon. If there also has been a release of glutamate by an astrocyte that was activated by the glutamate of a presynaptic neuron, more AP will take place (Pasti et al., 1997). Since the influence process controlled by the astrocyte is slower, the AP that are provoked by it will be easily detected because of their slowness. We know that the activation of the astrocytes and the communication between them through calcium signals is a slow process if we compare it to the neural activity (Araque, 2002). The same conclusion can be drawn from their effect on the synapse between two neurons, whose neurotransmitters activated the astrocyte, and which is 1,000 times slower than the propagation of the impulse in the neurons (60 s. astrocyte — 60 ms. neuron). This slowness has led to a consideration (cfr. below) on the presentation to the ANGN of each training pattern during more than one cycle or iteration. If it imitates this slowness, the ANGN will need n cycles or iterations to process each input pattern.

So far, we have not mentioned the idea that the if the astroyctes act so slowly, they are probably involved in the more complex processes of the brain, because the less developed species have less astrocytes and depend on their neurons to react rapidly to stimuli for hunting, escaping, and so forth. Since human beings usually depend less on fast reactions and more on abilities like thinking and conversing, the astrocytes may be elements that contribute to those particular processes. Research into this subject is being carried out on well-established grounds.

We also must also remember that the contribution of the astrocytes to the weights of the ANGN connections takes place according to the time factor, given the fact that they act slowly and their answers are non-linear. It would be interesting to know how astrocytes affect the CS, considering their influence on the synapses according to the activity of the neurons in the course of time. The more intense the activity of the neurons, the bigger the influence of the astrocyte on a connection, or even on another astrocyte that affects another network synapse, and so forth.

We know that there are 10 astrocytes for each neuron and that each astrocyte can affect thousands of neurons through all its ramifications. The ratio astrocytes/neurons can grow to is 50:1 in the areas with most cognitive activity.

Astrocytes have two activity levels: the neurons with their connections; the astrocytes with their connections, and their influence on the connections between neurons.

The response of the astrocyte is not “all or nothing”, but the response of the neuron can be made to be “all or nothing” according to the type of network that is being built and its activation function.

Considered Cerebral Events

Considering the functioning of the pyramidal neurons and the astrocytes (Porto, 2004), together with the existing hypotheses (LeRay et al., 2004; Perea & Araque, 2004), the main cerebral events that must be taken into account and reflected in the CS are the following: (1) Increase of the trigger potential in the postsynaptic neuron. (2) Decrease of the neurotransmitter release probability in the active synapse. (3) Decrease of the neurotransmitter release probability in other synapses, nearby or not. (4) Increase of the neurotransmitter release probability in the active synapse. (5) Increase of the neurotransmitter release probability in other synapses, nearby or not. (6) The release of neurotransmitters of an astrocyte can affect the presynaptic neuron, the postsynaptic neuron, or both. It also can open a route of influence to another synapse that is far away from the one that provoked the calcium increase prior to the release of the neurotransmitter. (7) Inhibition of inhibitory actions of presynaptic neurons in a synapse, that is, inhibitions that could take place will not do so, the synaptic transmission may take place or not depending on how the other axons in that synapse react. This point differs from point 2, in which the synaptic transmission does not take place, whereas here it may take place, regardless of the influence of the inhibitory axon that was inhibited by the astrocyte. (8) Inhibition of excitatory actions of presynaptic neurons in a synapse, that is, the excitation will not take place, the synaptic transmission may take place or not depending on the actions of the other axons in that synapse. This point also differs from point 2; the synaptic transmission may or may not take place, but this does not depend on the influence of the excitatory axon that was inhibited by the astrocyte. (9) Excitation of inhibitory actions of presynaptic neurons in a synapse, that is, the inhibition will be more powerful and the synaptic transmission may or may not occur depending on the behaviour of the other axons. (10) Excitation of the excitatory actions of presynaptic neurons in a synapse, that is, the excitation will be more powerful, the synaptic transmission may or may not occur depending on the behaviour of the other axons in that synapse.

The behaviour of neurons and astrocytes obviously makes room for certain ways and excludes others, like the eye that creates a contrast in order to distinguish between certain surrounding images.

Possibilities of the Influence of Elements and Cerebral Phenomena on CS

The analysis of the cerebral activities has opened various ways to convert CS into ANGN and as such provide them with a potential that improves their contribution to the information processing. The following paragraphs present a theoretic proposal that includes a series of modifications with an important biological basis.

The possibilities were classified according to what happens with connections between neurons, the activation value of the neurons, and combinations of both.

Connections Between Neurons

(a) **Considering each neuron individually:** The condition is that one neuron is activated. Depending on the activation function that we wish to use, we can establish in the testing system the output value that will activate the neuron, such as threshold (a value between 0 and 1), linear (value of the slope of the straight line), and so forth: If any of the neurons has been activated or not *x* times, the weight of the connections that enter into that neuron, depart from it, or both, is respectively increased or weakened with a determined percentage of its current value. This means that we reinforce the connections that reach that neuron and/or trigger in its interior the AP that provoke more powerful synapses. We can try to reinforce or weaken the connections that leave a neuron, those that enter a neuron, or both, and compare the results.

(b) **Considering two active or inactive contiguous neurons during *x* consecutive iterations:** Partly based on the postulate of Hebb (1949): Only the connection that unites these two neurons is reinforced; the aforementioned connection is weakened; the aforementioned connection, and all the connections that enter into the source neuron and/or those that leave the destination neuron, are reinforced or weakened.

(c) **Considering neurons of the same layer of an active or inactive neuron during *x* consecutive iterations:** Based on the fact that an astrocyte can influence many neurons simultaneously: The connections that enter or leave the neighbour neurons, or both types (in case that the neuron that is being managed is active during *x* iterations), are reinforced; the connections that enter or leave the neighbour neurons, or both types (in case that the neuron that is being managed is inactive during *x* iterations), are weakened.

(d) Combinations of a, b, and c.

Activation Value of the Neurons

The activation value of an artificial neuron at the present moment is influenced. This action is not a recurrence because it does not consider, for the calculation of the NET function in an artificial neuron, its own the output value or that of other artificial neurons; it considers the activation value of a neuron according to the own activity percentage or that of other neurons.

(a) **Considering each neuron individually:** The activation value of the neuron that was active or inactive during *x* consecutive iterations is increased or decreased.

(b) **Considering two active or inactive contiguous neurons during *x* consecutive iterations:** Following Hebb's postulate: The activation value of the postsynaptic

or presynaptic neuron, or both, is increased or decreased to a certain degree; the activation values of the two involved neurons and of all the contiguous neurons are increased or decreased.

(c) **Considering neighbour neurons (of the same layer) of an active or inactive neuron during *x* consecutive iterations:** Based on the fact that an astrocyte influences many neurons simultaneously, the activation value of these neighbour neurons (in case that the neuron being managed is active or inactive during *x* iterations) is increased or decreased respectively.

(d) Combinations of a, b, and c.

Combinations of Previous Cases

The resulting combinations symbolize inhibition of inhibitions, inhibition of excitations, excitation of inhibitions, excitation of excitations, of one or several neurons, of the connections that enter the neuron, of those that leave it, and so forth: When a determined neuron was inactive during *x* consecutive iterations, but had been active during *z* consecutive iterations, the value of the connections that enter or leave it, or both, does not decrease; when a neuron was inactive during *x* consecutive iterations, but had been active during *z* consecutive iterations, its associate negative outgoing connections become positive. This is an example of excitation of inhibitory synapses; when a neuron was active during *x* consecutive iterations, but had been inactive during *z* consecutive iterations, the associated connections are not reinforced; when a neuron was active during *x* consecutive iterations, but had been inactive during *z* consecutive iterations, its associate positive outgoing connections become 0. This is an example of inhibition of excitatory synapses.

Functioning Proposal of the ANGN

The construction and functioning of an ANGN follows all the stages of a CS, starting with the design of the network architecture, followed by the training, testing, and execution phases.

Design Phase

For reasons of simplification, the design is based on feedforward multilayer architectures that are totally connected, without backpropagation or lateral connections, and oriented toward the classification and recognition of patterns.

Training Phase

We have designed a hybrid training method that combines non-supervised learning (first phase) with the supervised learning that uses the evolutionary technique of GA (second phase).

Since the GA requires individuals, the first phase creates a set of individuals to work with. Each individual of the GA consists of as many values as there are connection weights in the ANGN, and each arbitrary set of values of all the weights constitutes a different individual.

The first phase consists of a non-supervised learning based on the behaviour of the cerebral cells that were modeled by the NEURON simulation environment (Hines, 1994) in the works of Porto (2004), Araque (2002), and LeRay et al. (2004). The functioning of the network with all its individuals is analysed. Each individual (i.e., the weights of the connections) is modified as each training pattern passes on to the network, according to how the activity of the neurons has been during the passage of that pattern. For each individual, each pattern or input example of the training set is presented to the network during a given number of times or iterations. These iterations represent the slowness of the astrocytic influence (cfr. above), and constitute a cycle of the pattern. The number of iterations can be established for any cycle. During each iteration of the cycle, the connections are modified according to the previously explained rules (cfr. above), which generally depend on the activity of the neurons. Once the cycle of the pattern is finished, we calculate the error of the network for that pattern to find the difference between the obtained and the desired output. We store the error of the network for each pattern. Afterwards, when all the training patterns have been passed on to the network, we calculate the mean square error (MSE) for that individual, since at the start of a pattern cycle, the individual that is applied to the network is once again the first of the used set of individuals. We have opted for the MSE because it gives a relative measure to the examples that are fed to the network to compare the error between different architectures and training games. Also, the square in the numerator favours the cases of individuals for which the output of the network is close to the optimal values for all the examples. The process is the same for all the individuals. This phase constitutes a non-supervised training, because the modifications of the connections' weights do not consider the error of the output, but take place at any time according to the activation frequency of each neuron, simulating reinforcements and inhibitions that in the brain would possibly be provoked by astrocytes (Perea & Araque, 2004) or depolarising ion streams (LeRay et al., 2004).

The second and last phase of the training is the supervised training phase. It consists in applying GA to the individuals according to the MSE made by the network with each of the individuals and stored during the first training phase (Rabuñal, 1998). Once the MSE of all the individuals are stored, the GA in a second phase carries out the corresponding cross-overs and mutations and selects the new individuals with which the first and second phases will be repeated until the least possible error, and preferably no error, is obtained. The second phase is considered a supervised training because the GA takes into account the error made by the network to select the individuals that will be

mutated and crossed over, that is, it makes the changes in the weights according to that error.

The GA training system applies the GA specifications formulated by Holland (1975).

Testing and Execution Phase

The training of the ANGN has provided us with the individual whose weights allow us to obtain the smallest error in the output. During the present phase, we use this individual to check whether the output obtained by the model is correct, that is, whether the generalisation capacity of the ANGN is correct with input patterns that differ from those used during the training stage, and to prepare the ANGN for its subsequent use.

In this phase, and in the course of all the subsequent executions, the network activity control elements that represent pyramidal neurons and astrocytes — which intervene during the non- supervised training phase — remain active. These new incorporated elements will therefore be a part of the model in all its stages and participate directly in the information processing, just like the artificial neurons. The input patterns will present themselves during the iterations that were determined in the training phase and hereby allow the new elements to carry out their activity.

Comparison Between the Proposed Learning Method and the Existing Methods

This section compares the proposed learning method with several methods that are usually applied in CS and present certain similarities; our purpose is to comment on the existing differences and the advantages of this new proposal.

The first difference resides in the modification moment of the weights. In the backpropagation method and other methods that use supervised learning rules, the weights are not modified each time a pattern is passed on: Once all the patterns of the training set are passed on, the MSE is calculated and on the basis of that error the weights are modified once. Afterwards, the whole training set is passed on again, the MSE is calculated again and the relevant modifications in the weights are carried out again. This continues until the error is as small as possible and the network converges. The proposed method, however, modifies the weights during each step of the cycle, regardless of the observed error and according to the activations that have taken place at each moment. This situation may cause a slight delay in the functioning of the CS, but it emulates the cerebral reality with more precision.

With respect to the criteria that must be followed to modify the weights, we copy the functionalities of the modeled cerebral elements. First, this procedure presents certain similarities with the modifications of the non-supervised learning method of Kohonen, except for the fact that in our method, which is tested on classification problems, there is no competitivity. Kohonen presents competitive networks that classify input patterns

into groups and uses the modification of the weights of the PE with more or less output value. Second, since our method takes into account all the activations of the artificial neurons, we believe it is important to comment on the difference with the method used by the networks based on delays. In those networks, the PE possess memories that store the values of previous activations in order to operate with them at the present moment. During the first phase of the proposed hybrid method, we count how often an artificial neuron has been activated, not what value it has obtained.

This method reinforces or weakens certain connections. According to the applied neurobiological rule, connections before or after the PE can be reinforced or weakened. By taking into account the modification of previous connections, we observe what could be a resemblance to a recurrence which is partial, because only certain connections are reinforced or inhibited under specific conditions. However, since the new control elements, outside the PE, influence the weights regardless of the current activation value of the PE, we can conclude that this is not a case of recurrence, as in partially or totally recurrent networks, but a case of "influence".

This may imply not only that the new element modifies previous connections, but also that the previous artificial neurons may have modified the magnitude of the corresponding synapse, as has been observed during in vitro experiments. This situation, which is based on the postulate of Hebb (1949), will allow the incorporation of phenomena that are modeled in synaptic potentiation (LeRay et al., 2004; Porto, 2004). It also suggests the future use of the Hebb rule, used in non-supervised learning, to make these weights' variations, combining this use with GA to continue considering the advantages of a hybrid method for the classification of multilayer networks.

Another important aspect that distinguishes the ANGN does not concern the training phase, but rather the evaluation and execution phase. When the network is used in the execution phase, the control actions of the new incorporated elements are maintained. This means that each pattern must be passed on *n* times, *n* being the number of iterations chosen from the pattern cycle. The ANGN needs *n* cycles to process each input pattern.

Future Developments

We have already begun to implement this theoretical proposal on CS, testing each of the presented possibilities and comparing their results with those of ANN trained with GA. We are considering the solution of a simple problem with an ANGN: We simulate an electronic device known as multiplexor (MUX), with four inputs and one output, by means of an ANGN with a totally connected and feedforward multilayer architecture, without backpropagation and lateral connections. The results are satisfactory and the experiments are being continued.

References

Araque, A. (2002, September). Conversación personal con A. Araque. Instituto Cajal (CSIC). Madrid.

Araque, A., Carmignoto, G., & Haydon, P. G. (2001). Dynamic signaling between astrocytes and neurons. *Annu. Rev. Physiol, 63*, 795-813.

Araque, A., Púrpura, V., Sanzgiri, R., & Haydon, P. G. (1999). Tripartite synapses: Glia, the unacknowledged partner. *Trends in Neuroscience*, *22*(5).

Dorado, J. (1999). *Modelo de un sistema para la selección automática en dominios complejos, con una estrategia cooperativa, de conjuntos de entrenamiento y arquitecturas ideales de redes de neuronas artificiales utilizando algoritmos genéticos*. Doctoral thesis. Facultad de Informática. Universidade da Coruña.

Haydon, P. G. (2001). Glia: Listening and talking to the synapse. *Nat. Rev. Neurosci., 2,* 185-193.

Haydon, P. G., & Araque, A. (2002). Astrocytes as modulators of synaptic transmission. In A. Volterra, P. Magistretti, & Haydon (Eds.), *Tripartite synapses: Synaptic transmission with glia* (pp. 185-198). Oxford University Press.

Hebb, D. O. (1949). *The organization of behaviour*. New York: J. Wiley.

Hines, M. (1994). The NEURON simulation program. In J. Skrzypek & Norwell (Ed.), *Neural network simulation environments* (pp. 147-163). MA: Kluwer.

Holland, J. H. (1975). *Adaptation in natural and artificial systems*. Ann Arbor: University of Michigan Press.

Hopfield, J. (1989). *Neural networks: Algorithms and microchardware*. Piscataway, NJ: Institute of Electrical and Electronics Engineers.

Kimelberg, H. K. (1983). Primary astrocyte culture. A key to astrocyte function. *Cellular and Molecular Neurobiology, 3*(3)1-16.

Kimelberg, H. K. (1989). Neuroglia. *Rev. Investigación y Ciencia, 153,* 44-55.

Kuwada, J. Y. (1986). Cell recognition by neuronal growth cones in a simple vertebrate embryo. *Science, 233,* 740-746.

Largo, C., Cuevas, P., Somjen, G. G., Martin del Rio, R., & Herreras, O. (1996). The effect of depressing glial function in rat brain in situ on ion homeostasis, synaptic transmission, and neuron survival. *Journal Neurosci., 16,* 1219-1229.

LeRay, D., Fernández, D., Porto, A., Fuenzalida, M., & Buño, W. (2004). Heterosynaptic metaplastic regulation of synaptic efficacy in CA1 pyramidal neurons of rat hippocampus. *Hippocampus*.

LeRoux, P. D., & Reh, T. A. (1986). Regional differences in glial derived factors that promote dendritic outgrowth from mouse cortical neurons in vitro. *Journal Neurosci., 14,* 4639-4655.

Lorenz, K. (1986). *Fundamentos de la etología: Estudio comparado de las conductas*. Barcelona: Paidós.

Martín, E. D., & Araque, A. (2005). *Astrocytes and the biological neural networks*. Artificial Neural Networks in Real-Life Applications (pp. 22-46). Hershey, PA: Idea Group Publishing.

Mauch, D. H., Nagler, K., Schumacher, S., Goritz, C., Muller, E. C., Otto, A., & Pfrieger, F. W. (2001). CNS synaptogenesis promoted by glia- derived cholesterol. *Science, 294,* 1354-1357.

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics, 5,* 115-133.

Mennerick, S., & Zorumski, C.F. (1994). Glial contribution to excitatory neurotransmission in cultured hippocampal cells. *Nature, 368,* 59-62.

Noremberg, M.D., Hertz, L., & Schousboe, A. (1988). *Biochemical pathology of astrocytes*. Alan R. Liss Inc.

Pasti, L., Volterra, A., Pozzan, R., & Carmignoto, G. (1997). Intracellular calcium oscillations in astrocytes: A highly plastic, bidirectional form of communication between neurons and astrocytes in situ. *Journal of Neuroscience, 17*, 7817-7830.

Pazos, A., & Col. (1991). *Estructura, dinámica y aplicaciones de las Redes de Neuronas Artificiales*. Centro de Estudios Ramón Areces, S.A. España.

Perea, G., & Araque, A. (2002). Communication between astrocytes and neurons: A complex language. *Journal of Physiology*. Paris: Elsevier Science.

Perea, G., & Araque, A. (2004). Properties of synaptically evoked astrocyte calcium signal reveal synaptic information processing by astrocytes. *Journal of Neuroscience*, in press.

Pfrieger, F.W., & Barres, B.A. (1997). Synaptic efficacy enhanced by glial cells in vitro. *Science, 277,* 1684-1687.

Porto, A. (2004). *Modelos computacionales para optimizar el aprendizaje y el procesamiento de la información en sistemas adaptativos: Redes neurogliales artificiales (RR.NG.AA.)*. Tesis Doctoral. Universidade da Coruña. A Coruña.

Rabuñal, J. (1998). *Entrenamiento de redes de neuronas artificiales con algoritmos genéticos*. Tesis de Licenciatura. Dep. Computación. Facultad de Informática. Universidade da Coruña.

Raff, M.C., Barres, B.A., Burne, J.F., Coles, H.S., Ishizaki & Jacobson, M.D. (1993). Programmed cell death and the control of cell survival: Lessons from the nervous system. *Science, 262*, 695-700.

Rakic, P. (1978). Neural migration and contact guidance in the primate telencephalon. *Postgraduated Medical Journal, 54*, 25-40.

Rakic, P. (1990). Principies of neuronal cell migration. *Experientia, 46,* 882-891.

Ramón y Cajal, S. (1904). *Textura del sistema nervioso del hombre y los vertebrados. Tomo II.* Madrid.

Ramón y Cajal, S. (1911). *Histologie du sisteme nerveux de I'homme et des vertebres*. Maloine, Paris.

Sanai, N., Tramontin, A., Quiñones, A., Barbaro, N., & Gupta, N. (2004). Unique astrocyte ribbon in adult human brain contains neural ítem cells but lacks chain migration. *Nature 427,* 740-744.

Smith-Thier, J. (1975). *Fisiopatología: Principios biológicos de la enfermedad* (2nd ed). Panamericana. México.

Verkhratsky, A., Orkand, R.K., & Kettenmann, H. (1998). Glial calcium: Homeostasis and signaling function. *Physiol Rev, 78,* 99-141.

Von Neumann, J. (1956). Probabilistic logics and the synthesis of reliable organisms from unreliable components. In C.E. Shannon & J. McCarthy (Eds.), *Automata studies* (pp. 43-98). Princeton University Press.

Von Neumann, J. (1956). *The Computer and the brain*. Yale University Press.

Wiener, N. (1985). *Cibernética*. Tusqets editores.

Chapter II

Astrocytes and the Biological Neural Networks

Eduardo D. Martín, University of Castilla - La Mancha, Spain

Alfonso Araque, Instituto Cajal, CSIC, Spain

Abstract

Artificial neural networks are a neurobiologically inspired paradigm that emulates the functioning of the brain. They are based on neuronal function, because neurons are recognized as the cellular elements responsible for the brain information processing. However, recent studies have demonstrated that astrocytes can signal to other astrocytes and can communicate reciprocally with neurons, which suggests a more active role of astrocytes in the nervous system physiology and fundamental brain functions. This novel vision of the glial role on brain function calls for a reexamination of our current vision of artificial neural networks, which should be expanded to consider artificial neuroglial networks. The neuroglial network concept has not been yet applied to the computational and artificial intelligent sciences. However, the implementation of artificial neuroglial networks by incorporating glial cells as part of artificial neural networks may be as fruitful and successful for artificial networks as they have been for biological networks.

Introduction

Artificial neural networks — a neurobiologically inspired paradigm that emulates the functioning of the brain — are based on the way we believe that neurons work, because they are recognized as the cellular elements responsible for the brain information processing. Two main cell types exist in the brain: neurons and glia. Among the four main subtypes of glia, astrocytes are the most common cells in the central nervous system (CNS). Astrocyte function has long been thought to be merely supportive of neural function. However, recent studies have demonstrated that astrocytes can signal to other astrocytes — forming a new type of cellular network in the brain — and can communicate bidirectionally with neurons, which suggests a more active role of astrocytes in fundamental brain functions, regulating neuronal excitability and synaptic transmission (for a review see Araque, Carmignoto, & Haydon, 2001). Based on these new findings, glia is now considered as an active partner of the synapse, dynamically regulating synaptic information transfer as well as neuronal information processing. This novel vision of the glial role on brain function calls for a reexamination of our current vision of artificial neural networks, which should be expanded to consider glial cells to create artificial neuroglial networks.

In some areas of the nervous system, glial cells outnumber nerve cells 10 to 1. Glia (from the Greek, meaning glue) is important in providing a homeostatic environment to the nerve cells as well as being involved in other functions. There are three main types of glial cells in the central nervous system: astrocytes, oligodendrocytes, and microglia. Astrocytes have many processes that branch out in a starlike formation. Functions of astrocytes include: structural support for nerve cells; proliferation and repair following injury to nerves; participation in metabolic pathways that modulate extracellular concentration of ions, transmitters, and metabolites involved in functions of nerve cells and synapses. Oligodendrocytes are mainly responsible for the formation of myelin around axons in the central nervous system. These myelin sheaths play an important role in the improvement of the nerve conduction properties. While oligodendrocytes are specifically present in the central nervous system, the myelin is formed by Schwann cells in the peripheral nervous system. The third type of glial cells, microglia, are smaller cells present throughout the central nervous system that function as immune system cells in the CNS.

The astroglial cells, or astrocytes, are connected through gap junctions forming a relatively large electrically coupled syncytium. The single cells have long processes, and some of them establish contacts with blood vessels, forming part of the blood-brain barrier. Other processes extend toward and encapsulate synapses, especially glutamatergic synapses (i.e., excitatory synapses that release the neurotransmitter glutamate) and also the varicosities, from which other neurotransmitters such as monoamines are released. Neuronal cell bodies, neuronal processes, and the brain surface are also encapsulated by astroglial processes.

The astroglial cell mass constitutes a prominent part of the total brain cell number and volume (Peters, Palay, & Webster, 1991). More than 100 years ago, Virchow proposed that these cells have a metabolic and structural supportive role for neurons. Since then and until the last 15 to 20 years, this idea of astrocytes as simple supportive and passive cells has been maintained. Very little attention was paid to the astroglial cells for decades,

mostly because the absence of conspicuous physiological function in the electrophysiological behaviour of the nervous system. Indeed, while neurons were relatively easy to identify using electrophysiological techniques due to their ability to fire action potentials, astrocytes can be depolarized but no action potential or other significant active electrical behaviour can be elicited.

In the last years, it has been shown that, in addition to the relevant functions in brain homeostasis (e.g., supplying energy to neurons, controlling the concentration of ions and neurotransmitters in the extracellular space, and synthesizing and releasing neurotrophic factors), astrocytes have the capacity to monitor synaptic activity, to sense the composition of the extracellular space and the blood serum, to integrate the information obtained, and to influence neuronal activity and synaptic transmission by regulating the extracellular concentration of neurotransmitters and by releasing neuroactive substances (called gliotransmitters) (for reviews see Araque et al., 2001; Volterra, Magistretti, & Haydon, 2002).

In this chapter, we will provide an overview of our current knowledge of astroglial physiology and their impact in the neuronal physiology, and we will discuss the relevance of these new findings on the artificial neuronal network concept. We will first present a general outline of the neural network function, summarizing the well-known biological concepts and cellular mechanisms of neuronal function and synaptic physiology. We will next briefly describe astroglial cell physiology, discussing the Ca^{2+}-based astroglial excitability, the existence of astrocyte-neuron communication and its relevant physiological consequences for the functioning of the nervous system. Finally, we will suggest a reassessment of our current vision of artificial neural networks proposing the necessity and convenience of considering artificial neuroglial networks.

Biological Neural Networks

In the following paragraphs we will briefly present some key concepts on biological neural networks, and specifically on neuronal and synaptic physiology. The interested reader may find a broader description of the topics in several excellent textbooks in the field (e.g., Kandel, Schwartz, & Jessell, 1991; Nicholls, Martin, & Wallace, 1992; Shepherd, 1994; Kuno, 1995).

The nervous system can be considered as an organized assembly of cells interconnected through synapses. Brain functions emerge out of the highly complex and dynamic organization of the central nervous system. At the highest levels of nervous system organization are systems and pathways. Pathways are sequences of connections through several neuronal centers, that is, functional and anatomical groups of neurons. Their function is to transmit information from the periphery into the central nervous system (as in the sensory pathway), or from center to periphery (as in the motor pathway). Central systems, or distributed systems, are sets of connections between a number of centers which together mediate functions necessary for the coordinated behaviour of the whole organism. There is a gradual progression from pathways that perform specific

tasks of sensory processing and motor control to central systems that deal with the global aspects of behaviour.

Nerve centers, or nuclei, and local circuits are populations of neurons and glial cells that are connected together in networks. These populations receive information from other centers, perform some specific processing operations, and send the output to other centers or to effectors organs (such as muscles or glands). This level of organization is equivalent to the interconnections among processing elements in artificial neural networks.

Neuron Structure and Function

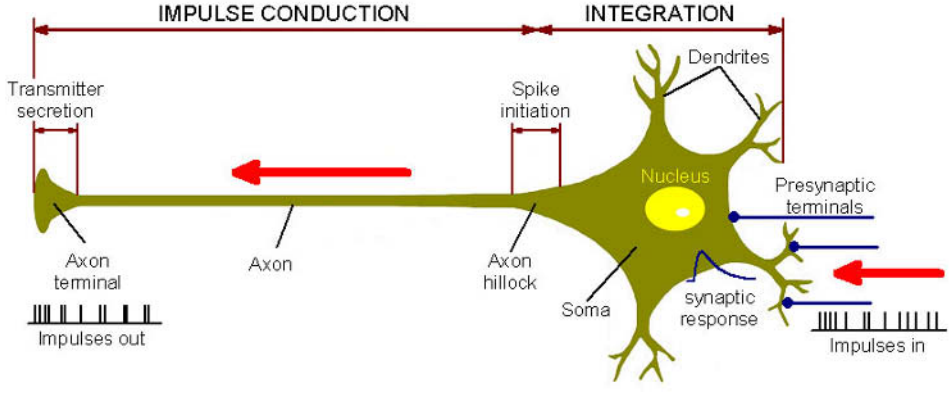
The neuron, which may be considered as the fundamental anatomical unit of the nervous system, is a specialized cell that receives signals from other cells through its soma and dendrites, integrates them (i.e., elaborate a response which is a function of the multiple incoming signals), and sends output signals to other cells through its axon. Figure 1 shows these basic elements.

Like any other cell, the neuron is enclosed by a lipid bi-layer membrane, has a nucleus with genetic material, and has the usual complement of cellular organelles such as mitochondrian, endoplasmic reticulum, lysosomes, and so forth. However, neurons have particular morphologies, depending on their role and position in the nervous system, that follow a similar morphological pattern: a cell body from where many branches extend. As a neuron grows, only one of these branches becomes an axon, while the rest become dendrites. In a prototypic neuron, each region has distinctive signaling functions: (1) soma, which is the metabolic center of the neuron; (2) dendrites, which correspond to the receptive area of the neuron; (3) the axon, which is the neuronal conducting unit that conveys the information to relatively distant cells; (4) presynaptic terminals of the axon, which are the transmitting elements of the neuron. Through these presynaptic terminals, one neuron contacts and transmits information to the receptive surfaces of another neuron, muscle, or gland cell. This point of contact is known as the synapse.

There are two main types of neurons that can be identified by the scope of their axonal connections. The so-called projection, principal, or relay neurons typically have long axons that make distant connections. Examples of these cells are the sensory and motor neurons that go outside the brain, and neurons in the brain that reach different nervous system centers. The other main type of neuron has a much shorter axon and only connects to other neurons within a center. These neurons are called intrinsic neurons or interneurons.

Like any other cell, neurons have what is known as a membrane potential, that is, an electric potential difference between the intracellular and extracellular compartments. This membrane potential is based on the fact that there is normally about 10 times more sodium in the extracellular fluid than in the cytoplasm and about 40 times more potassium in the cytoplasm than in the extracellular fluid. Neuronal membrane is endowed with relatively selective ionic channels that allow some specific ions to cross the membrane. In resting conditions, sodium ionic channels are closed, thus preventing sodium ions

Figure 1. Schematic drawing of a prototypical neuron. The soma (cell body) and dendrites represent the site of input reception, integration, and coordination of information signals coming from pre-synaptic nerve terminals. Information propagates from the dendrites and the cell body to the axon hillock. If the sum of the synaptic potentials conducted to the axon hillock is greater than the depolarization threshold for firing action potentials, one or more output action potentials occur. The axon is responsible for the transmission of the action potential. The CNS switches between amplitude modulation and frequency modulation. While the action potential is propagated regeneratively in an all-or-non form in the axonal region of the nerve, the frequency of action potentials generated at the axon hillock is proportional to the magnitude of the net synaptic response at that location. Information about the sum of the electrical activity reaching the axon hillock is represented in the frequency of these action potentials.



from "leaking" in. Some potassium channels, however, are open, and potassium is allowed to "leak" out of the cytoplasm, following its concentration gradient. In this situation, the cytoplasm becomes more negative than the extracellular fluid and eventually becomes negative enough to start attracting potassium ions back into the cell. The point at which this attraction balances with potassium following its concentration results in the resting membrane potential, which is close to the equilibrium potential of the electrochemical gradient of potassium ions, that is, around -70 mV.

Unlike many other cells, however, neurons have excitable membranes because some of the selective ionic channels present in the membrane are voltage-gated, that is, are opened or closed depending on the membrane potential. Signals from dendrites, the "input" region of a neuron, cause the opening of sodium channels. Sodium ions, following their concentration gradient, cross the membrane, depolarizing the membrane potential. Once enough of a depolarization threshold is reached, the voltage difference

between the cytoplasm and extracellular fluid suddenly reverses. This abrupt reversal is called action potential and may be propagated down the length of the axon to the axon terminal. The action potential represents the biophysical substrate that encodes the neuronal information.

Synaptic Physiology

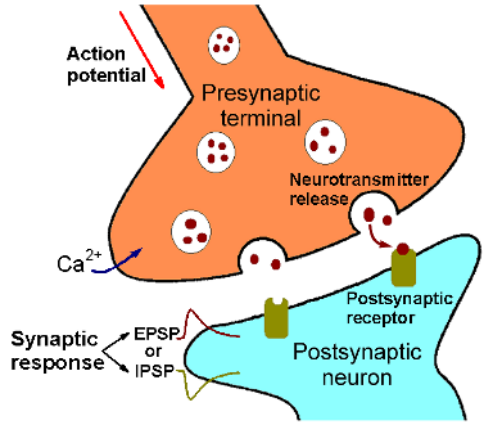
The synapse is the basic input-output unit for transmission of information between neurons. There are two types of synapses: electrical and chemical. Electrical synapses are able to transmit nerve signals very fast and directly across a very small gap between two nerve cells, which are connected through special structures called gap junctions. Gap junctions are membrane specializations that provide a pathway for intercellular exchange of small molecules and ions between cells. Gap junctional communication is thought to facilitate coordination of cellular activity in several physiological processes such as tissue homeostasis, neuronal synchronization, and cardiac and smooth muscle contraction and development (Bennett et al., 1991). Gap junctional channels are formed by members of closely related membrane proteins called connexins. Each channel is formed by two sets of six connexin molecules spanning the membrane of each connecting cell, forming the connexon, the functional gap junction channel. While in electrical synapses the electrical signal pass from one neuron to another, in chemical synapses the communication signal relies on a chemical substance called neurotransmitter. Although electrical synapses are extremely important for the nervous system function, we will focus our discussion in this chapter on chemical synapses, which are the most abundant and well studied.

When the action potential reaches the axon terminal of the presynaptic neuron, voltage-gated Ca^{2+} channels are opened, allowing the influx of Ca^{2+} into the terminal. The subsequent increase of the intracellular Ca^{2+} concentration causes the fusion of vesicles that contain neurotransmitters. These neurotransmitters released from synaptic terminals diffuse across the synaptic cleft and bind to receptors on the surface of the postsynaptic neuron. This binding leads to membrane permeability changes that will either stimulate or inhibit the firing of the second neuron (Figure 2). There is a considerable amount of different molecules that may act as neurotransmitters and they are classified based on their function as excitatory or inhibitory. On the other hand, according to the mechanism of action, neurotransmitter receptors are classified as ionotropic or metabotropic. Binding of neurotransmitters to metabotropic receptors leads to activation of specific proteins and production of intracellular second messengers that can ultimately affect the behavior of the neuron. Ionotropic receptors are also called ligand-gated channels because they are transmembrane proteins that form ionic channels that undergo conformational changes upon binding of neurotransmitters, that is, these ionic channels are opened by neurotransmitters. These ionotropic channels are specific in the type of ion that they allow to pass through. If a receptor allows negative ions, such as chlorine (Cl^-), to flow into the cytoplasm, then this has an inhibitory effect as it moves the membrane potential away from the threshold to fire action potentials. On the other hand, if a receptor allows a positive ion to enter the cytoplasm, then this has

an excitatory effect as it depolarizes the local membrane potential, bringing it closer to the action potential threshold.

Integration of synaptic information by neurons involves the complex interactions between ionic currents and synaptic configurations that lead to the global behavior of the neuron. At individual synapses, an excitatory postsynaptic potential (EPSP) depolarizes the membrane potential. Conversely, an inhibitory postsynaptic potential (IPSP) usually hyperpolarizes the membrane potential (Figure 2). These actions are not discrete like signals in an electric circuit or numbers in an equation. Like the extracellular fluid, the inside of the neuron is a continuous milieu, which diffuses ions within it. When there is an EPSP at a dendrite, positive ions diffuse not only into the local region of the cytoplasm, but also away from it in both directions (likewise for IPSPs). This ion flow can affect other regions. Activation of an excitatory synapse in a dendrite results in an inward current that elicits a depolarization that spreads along the proximal dendrite, the soma, and the axon hillock (the initial segment of the axon, where the action potential is usually generated). However, when an inhibitory synapse located between the excitatory synapse and the cell body is activated, the resulting outward current tends to hyperpolarize the membrane and, when it occurred concomitantly with an EPSP, it reduces the

Figure 2. Communication between neurons takes place via a specialized structure called synapse. Action potentials in the presynaptic neuron can either generate or inhibit action potentials in the postsynaptic neuron via synaptic potentials. The cleft between the neurons is bridged either electrically (an electrical synapse) or chemically via neurotransmitters (a chemical synapse). In the distal end of the nerve fiber there are voltage-gated calcium channels along with voltage-gated sodium and potassium channels. Action potentials in the nerve terminals cause an influx of Ca^{2+} due to the opening of calcium channels. The resultant increase in intracellular calcium is responsible for the release of neurotransmitters that transfer the information to the postsynaptic neuron during synaptic transmission.



EPSP-evoked depolarization of the cell body and the axon hillock. Therefore, opposing EPSPs and IPSPs "compete" for the control of the membrane potential in a non-linear fashion. The integration of all excitatory and inhibitory synapses, that is, their non-linear summation, may finally result in the eventual generation of an action potential that propagates along the axon.

On average, every one of the hundreds of millions of neurons estimated to be present in the rat brain receives about 6,000 synaptic contacts that can be either excitatory or inhibitory. Therefore, the next level of nervous system organization applies to the synaptic organization that involves patterns of interconnecting synapses. The simplest of these patterns is formed by two or more synapses situated near each other and oriented in the same direction. A simple synapse may be defined as one in which an axon terminal of neuron is paired with the dendrite of another (axodendritic), or its cell body (axosomatic). However, most parts of the system are organized in more complex patterns. For example, dendrodendritic synapses are those in which one dendrite signals uni-directionally with another dendrite. Reciprocal synapses are those in which signaling is bi-directional for dendrodendritic synapses. Axoaxonic synapses are formed when the axon terminal of one neuron contacts with the initial segment or axon of the postsynaptic neuron. Serial synapses are sequences of uni-directional signaling from more than two processes. Examples are axodendrodendritic sequences, in which an axon signals a dendrite which then signals another dendrite, and axoaxodendritic sequences in which an axon signals another axon that signals a dendrite. These complex arrangements of synapses, which constitute the microcircuit level of organization, presumably operate in a unified manner in order to process information (Kuno, 1995; Kandel et al., 2000). In summary, the nervous system is structured in a high degree of both morphological and functional complexity present in all the successive levels of organization, from molecular (e.g., specific receptor expression and distribution) to cellular (e.g., morphological neuronal characteristics and physiological phenotype of synapses) and supracellular levels (e.g., microcircuit functioning and nuclei connections).

Neuroglial Networks

Neurons have long been known to signal to each other by various kinds of transmitter substances. Recent data have revealed that glial activity is probably determined by neuronal activity and that glial cells have the capacity to signal not only to each other, but also back to neurons. Astrocytes are the most abundant cells in the brain, constituting over 50% of the total cell number in the cerebral cortex (Peters et al., 1991). Their relative number is especially high in humans and other highly developed mammals. They are recognized as star-shaped cells whose processes extend into the surrounding neuropil, and they are extensively coupled in a cellular network. They also are intimately associated with neurons both structurally and functionally, indicating important roles for the astroglial cells in brain function. Indeed, they are involved in neuronal guidance during development (Hatten & Mason, 1990; Ullian, Sapperstein, Christopherson, & Barres, 2001), neuronal survival (Banker, 1980; Pfrieger & Barres, 1995; Tsacopoulos &

Magistretti, 1996) and differentiation (Takeshima, Shimoda, Sauve, & Commissiong, 1994), neuronal guidance (Bastiani & Goodman, 1986; Kuwada, 1986; Rakic, 1990), neurite outgrowth (e.g., Johnson et al., 1989; Le Roux & Reh, 1994; Noble et al., 1984), synaptogenesis (Slezak & Pfrieger, 2003; Ullian et al., 2001; Ullian, Christopherson, & Barres, 2004), and control of the local extracellular concentration of ions (Largo, Cuevas, Somjen, Martin del Rio, & Herreras, 1996; Orkand, Nicholls, & Kuffler, 1966) and neurotransmitters (Largo et al., 1996; Mennerick & Zorumski, 1994; Szatkowski, Barbour, & Attwell, 1990; Walz, 1989; Vernadakis, 1996). Morphologically, astrocytes are closely associated with neurons. They can encase synaptic terminals, make extensive contact with endothelial cells from capillaries, and are interconnected through gap junctions (Ventura & Harris, 1999). Many of the ionic channels found in neurons (voltage-, ligand-, and mechanically-activated channels) are also present in astrocytes (Barres, Chun, & Corey, 1990; Sontheimer, 1994). Although the functional significance of these ionic channels is not fully understood, data indicate that they have important functions. For example, ionic channels constitute a prerequisite for astroglial extracellular buffering, responsiveness to synaptic activity, and intercellular communication (see next section).

Astrocytes Support and Monitor Neuronal Activity

Many astrocytes have processes that contact the surfaces of blood vessels with their "end-feet," forming part of the blood-brain barrier together with the capillary endothelium (Nedergaard, Ransom, & Goldman, 2003). Other processes extend to the neuronal cell bodies, and the astrocytes thereby serve as a connecting link between the neurons and the blood circulation (Figure 3). In addition to these vital extensions, astrocyte processes also reach the ependymal cells, connecting them with the cerebral ventricular system, while other processes extend to the brain surface to form expansions that constitute the glial limiting membrane (Peters et al., 1991). Furthermore, other processes closely approach the synaptic regions and ensheathe the synaptic clefts (Ventura & Harris, 1999). In view of the fact that astrocytes express a variety of ion channels, a large number of neurotransmitter receptors, and several active release and uptake mechanisms for neuroactive substances, this close proximity of astrocytes to synapses enables them to communicate with neurons.

One of the first recognized roles of astrocytes is the metabolic support for neurons. Glucose is taken up into the astroglia from the blood, transformed to glycogen, and stored. When required, and especially upon demand during neuronal activity, the astrocytes release lactate and other energy-rich compounds to be used as metabolic fuel for the neurons (Figure 3). Neuron-glial interaction is vital to the energy metabolism of the brain, and accumulating data indicate that cerebral energy metabolism is under detailed regulation of neurotransmitters acting on these glial cells (Hamprecht & Dringen, 1995; Magistretti et al., 1994; Tsacopoulos & Magistretti, 1996; Bernardinelli, Magistretti, & Chatton, 2004; Pellerin & Magistretti, 2004). Metabolic inhibition of glial cells reduces and modulates synaptic transmission, and the astrocytes have a verified neuroprotective effect under conditions of metabolic stress (Takuma, Baba, & Matsuda, 2004), probably because their high tolerance for cellular stress such as hypoxia and hypoglycemia is unique among the cells of the brain. A considerable amount of the

energy generated is, however, used within the astrocyte itself to operate the membrane pumps necessary for vital brain functions. Ion buffering, glutamate uptake, and cell-volume regulation are the highest metabolic priorities. Energy status also plays a role in regulating intercellular communication (Hertz & Peng, 1992; Swanson & Choi, 1993; Tsacopoulos & Magistretti, 1996).

Glutamate (Glu) is the most abundant excitatory neurotransmitter in the central nervous system. Glutamatergic synaptic transmission is largely terminated by high affinity, sodium-dependent transport of glutamate from the synaptic cleft (e.g., Bergles, Diamond, & Jahr, 1999). Furthermore, Glu transporters also keep the extracellular Glu concentration at low levels (1 to 3 mM), which is crucial because the well-known excitotoxic effects of high levels of the ambient Glu (Choi & Rothman, 1990) (Figure 3). Both astroglial cells and neurons possess similar, although not identical, Glu uptake carriers on their plasma membranes. However, the capacity of neurons to take up Glu seems to be lower than that of glia, even though the anatomy of the synaptic cleft might favor a neuronal removal of Glu after its release from the presynaptic region. The uptake capacity of Glu by astroglia, however, is considered to be sufficient to account for all Glu released by neurons, although the relative contribution of neurons and glial cells to Glu uptake may vary between different brain areas (Schousboe, 1981; Bergles et al., 1999).

Intercellular Astrocyte Communication

Recently, it has become evident that some cell types, including astroglial cells, communicate intercellularly via complex spatio-temporal calcium fluxes (Cornell-Bell, Finkbeiner, Cooper, & Smith, 1990; Charles & Giaume, 2002). Intercellular Ca^{2+} waves can be initiated in astrocytes and endothelial cells by mechanical, chemical, or electrical stimuli. Stimulation results in an increase in the intracellular Ca^{2+} concentration, which is propagated from cell to cell (Figure 4) (Verkhartsky, Solovyeva, & Toescu, 2002; Charles & Giaume, 2002). Different stimuli, including several neurotransmitters, may induce intracellular Ca^{2+} increases that can propagate as Ca^{2+} waves between astrocytes in cultured cells as well as organotypic and acute brain slices (Cornell-Bell et al., 1990; Dani, Chernjavsky, & Smith, 1992; Schipke, Boucsein, Ohlemeyer, Kirchhoff, & Kettenmann, 2002; Sul, Orosz, Givens, & Haydon, 2004). These astrocyte Ca^{2+} waves may extend for relatively long distances (<500 μm) at relatively low speed (~14 μm/s) (Schipke et al., 2002).

The cellular mechanisms underlying astrocytic Ca^{2+} wave propagation have been the focus of several studies due to their possible important functions as a novel form of cellular communication in the nervous system. Astrocytes express a wide range of neurotransmitter receptors, and many of them are metabotropic. The activation of these receptors elicit intracellular Ca^{2+} elevations that result from the release of Ca^{2+} from intracellular stores, which are activated by elevations of the second messenger inositol-1,4,5-trisphosphate (IP_3). This messenger is generated as a consequence of the activity of phospholipase C, which in turn is activated by certain G-protein-coupled metabotropic receptors. Calcium waves were first thought to spread as a result of gap junction-mediated diffusion of IP_3 between astrocytes (Sneyd, Charles, & Sanderson, 1994) (Figure 4). Later studies indicated that in addition to diffusion of IP_3, other mechanisms

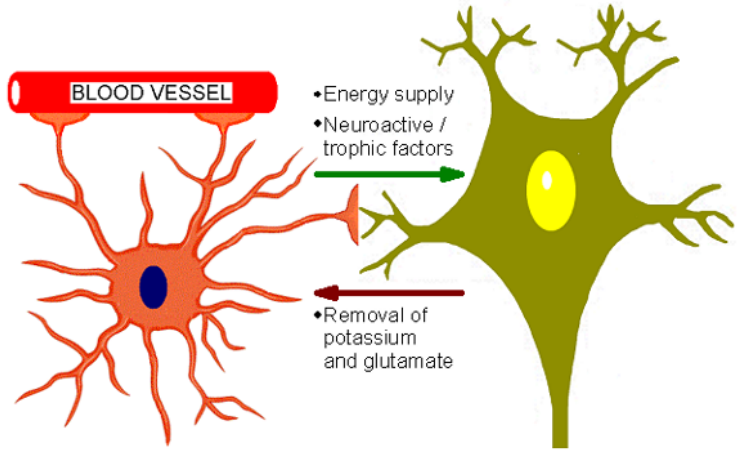
might underlie the propagation of astrocyte Ca^{2+} waves, and the involvement of an extracellular component in the intercellular Ca^{2+} wave was suggested (Hassinger, Guthrie, Atkinson, Bennett, & Kater, 1996), because Ca^{2+} waves can pass frequently between disconnected cells as long as the gap between them does not exceed ~120 μm. The extracellular component is likely to be ATP that can be released from astrocytes. ATP release appears to be an important component of long-range Ca^{2+} signaling, whereas shorter range signaling may be mediated by cellular coupling through gap junctions (Figure 4) (Cotrina et al., 1998; Guthrie et al., 1999; Charles & Giaume, 2002).

Therefore, our current knowledge indicates that astrocytes can promote long-distance signaling from one to another neuronal network, and that this communication might take place as long as astrocytes are communicated, and even if the two neuronal networks are not coupled through synapses.

Astrocyte-Neuron Communication

Glial cells were classically thought to be non-excitable cells, because unlike neurons, they do not show electrical excitability. However, glial cells possess a form of cellular excitability that is based on variations of the intracellular Ca^{2+} concentration (for reviews see Volterra et al., 2002). Indeed, a little more than a decade ago two pioneering studies

Figure 3. Astrocytes support and monitor neuronal activity. Astrocytes, which are in key position between the capillaries and neurons, take up glucose from the blood, and transform it to glycogen that serves as storage energy supply. Upon demand, metabolic substrates, such as lactate or glutamine, are exported to neurons. On the other hand, astrocytes control the extracellular levels of ions and neuroactive substances, such as glutamate.



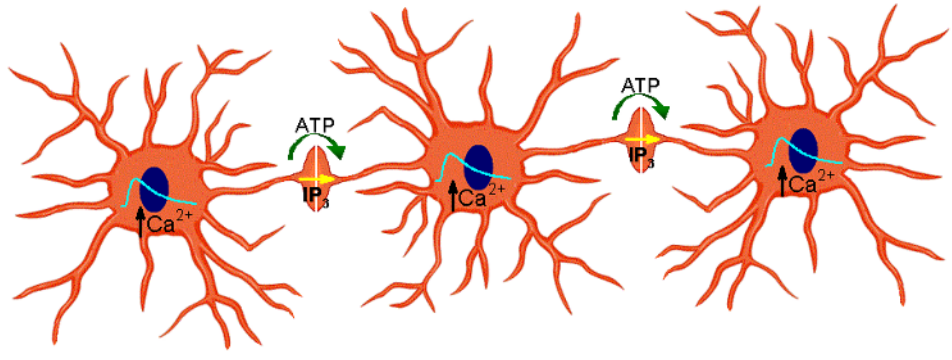
performed on cultures' cells showed that stimulation of an astrocyte, either mechanically or with the neurotransmitter glutamate, caused the elevation of the astrocytic Ca^{2+} that subsequently may propagate nondecrementally to neighboring astrocytes in the form of a wave of elevated Ca^{2+} that can extend for several hundreds of micrometers (Cornell-Bell et al., 1990; Charles, Merrill, Dirksen, & Sanderson, 1991; Innocenti, Parpura, & Haydon, 2000; Newman & Zahs, 1997). Therefore, these Ca^{2+} variations may serve as an intercellular signal that can propagate between astrocytes at a relatively low speed (Schipke et al., 2002), constituting a new type of long-range, slow intercellular communication in the nervous system. In addition, studies performed in brain slices have demonstrated that astrocytes from different brain regions show Ca^{2+} elevations and oscillations that occur spontaneously (Aguado, Espinosa-Parrilla, Carmona, & Soriano, 2002; Nett, Oloff, & McCarthy, 2002; Parri, Gould, & Crunelli, 2001).

A great number of neuroactive substances including neurotransmitters have been shown to elevate the intracellular Ca^{2+} concentration in astrocytes (Figure 5) (Porter & McCarthy, 1997; Verkhratsky, Orkand, & Kettenmann, 1998). Indeed, exogenous application of glutamate (Porter & McCarthy, 1996), norepinephrine (Muyderman et al., 1998), 5-hydroxytryptamine (5-HT) (Hagberg, Blomstrand, Nilsson, Tamir, & Hansson, 1998), histamine (Shelton & McCarthy, 2000), acetylcholine (Shelton & McCarthy, 2000; Sharma & Vijayaraghavan, 2001; Araque, Martin, Perea, Arellano, & Buno, 2002), ATP (Guthrie et al., 1999) and gamma-aminobutyric acid (GABA) (Kang, Jiang, Goldman, & Nedergaard, 1998) may increase the internal Ca^{2+} levels of glial cells through activation of receptors expressed by astrocytes. Consistent with these findings, it has been demonstrated that neurotransmitters released by synaptic terminals can elevate the astrocyte $[Ca^{2+}]_i$ (Araque et al., 2002; Bezzi et al., 1998; Dani et al., 1992; Kang et al., 1998; Kulik, Haentzsch, Luckermann, Reichelt, & Ballanyi, 1999; Nett et al., 2002; Parri et al., 2001; Pasti et al., 1997; Porter & McCarthy, 1996). Therefore, the astrocytic cellular excitability, that is, the astrocyte Ca^{2+} levels, is under the control of the synaptic activity.

Furthermore, recent results obtained in rat hippocampus — a brain region thought to be involved in learning and memory processes — demonstrate that the astrocyte Ca^{2+} signal does not simply reflect synaptic activity, but that astrocytes display integrative properties for synaptic information processing (Perea & Araque, 2005). Indeed, we have shown that astrocytes discriminate between the activity of different synaptic terminals belonging to different axon pathways, and that the synaptic-evoked astrocyte Ca^{2+} signal can be bidirectionally modulated by interaction of different synaptic inputs, being potentiated or depressed depending on the level of synaptic activity. We also have demonstrated that this modulation controls the intracellular expansion of the Ca^{2+} signal and is due to the existence of cellular intrinsic properties in astrocytes. Taken together, these results indicate that astrocytes are endowed with cellular computational characteristics that integrate synaptic information (Perea & Araque, 2005). Therefore, in addition to neurons, astrocytes also could be considered as cellular elements involved in the information processing by the nervous system.

Glial cells may synthesize and release a great number of neuroactive substances, such as glutamate, D-serine, TNFa, or ATP (Araque, Parpura, Sanzgiri, & Haydon, 1998a; Araque, Sanzgiri, Parpura, & Haydon, 1998b; Araque, Li, Doyle, & Haydon, 2000; Araque & Perea, 2005; Arcuino et al., 2002; Beattie et al., 2002; Coco et al., 2003; Haydon &

Figure 4. Astrocytes form a syncitium that exhibit a form of excitability based on intracellular Ca^{2+} variations. These Ca^{2+} variations can propagate intercellularly to adjacent astrocytes, forming a wave of elevated Ca^{2+}. The astrocyte Ca^{2+} waves are mainly mediated by the release of ATP that, acting on purinergic receptors in the adjacent astrocytes, stimulates the production of IP_3, which increase the intracellular Ca^{2+} through the activation of intracellular Ca^{2+} stores. Upon Ca^{2+} oscillations, excitatory amino acids like glutamate and aspartate are released by astrocytes and modulate synaptic activity. The information received might be integrated in the astroglial network via the Ca^{2+} signaling within the electrically coupled astroglial network.



Araque, 2002; Newman, 2003a; Wolosker, Blackshaw, & Snyder, 1999; Zhang et al., 2003). These transmitters — termed gliotransmitters when released by glial cells (Bezzi & Volterra, 2001) — can serve as intercellular signals from glial cells that can signal back to neurons, regulating the postsynaptic neuronal excitability and the neurotransmitter release from presynaptic terminals (Figure 5) (for reviews see, e.g., Araque & Perea, 2004; Auld & Robitaille, 2003; Newman, 2003b). Indeed, we demonstrated in cultured hippocampal cells that the amount of synaptic transmitter released when an action potential reached the synaptic terminals was transiently reduced by stimuli that evoked Ca^{2+} elevations in astrocytes. Additionally, astrocyte Ca^{2+} elevations induced a transient increase in the frequency of the miniature postsynaptic currents (i.e., the unitary events due to spontaneous transmitter release from presynaptic terminals that do not depend on the generation of action potentials). These modulatory phenomena of the synaptic transmission were mediated by glutamate that was released through a Ca^{2+}-dependent process from astrocytes and that activated either presynaptic metabotropic glutamate receptors or NMDA receptors, respectively (Araque et al. 1998a, 1998b). Interestingly, this astrocyte-induced glutamate-mediated modulation of synaptic transmission was present in both excitatory glutamatergic and inhibitory GABAergic synapses. Although this general broad modulation of both types of synapses is likely to be more specifically controlled in the brain, it suggests that different types of synapses may be under the control of astrocytes. Glutamate released from astrocytes also has been shown to

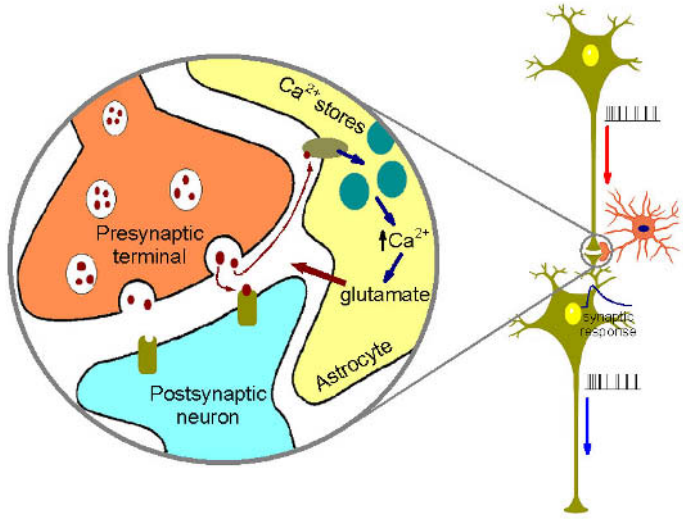
modulate synaptic transmission in more intact preparations. In the retina, it modulates the light-evoked activity of the ganglion cells (Newman & Zahs, 1998), and in the hippocampus, it potentiates inhibitory transmission (Kang et al., 1998; Liu, Xu, Arcuino, Kang, & Nedergaard, 2004) and increases the probability of spontaneous excitatory synaptic transmission (Fiacco & McCarthy, 2004).

In addition to glutamate, glial cells also may release other amino acids (such as aspartate and taurine), neuropeptides, eicosanoids, steroids, and growth factors (for review, see Bezzi & Volterra, 2001). Although most of these gliotransmitters might influence synaptic transmission, this fact has only been demonstrated in a few cases. Besides glutamate, D-Serine and ATP are probably the best-studied examples of gliotransmitters that can modulate synaptic transmission (e.g., Boehning & Snyder, 2003; Koizumi, Fujishita, Tsuda, Shigemoto-Mogami, & Inoue, 2003; Zhang et al., 2003; Miller, 2004).

Several studies have demonstrated that astrocytes also can modulate neuronal excitability. Using cultures of rat hippocampal cells, we showed that elevations of astrocyte intracellular Ca^{2+} levels led to a Ca^{2+}-dependent release of glutamate, which evoked glutamate-dependent slow inward currents (SIC) in adjacent neurons by activation of ionotropic glutamate receptors (Araque et al., 1998a, 1998b, 2000). This direct stimulation of postsynaptic neurons by glutamate release from astrocytes can be observed as neuronal membrane depolarizations that can trigger action potential discharges (Hassinger et al., 1995; Araque et al., 1998a). Astrocyte-induced glutamate-mediated SIC in neurons also have been demonstrated in acute thalamic (Parri et al., 2001) and hippocampal slices (Angulo, Kozlov, Charpak, & Audinat, 2004; Fellin et al., 2004; Perea & Araque, 2004). These neuronal SICs can occur synchronously in multiple neurons suggesting that they may synchronize neuronal activity (Angulo et al., 2004; Fellin et al., 2004). Moreover, astrocytes activated by a specific synapse can transfer information to neighboring neurons and synapses because a single astrocyte can contact multiple neurons and can eventually influence ~140.000 synapses (Ventura & Harris, 1999) (Figure 6). In addition, the intercellular signaling between astrocytes through Ca^{2+} waves may serve as a mechanism for a long-range slower information transfer in the CNS, representing an alternative parallel information pathway to the rapid action potential-based neuronal communication.

In summary, astrocytes exhibit both a form of excitability based on variations of the intracellular Ca^{2+} concentration, they possess a form of intercellular communication based on intercellular Ca^{2+} waves, the astrocytic cellular excitability is triggered and regulated by the synaptic activity, and, in turn, astrocytes release gliotransmitters that modulate the neuronal electrical activity and the synaptic transmission. As a consequence of the demonstration of these new forms of cellular signaling between astrocytes and neurons supports the existence of new and complex information pathways in the CNS, which are based on the existence of bidirectional communication between astrocytes and neurons, and which have relevant consequences on the cellular mechanisms responsible for the information processing of the CNS.

Figure 5. Schematic drawing illustrating the new concept of the synaptic physiology — the tripartite synapse — where astrocytes play an active role by exchanging information with the synaptic elements. Two pairs of neurons with pre- and post-synaptic contact are shown, as well as an astrocyte in close proximity to the synapse. During synaptic activity, neurotransmitters released form presynaptic terminals elicit postsynaptic potentials. The neurotransmitters eventually reach the astrocytic membrane, activating receptors that increase astrocytic Ca^{2+} levels through the release of Ca^{2+} from the internal stores. Elevated astrocytic Ca^{2+} may evoke the release of the neuroactive substances such as the chemical transmitter glutamate, which can modulate the synapse transmission.



Future Trends

Neurobiological modeling has the goal to develop models of artificial neuronal network. In this context, the knowledge of the exact cellular properties of the nervous system is essential. The last few decades have produced a vast amount of knowledge about the function of the nervous system, mostly concerning the function of neurons. Much more recently, the possible importance of astroglial cells in the biological neuronal network has emerged. One of the most novel and exciting areas of neuroscience has emerged after the demonstration of the existence of intercellular astrocyte communication, which may represent a novel extraneuronal communication system, possibly with information processing capacity. Furthermore, the existence of reciprocal communication between astrocytes and neurons adds further complexity to the communication pathways in the nervous system. Therefore, future developments concerning artificial neuronal networks might be improved by including the possibility that an artificial glial network would provide a parallel super-regulatory system. Therefore, three different components of the overall nervous system must be considered: the astrocyte network, the network of

neurons, and, finally, the neuroglial network that is based on the reciprocal communication between neuron and astrocyte networks. These networks are based on different and successive organizational levels in the brain, from molecular to cellular and supracellular levels.

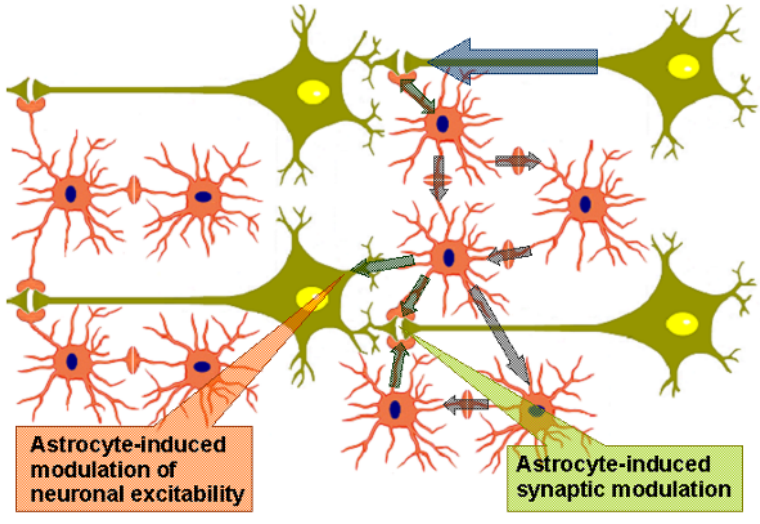
Although a great advance has occurred in the last few years in our understanding of basic cellular and molecular processes in the astrocyte-neuron communication, many questions exist about the specific functional roles as well as the mechanisms by which glial cells might participate in information processing in the CNS. For example, it is well established at the synaptic level that astrocytes respond to a neurotransmitter signal from a nearby neuron and that they can signal back to neurons through the release of gliotransmitters, however, it is largely unknown how signals, either from the astrocyte or the presynaptic terminal or both, are decoded in the post-synaptic terminal and transformed into cellular and network activity.

Animal complex behaviours, including the so-called higher functions of the nervous system, seem to be based on emergent properties that depend on the complexity of the dynamics and interactions between the various levels of brain organization. Due to the high structural and functional complexity of the nervous systems, it seems almost impossible to even approach to "simulate" the architecture and dynamics of the brain on an artificial neuronal network. Therefore, engineering an artificial equivalent of a biological nervous system can be considered as a practically impossible task. However, it may be possible to construct an artificial neuronal network in the future that incorporates synaptically connected neurons in a background or parallel network of astrocytes which may lead to improvements in the ability of artificial networks to behave and perform different tasks that are beyond the capacity of current artificial neuronal networks.

Conclusions

During the last few years, research in glial physiology has dramatically increased our current knowledge of cellular mechanisms underlying relevant aspects of the physiology of the nervous system. It is now well known that these cells have features that were previously thought to be specific of neurons, such as ionic channels, membrane receptors for most neurotransmitters and neuromodulators, functional signal transduction systems responding to those signaling molecules, amino acid neurotransmitter transporters, and the cellular and molecular machinery to release neuroactive substances. Therefore, astrocytes have the cellular and molecular prerequisites that enable them to monitor synaptic activity and to sense the composition of the extracellular milieu and the blood serum. These cells form a gap junction linked syncytium and display a new Ca^{2+} signaling system from cell to cell within the astroglial network. Furthermore, astrocytes and neurons reciprocally communicate between them by exchanging information through the release of gliotransmitters and neurotransmitters that modulate their cellular and network activity.

Figure 6. Schematic drawing illustrating the neuroglial network concept. Astrocytes respond to synaptic activity with intracellular Ca^{2+} elevations, that is, neurons communicate with astrocytes. These Ca^{2+} elevations can propagate to other astrocytes forming a wave of elevated Ca^{2+} that can slowly spread to relatively long distances, hence, astrocytes communicate between them. Finally, astrocytes may release neuroactive substances, such as glutamate or ATP, that can modulate neuronal excitability and synaptic transmission. Thus, astrocytes can signal back to neurons. This loop of information exchange between astrocytes and neurons represents a new form of intercellular communication in the CNS.



Our vision of the CNS physiology has dramatically changed after the demonstration of the presence of new and complex information pathways in the CNS, which are based on the existence of bidirectional communication between astrocytes and neurons, and which have relevant consequences on the cellular mechanisms responsible for the information processing of the CNS. As a consequence of these findings, a new vision of the physiology of the CNS has emerged in which the CNS can no longer be considered to be simply functioning as a cellular network of neurons. Rather, the CNS comprises a complex ensemble of cellular networks of neurons and glial cells. Thus, the CNS must be considered as a superstructural neuroglial network.

Acknowledgments

We thank G. Perea for helpful comments on the manuscript.

This article was written with grant support to A.A. from Ministerio de Ciencia y Tecnología (BFI2001-0206) and Ministerio de Educación y Ciencia (BFU2004-00448), Spain.

References

Aguado, F., Espinosa-Parrilla, J.F., Carmona, M.A., & Soriano, E. (2002). Neuronal activity regulates correlated network properties of spontaneous calcium transients in astrocytes in situ. *Journal of Neuroscience, 22*, 9430-9444.

Angulo, M.C., Kozlov, A.S., Charpak, S., & Audinat, E. (2004). Glutamate released from glial cells synchronizes neuronal activity in the hippocampus. *Journal of Neuroscience, 24*, 6920-6927.

Araque, A., Carmignoto, G., & Haydon, P.G. (2001). Dynamic signaling between astrocytes and neurons. *Annual Review of Physiology, 63*, 795-813.

Araque, A., Li, N., Doyle, R.T., & Haydon, P.G. (2000). SNARE protein-dependent glutamate release from astrocytes. *Journal of Neuroscience, 20*, 666-673.

Araque, A., Martin, E.D., Perea, G., Arellano, J.I., & Buno, W. (2002). Synaptically released acetylcholine evokes Ca^{2+} elevations in astrocytes in hippocampal slices. *Journal of Neuroscience, 22*, 2443-2450.

Araque, A., Parpura, V., Sanzgiri, R.P., & Haydon, P.G. (1998a). Glutamate-dependent astrocyte modulation of synaptic transmission between cultured hippocampal neurons. *European Journal of Neuroscience, 10*, 2129-2142.

Araque, A., & Perea, G. (2004). Glial modulation of synaptic transmission in culture. *Glia, 47*, 241-248.

Araque, A., Sanzgiri, R.P., Parpura, V., & Haydon, P.G. (1998b). Calcium elevation in astrocytes causes an NMDA receptor-dependent increase in the frequency of miniature synaptic currents in cultured hippocampal neurons. *Journal of Neuroscience, 18*, 822-6829.

Arcuino, G., Lin, J.H., Takano, T., Liu, C., Jiang, L., Gao, Q., Kang, J., & Nedergaard, M. (2002). Intercellular calcium signaling mediated by point-source burst release of ATP. *Proceedings of the National Academy of Sciences of the USA, 99* (pp. 9840-9845).

Auld, D.S., & Robitaille R. (2003). Glial cells and neurotransmission: An inclusive view of synaptic function. *Neuron, 40*, 389-400.

Banker, G.A. (1980). Trophic interactions between astroglial cells and hippocampal neurons in culture. *Science, 209*, 809-810.

Barres, B.A., Chun, LL.Y., & Corey, D.P. (1990). Ion channels in vertebrate glia. *Annual Review of Neuroscience, 13*, 441-474.

Bastiani, M.J., & Goodman C.S. (1986). Guidance of neuronal growth cones in the grasshopper embryo. III. Recognition of specific glial pathways. *Journal of Neuroscience, 6*, 3542-3551.

Beattie, E.C., Stellwagen, D., Morishita, W., Bresnahan, J.C., Ha, B.K., Von Zastrow, M., Beattie, M.S., & Malenka, R.C. (2002). Control of synaptic strength by glial TNFalpha. *Science, 295*, 2282-2285.

Bennett, M.V., Barrio, L.C., Bargiello, T.A., Spray, D.C., Hertzberg, E., & Saez, J.C. (1991). Gap junctions: New tools, new answers, new questions. *Neuron, 6*, 305-320.

Bergles, D.E., Diamond, J.S., & Jahr, C.E. (1999). Clearance of glutamate inside the synapse and beyond. *Current Opinion in Neurobiology, 9,* 293-298.

Bernardinelli, Y., Magistretti, P.J., & Chatton, J.Y. (2004). Astrocytes generate Na+-mediated metabolic waves. *Proceedings of the National Academy of Sciences of the USA, 101* (pp. 14937-14942).

Bezzi, P., Carmignoto, G., Pasti, L., Vesce, S., Rossi, D., Lodi Rizzini, B., Pozzan, T., & Volterra, A. (1998). Prostaglandins stimulate calciumdependent glutamate release in astrocytes. *Nature, 391*, 281-285.

Bezzi P., & Volterra A. (2001). A neuron-glia signalling network in the active brain. *Current Opinion in Neurobiology, 11*, 387-394.

Boehning D., & Snyder S.H. (2003). Novel neural modulators. *Annual Review of Neuroscience, 26*, 105-131.

Charles, A.C., Merrill, J.E., Dirksen, E.R., & Sanderson, M.J. (1991). Intracellular signaling in glial cells: Calcium waves and oscillations in response to mechanical stimulation and glutamate. *Neuron, 6*, 983-992.

Charles, A.C., & Giaume, C. (2002). Intercellular calcium waves in astrocytes: Underlying mechanisms and functional significance. In A. Volterra, P.J. Magistretti., & P.G. Haydon (Eds.), *The tripartite synapse: Glia in synaptic transmission* (pp. 110-126). New York: Oxford University Press.

Choi, D.W., & Rothman, S.M. (1990). The role of glutamate neurotoxicity in hypoxic-ischemic neuronal death. *Annual Review of Neuroscience, 13*, 171-182.

Coco, S., Calegari, F., Pravettoni, E., Pozzi, D., Taverna, E., Rosa, P., Matteoli, M., & Verderio, C. (2003). Storage and release of ATP from astrocytes in culture. *Journal of Biological Chemistry, 278*, 1354-1362.

Cornell-Bell, A.H., Finkbeiner, S.M., Cooper, M.S., & Smith, S.J. (1990). Glutamate induces calcium waves in cultured astrocytes: Long-range glial signaling. *Science, 247*, 470-473.

Cotrina, M.L., Lin, J.H., Alves-Rodrigues, A., Liu, S., Li, J., Azmi-Ghadimi, H., Kang, J., Naus, C.C., & Nedergaard, M. (1998). Connexins regulate calcium signaling by controlling ATP release. *Proceedings of the National Academy of Sciences of the USA, 95* (pp. 15735-15740).

Dani, J.W., Chernjavsky, A., & Smith, S.J. (1992). Neuronal activity triggers calcium waves in hippocampal astrocyte networks. *Neuron, 8*, 429-440.

Fellin, T., Pascual, O., Gobbo, S., Pozzan, T., Haydon, P.G., & Carmignoto, G. (2004). Neuronal synchrony mediated by astrocytic glutamate through activation of extrasynaptic NMDA receptors. *Neuron, 43*, 729-743.

Fiacco, T.A., & McCarthy, K.D. (2004). Intracellular astrocyte calcium waves in situ increase the frequency of spontaneous AMPA receptor currents in CA1 pyramidal neurons. *Journal of Neuroscience, 24,* 722-732.

Guthrie, P.B., Knappenberger, J., Segal, M., Bennett, M.V.L., Charles, A.C., & Kater, S.B. (1999). ATP released from astrocytes mediates glial calcium waves. *Journal of Neuroscience, 19*, 520-528.

Hagberg, G.B., Blomstrand, F., Nilsson, M., Tamir, H., & Hansson, E. (1998). Stimulation of 5-HT2A receptors on astrocytes in primary culture opens voltage-independent Ca^{2+} channels. *Neurochemistry International, 32*, 153-162.

Hamprecht, B., & Dringen, R. (1995). Energy metabolism. In H. Kettenman & B.R. Ransom (Eds.), *Neuroglia* (pp. 473-487). New York: Oxford University Press.

Hassinger, T.D., Atkinson, P.B., Strecker, G.J., Whalen, L.R., Dudek, F.E., Kossel, A.H., & Kater S.B. (1995). Evidence for glutamate-mediated activation of hippocampal neurons by glial calcium waves. *Journal of Neurobiology, 28*, 159-170.

Hassinger, T.D., Guthrie, P.B., Atkinson, P.B., Bennett, M.V., & Kater, S.B. (1996). An extracellular signaling component in propagation of astrocytic calcium waves. *Proceedings of the National Academy of Sciences USA, 93*, 13268-13273.

Hatten M.E., & Mason, C.A. (1990). Mechanisms of glial-guided neuronal migration in vitro and in vivo. *Experientia, 46*, 907-916.

Haydon, P.G., & Araque, A. (2002). Astrocytes as modulators of synaptic transmission. In A. Volterra, P.J. Magistretti., & P.G. Haydon (Eds.), *The tripartite synapse: Glia in synaptic transmission* (pp. 185-198). New York: Oxford University Press.

Hertz, L., & Peng, L. (1992). Effects of monoamine transmitters on neurons and astrocytes: Correlation between energy metabolism and intracellular messengers. *Progress in Brain Research, 94*, 283-301.

Innocenti, B., Parpura, V., & Haydon, P.G. (2000). Imaging extracellular waves of glutamate during calcium signaling in cultured astrocytes. *Journal of Neuroscience, 20*, 1800-1808.

Johnson, P.W., Abramow-Newerly, W., Seilheimer, B., Sadoul, R., Tropak, M.B., Arquint, M., Dunn, R.J., Schachner, M., & Roder J.C. (1989). Recombinant myelin-associated glycoprotein confers neural adhesion and neurite outgrowth function. *Neuron, 3*, 377-385.

Kandel, E.R., Schwartz, J.H., & Jessell, T.M. (2000). *Principles of neural science* (4th ed.). New York: McGraw-Hill.

Kang, J., Jiang, L., Goldman, S.A., & Nedergaard, M. (1998). Astrocyte-mediated potentiation of inhibitory synaptic transmission. *Nature Neuroscience, 1*, 683-692.

Koizumi, S., Fujishita, K., Tsuda, M., Shigemoto-Mogami, Y., & Inoue, K. (2003). Dynamic inhibition of excitatory synaptic transmission by astrocyte-derived ATP in hippocampal cultures. *Proceedings of the National Academy of Sciences USA, 100* (pp. 11023-11028).

Kulik, A., Haentzsch, A., Luckermann, M., Reichelt, W., & Ballanyi, K. (1999). Neuron-glia signaling via a_1 adrenoceptor-mediated Ca^{2+} release in Bergmann glial cells in situ. *Journal of Neuroscience, 19*, 8401-8408.

Kuno, M. (1995). *The synapse: Function, plasticity, and neurotrophism*. New York: Oxford University Press.

Kuwada, J.Y. (1986). Cell recognition by neuronal growth cones in a simple vertebrate embryo. *Science, 233*, 740-746.

Largo, C., Cuevas, P., Somjen, G.G., Martin del Rio, R., & Herreras, O. (1996). The effect of depressing glial function in rat brain in situ on ion homeostasis, synaptic transmission, and neuron survival. *Journal of Neuroscience, 16,* 1219-1229.

Le Roux, P.D., & Reh, T.A. (1994). Regional differences in glial-derived factors that promote dendritic outgrowth from mouse cortical neurons in vitro. *Journal of Neuroscience, 14,* 4639-4655.

Liu, Q.S., Xu, Q., Arcuino, G., Kang, J., & Nedergaard, M. (2004). Astrocyte-mediated activation of neuronal kainate receptors. *Proceedings of the National Academy of Sciences USA, 101* (pp. 3172-3177).

Magistretti, P.J., Sorg, O., Yu, N., Pellerin, L., de Rham, S., & Martin, J.L. (1994). Regulation of astrocyte energy metabolism by neurotransmitters. *Renal Physiological Biochemistry, 17*, 168-171.

Mennerick, S., & Zorumski, C.F. (1994). Glial contributions to excitatory neurotransmission in cultured hippocampal cells. *Nature, 368*, 59-62.

Miller, R.F. (2004). D-serine as a glial modulator of nerve cells. *Glia, 47*, 275-288.

Muyderman, H., Nilsson, M., Blomstrand, F., Khatibi, S., Olsson, T., Hansson, E., & Rönnbäck, L. (1998). Modulation of mechanically induced calcium waves in hippocampal astroglial cells. Inhibitory effects of ±1-adrenergic stimulation. *Brain Research, 793*, 127-135.

Nedergaard, M., Ransom, B., & Goldman, S.A. (2003). New roles for astrocytes: Redefining the functional architecture of the brain. *Trends in Neurosciences, 26,* 523-530.

Nett, W.J., Oloff, S.H., & McCarthy, K.D. (2002). Hippocampal astrocytes in situ exhibit calcium oscillations that occur independent of neuronal activity. *Journal of Neurophysiology*, *87*, 528-537.

Newman, E.A. (2003a). Glial cell inhibition of neurons by release of ATP. *Journal of Neuroscience, 23*, 1659-1666.

Newman, E.A. (2003b). New roles for astrocytes: Regulation of synaptic transmission. *Trends Neuroscience, 26*, 536-542.

Newman, E.A., & Zahs, K.R. (1997). Calcium waves in retinal glial cells. *Science, 275*, 844-847.

Newman, E.A., & Zahs, K.R. (1998). Modulation of neuronal activity by glial cells in the retina. *Journal of Neuroscience 18*, 4022-4028.

Nicholls, J.G., Martin, A.R., & Wallace, B.G. (1992). *From neuron to brain: A cellular and molecular approach to the function of the nervous system*. Sunderland, MA: Sinauer Associates.

Noble, M., Fok-Seang, J., & Cohen, J. (1984). Glia are a unique substrate for the in vitro growth of central nervous system neurons. *Journal of Neuroscience, 4*, 1892-1903.

Orkand, R.K., Nicholls, J.G., & Kuffler, S.W. (1966). Effect of nerve impulses on the membrane potential of glial cells in the central nervous system of amphibia. *Journal of Neurophysiology, 29*, 788-806.

Parri, H.R., Gould, T.M., & Crunelli, V. (2001). Spontaneous astrocytic Ca2+ oscillations in situ drive NMDAR-mediated neuronal excitation. *Nature Neuroscience, 4*, 803-812.

Pasti, L., Volterra, A., Pozzan, T., & Carmignoto, G. (1997). Intracellular calcium oscillations in astrocytes: A highly plastic, bidirectional form of communication between neurons and astrocytes in situ. *Journal of Neuroscience, 17*, 7817-7830.

Pellerin, L., & Magistretti, P.J. (2004). Neuroenergetics: Calling upon astrocytes to satisfy hungry neurons. *Neuroscientist, 10,* 53-62.

Perea, G., & Araque, A. (2005). Properties of synaptically evoked astrocyte calcium signal reveal synaptic information processing by astrocytes. *Journal of Neuroscience, 25,* 2192-2203.

Peters, A., Palay, S.L., & Webster, H.F. (1991). *The fine structure of the nervous system: Neurons and their supporting cells* (3rd ed.). New York: Oxford University Press.

Pfrieger, F.W., & Barres, B.A. (1995). What the fly's glia tell the fly's brain. *Cell, 83*, 671-674.

Porter, J.T., & McCarthy, K.D. (1996). Hippocampal astrocytes in situ respond to glutamate released from synaptic terminals. *Journal of Neuroscience, 16*, 5073-5081.

Porter, J.T., & McCarthy K.D. (1997). Astrocytic neurotransmitter receptors in situ and in vivo. *Progress in Neurobiology, 51*, 439-455.

Rakic, P. (1990). Principles of neural cell migration. *Experientia*, *46*, 882-891.

Schipke, C.G., Boucsein, C., Ohlemeyer, C., Kirchhoff, F., & Kettenmann, H. (2002). Astrocyte Ca2+ waves trigger responses in microglial cells in brain slices. *The FASEB Journal, 16*, 255-257.

Schousboe, A. (1981). Transport and metabolism of glutamate and GABA in neurons and glial cells. *International Review in Neurobiology, 22*, 1-45.

Sharma, G., & Vijayaraghavan, S. (2001). Nicotinic cholinergic signaling in hippocampal astrocytes involves calcium-induced calcium release from intracellular stores. *Proceedings of the National Academy of Sciences USA, 98* (pp. 4148-4153).

Shelton, M.K., & McCarthy, K.D. (2000). Hippocampal astrocytes exhibit Ca^{2+}-elevating muscarinic cholinergic and histaminergic receptors in situ. *Journal of Neurochemistry, 74,* 555-563.

Shepherd, G.M. (1994). *Neurobiology*. New York: Oxford University Press.

Slezak, M., & Pfrieger, F.W. (2003). New roles for astrocytes: Regulation of CNS synaptogenesis. *Trends in Neuroscience, 26*, 531-535.

Sneyd, J., Charles, A.C., & Sanderson, M.J. (1994). A model for the propagation of intercellular calcium waves. *American Journal of Physiology, 266*, C293-C302.

Sontheimer, H. (1994). Voltage-dependent ion channels in glial cells. *Glia, 11*, 156-172.

Sul, J.Y., Orosz, G., Givens, R.S., & Haydon, P.G. (2004). Astrocytic connectivity in the hippocampus. *Neuron Glia Biology, 1*, 3-11.

Swanson, R.A., & Choi, D.W. (1993). Glial glycogen stores affect neuronal survival during glucose deprivation in vitro. *Journal of Cerebral Blood Flow and Metabolism, 13*, 162-169.

Szatkowski, M., Barbour, B., & Attwell, D. (1990). Non-vesicular release of glutamate from glial cells by reversed electrogenic glutamate uptake. *Nature, 348*, 443-446.

Takeshima, T., Shimoda, K., Sauve, Y., & Commissiong, J.W. (1994). Astrocyte-dependent and -independent phases of the development and survival of rat embryonic day 14 mesencephalic, dopaminergic neurons in culture. *Neuroscience, 60*, 809-823.

Takuma, K., Baba, A., & Matsuda, T. (2004). Astrocyte apoptosis: Implications for neuroprotection. *Progress in Neurobiology, 72*, 111-27.

Tsacopoulos, M., & Magistretti, P.J. (1996). Metabolic coupling between glia and neurons. *Journal of Neuroscience, 16,* 877-885.

Ullian, E.M., Christopherson, K.S., & Barres, B.A. (2004). Role for glia in synaptogenesis. *Glia, 47*, 209-216.

Ullian, E.M., Sapperstein, S.K., Christopherson, K.S., & Barres, B.A. (2001). Control synapse number by glia. *Science, 291*, 657-661.

Ventura, R., & Harris, K.M. (1999). Three-dimensional relationships between hippocampal synapses and astrocytes. *Journal of Neuroscience, 19*, 6897-6906.

Verkhratsky, A., Orkand, R.K., & Kettenmann, H. (1998). Glial calcium: Homeostasis and signaling function. *Physiological Review, 78*, 99-141.

Verkhratsky, A., Solovyeva, N., & Toescu, E.C. (2002). Calcium excitability of glial cells. In A. Volterra, P.J. Magistretti., & P.G. Haydon (Eds.), *The tripartite synapse: Glia in synaptic transmission* (pp. 99-109). New York: Oxford University Press.

Vernadakis, A. (1996). Glia-neuron intercommunications and synaptic plasticity. *Progress in Neurobiology, 49*, 185-214.

Volterra, A., Magistretti, P.J., & Haydon, P.G. (2002). *The tripartite synapse: Glia in synaptic transmission*. New York: Oxford University Press.

Walz, W. (1989). Role of glial cells in regulation of the brain ion microenvironment. *Progress in Neurobiology, 33*, 309-333.

Wolosker, H., Blackshaw, S., & Snyder, S.H. (1999). Serine racemase: A glial enzyme synthesizing D-serine to regulate glutamate-N-methyl-D-aspartate neurotransmission. *Proceedings of the National Academy of Sciences USA, 96* (pp. 13409-13414).

Zhang, J.M., Wang, H.K., Ye, C.Q., Ge, W., Chen, Y., Jiang, Z.L., Wu, C.P., Poo, M.M., & Duan, S. (2003). ATP released by astrocytes mediates glutamatergic activity-dependent heterosynaptic suppression. *Neuron, 40*, 971-982.

Section II

Time Series Forecasting

Chapter III

Time Series Forecasting by Evolutionary Neural Networks

Paulo Cortez, University of Minho, Portugal

Miguel Rocha, University of Minho, Portugal

José Neves, University of Minho, Portugal

Abstract

This chapter presents a hybrid evolutionary computation/neural network combination for time series prediction. Neural networks are innate candidates for the forecasting domain due to advantages such as nonlinear learning and noise tolerance. However, the search for the ideal network structure is a complex and crucial task. Under this context, evolutionary computation, guided by the Bayesian Information Criterion, makes a promising global search approach for feature and model selection. A set of 10 time series, from different domains, were used to evaluate this strategy, comparing it with a heuristic model selection, as well as with conventional forecasting methods (e.g., Holt-Winters & Box-Jenkins methodology).

Introduction

Nowadays, the fierce competition between individuals and organizations is a trademark of modern societies, where the gain of strategic advantages may be the key to success. The ability to forecast the future, based on past data, makes an important leverage that can push organizations forward. *Time series forecasting*, the forecast of a time ordered variable, is an important tool under this scenario, where the goal is to predict the behavior of complex systems solely by looking at patterns in past data. Indeed, an increasing focus has been put over this field. Contributions from the arenas of *operational research*, *statistics,* and *computer science* has led to solid forecasting methods that replaced intuition, such as the *Holt-Winters* (Winters, 1960) and *Box-Jenkins* (1976) methodology. However, these models were developed decades ago, where higher computational restrictions prevailed. Although these methods give accurate forecasts on linear time series, they carry a handicap with noisy or nonlinear components (Shoneburg, 1990), which are common in real-world situations (e.g., financial data).

An alternative approach for time series forecasting arises from the use of *artificial neural networks*, connectionist models inspired in the behavior of the central nervous system, being attractive artifacts for the design of intelligent systems in *data mining* and *control* applications. In particular, the *multilayer perceptron* is the most popular neural architecture, where *neurons* are grouped in *layers* and only *forward connections* exist, providing a powerful base-learner with advantages such as nonlinear learning and noise tolerance (Haykin, 1999). Indeed, the use of multilayer perceptrons for time series modeling began in the late 1980s and the field has been consistently growing since (Lapedes & Farber, 1987; Cortez, Rocha, Machado, & Neves, 1995; Huang, Xu, & Chan-Hilton, 2004).

The interest in multilayer perceptrons was stimulated by the advent of the *back-propagation* algorithm in 1986, and since then several fast variants have been proposed, such as the *RPROP* algorithm (Riedmiller, 1994). Yet, these training procedures minimize an error function by tuning the modifiable parameters (or *weights*) of a fixed architecture, which needs to be set a priori. Moreover, the neural performance will be sensitive to this choice: A small network will provide limited learning capabilities, while a large one will overfit the training data, inducing generalization loss.

The neural network topology (i.e., connectivity) design is a complex task, commonly addressed by simple trial-and-error procedures (e.g., by exploring a different number of hidden nodes), in a *blind* search strategy that only goes through a small set of possible configurations. More elaborated methods have been proposed, such as the *pruning* (Thimm & Fiesler, 1995) and *constructive* (Kwok & Yeung, 1999) algorithms, which present an effort toward an automatic neural design. However, these *hill-climbing* procedures present two major drawbacks: They tend to get trapped into local minima, and they search through a small portion of architectures, rather than the entire search space.

A different alternative is offered by *evolutionary computation*, which denotes a family of computational procedures inspired in the process of natural selection. The evolutionary algorithms are innate candidates for optimization tasks, performing a global multi-point (or *beam*) search, quickly locating areas of high quality, even when the search space

is very large and complex (Michalewicz, 1996). The hybrid combination of evolutionary computation and neural networks, often called *evolutionary neural networks* (Yao, 1999), is a suitable candidate for topology design due to the characteristics of the error surface: The mapping from the neural structure to its performance is indirect; changes are discrete and can provide discontinuous effects; and similar topologies may present different performances while distinct architectures may provide similar outputs.

The field of *time series forecasting* is a fertile domain for the test of evolutionary neural combinations, since the neural performance will depend not only on the topology internal setup but also on the correct set of its inputs (time lags). Therefore, the present work presents a self-adaptive mechanism for the design of multilayer perceptrons in order to predict a wide range of real and artificial time series, such as the *airline passengers* (Box & Jenkins, 1976). The evolutionary optimization is materialized by a *genetic algorithm*, being the whole process guided by the *Bayesian Information Criterion*, a simple *information theory* statistic that prevents overfitting by adding a model complexity penalty (Schwarz, 1978). The handicap of this strategy is compared with a heuristic approach for neural network model selection and conventional forecasting methods, being competitive.

The chapter will be organized as follows. First, the basic concepts are introduced for the time series analysis and neural forecasting models. Next, the time series data used in the experiments is described. Then, the two main issues for neural forecasting are presented, namely the *input time lags* and *topology* selection. In the next section, the evolutionary approach for neural design is described and tested, being the results compared with the conventional forecasting methods. Finally, future trends are discussed and closing conclusions are drawn.

Background

Time Series Analysis

A *time series* is a collection of periodic ordered observations $(x_1, x_2, \ldots, x_t)$, appearing in a wide set of domains such as *agriculture*, *physics*, *finance,* or *engineering*, just to name a few. Indeed, the analysis of time series corporals an important area of the *statistics* discipline (Makridakis, Wheelwright, & Hyndman, 1998). A time series model ($\hat{x}_t$) assumes that the observations are dependent; that is, past patterns will occur in the future. The aim of a forecasting model is to capture (i.e., *predict*) the behavior of a complex phenomenon (in a *black-box* approach) and not the cause-and-effect relationships of its main components. This is due to the fact there are several systems which are very difficult to understand (e.g., stock market behavior), although being tangible to prediction.

A series is denoted as *deterministic* if it is possible to fully predict it with 100% accuracy. In practice, the majority of the time series will contain a stochastic component, leading to a *forecasting error*. Thus, the overall performance of a model is evaluated by accuracy

measures, namely the *sum of squared errors (SSE)* and *root mean squared error (RMSE)*, which are given by the equations:

$$SSE = \sum_{i=t+1}^{t+L} e_i^2$$
$$RMSE = \sqrt{\frac{SSE}{L}} \quad (1)$$

where $e_t = x_t - \hat{x}_t$ denotes the forecasting error and L the number of forecasts.

A common statistic for time series analysis is the *autocorrelation* coefficient (within the range [-1.0; 1.0]), which gives a measure of the correlation between a series and itself, lagged of k periods, being computed as (Box & Jenkins, 1976):

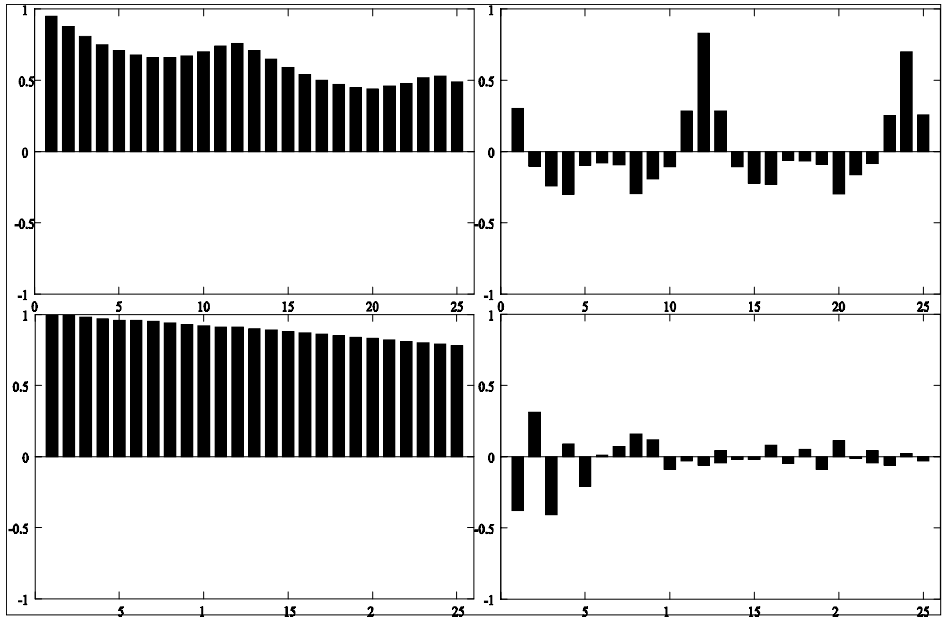
$$r_k = \frac{\sum_{t=1}^{s-k}(x_t - \bar{x})(x_{t+k} - \bar{x})}{\sum_{t=1}^{s}(x_t - \bar{x})} \quad (2)$$

where s and $\bar{x}$ stand for the series size and average. Autocorrelations can be useful for decomposition of the time series main features, such as the *trend* and *seasonal* effects (Figure 1). A *trend* stands for a constant grow (or decline) in the data, being due to factors like inflation or technological improvements. The *seasonal* factor is found in series with a periodic behavior and it is very common in monthly series (e.g., ice cream sales).

The *Holt-Winters* is an important forecasting technique from the family of *exponential smoothing* methods. The predictive model is based on some underlying patterns (e.g., *trended* and *seasonable*) that are distinguished from random noise by averaging the historical values (Winters, 1960). Its popularity is due to advantages such as the simplicity of use, the reduced computational demand, and the accuracy of short term forecasts, especially with seasonal series. The general model can be defined by four equations (Hanke & Reitsch, 1989):

$$F_t = \alpha \frac{x_t}{S_{t-K}} + (1-\alpha)(F_{t-1} + T_{t-1})$$
$$T_t = \beta(F_t - F_{t-1}) + (1-\beta)T_{t-1}$$
$$S_t = \gamma \frac{x_t}{F_t} + (1-\gamma)S_{t-K} \quad (3)$$
$$\hat{x}_t = (F_{t-1} + T_{t-1})S_{t-K}$$

Figure 1. Autocorrelation coefficients (r_k) of typical seasonal & trended, seasonal, trended, and non-trended series (x-axis denotes the k time lags)



where F_t, T_t and S_t denote the smoothing, trend and seasonal estimates, K the seasonal period; and α, β and γ the model parameters. Usually, the constants (α, β and γ) are set by *trial-and-error* procedures. When there is no seasonal component, the γ parameter is discarded and the S_{t-k} factor in the last equation is replaced by the unity.

Another popular forecasting approach is the *Box-Jenkins* (1976) methodology, going through a set of steps such as *model identification*, *parameter estimation,* and *model validation*. The main advantage of this method relies on the accuracy over a wider domain of series, despite being more complex, in terms of usability and computational effort, than Holt-Winters. The global model, based on a linear combination of past values (*AR* components) and errors (*MA* components), is named *autoregressive integrated moving-average (ARIMA)*, while the seasonal version is called *SARIMA*. Both *ARIMA* and *SARIMA* models can be postulated as an *ARMA*(O_A,O_M) one, given by:

$$\hat{x}_t = \mu + \sum_{i=1}^{O_A} A_i x_{t-i} + \sum_{j=1}^{O_M} M_j e_{t-j} \qquad (4)$$

where O_A and O_M denote the *AR* and *MA* orders; A_i and M_j the *AR* and *MA* coefficients; and μ is a constant value. Both the coefficients and the constant of the model are

estimated using statistical methods (e.g., least squares). The trended series require a differencing of the original values and the methodology also may use transformations in the original data (e.g., logarithm function).

Artificial Neural Networks in Time Series Forecasting

Connectionist models are innate candidates for forecasting due to their nonlinear and noise tolerance capabilities. The basic idea is to train a neural network with past data and then use this network to predict future values. In effect, the use of neural forecasting systems began in the late 1980s with the work of Lapedes and Farber (1987), where multilayer perceptrons were used to predict chaotic series. Since then, several neural architectures have been proposed, such as *radial-basis functions* (Shi et al., 1999) or *recurrent networks* (Ulbricht, 1994), although the majority of the studies adopt the multilayer network (Shoneburg, 1990; Tang & Fishwick, 1993; Cortez et al., 1995; Faraday & Chatfield, 1998; Huang et al., 2004). Forecasting competitions, confronting neural networks with traditional methods, have reported either poor (Chatfield, 1993) or favorable (Tang & Fishwick, 1993) results, suggesting that great care is needed when fitting neural forecasting models.

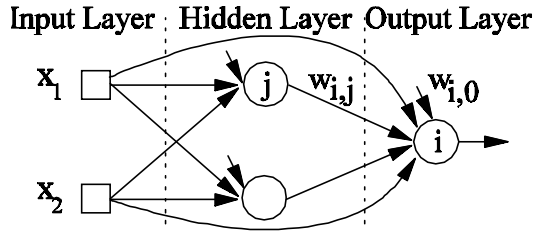
In order to perform time series modeling with a multilayer perceptron, a set of training cases needs to be created from the data by adopting a *sliding time window*. In the literature, this combination is known by the term *time lagged feedforward network*. A *sliding window* is defined by the sequence $<k_1, k_2, \ldots, k_I>$, for a network with I inputs and time lags. For instance, consider the monthly series given by sequence: …, $8_{January}$, $11_{February}$, 16_{March}, 20_{April}, 17_{May}, and 22_{June}. When assuming a time window of <1, 2, 4>, the last training cases that can be created in June are: 8, 16, 20 → 17 and 11, 20, 17 → 22.

In this work, a base fully connected topology will be adopted, with one hidden layer, *bias* and direct *shortcut* connections, from input to output nodes (Figure 2). To enhance nonlinearity, the *logistic* activation function was applied on the hidden nodes, while in the output node, the *linear* function was used instead, to scale the range of the outputs, since the logistic function has a [0, 1] output range. This architecture, often adopted by the forecasting community (Faraday & Chatfield, 1998; Huang et al., 2004), avoids the need of filtering procedures (e.g., *rescaling*), which may give rise to information loss. Therefore, the general model is given in the form:

$$\hat{x}_t = w_{Out,0} + \sum_{i=1}^{I} x_{t-k_i} w_{Out,i} + \sum_{j=I+1}^{Out-1} f\left(\sum_{i=1}^{I} x_{t-k_i} w_{j,i} + w_{j,0}\right) w_{Out,i} \quad (5)$$

where $w_{i,j}$ denotes the weight of the connection from node j to i (if j=0 then it is a *bias* connection), *Out* denotes the output node, f the logistic function ($\frac{1}{1+e^{-x}}$) and I the number of input neurons.

Figure 2. A fully connected multilayer perceptron with 2 inputs, 2 hidden neurons, 1 output, bias, and shortcut connections



Time Series Data

To the experiments carried out in this work, a set of 10 series was selected from different domains and sources. The intention was to choose two datasets to represent each of the five main categories namely: *seasonal & trended*, *seasonal*, *trended*, *non-trended,* and *chaotic*. With the exception of the artificially generated data (**quadratic** and **henon**), all series are available at the *Time Series Data Library* repository, which is maintained by Hyndman and Akram (2005) and contains more than 800 series.

The first type, the *seasonal & trended* series, is very common on monthly data, such as the ones found in sales. The **passengers'** dataset is such an example, representing the number of international passengers (in thousands) of an airline company, from 1949 to 1969. This is a well-known series, being analyzed in the classic study by Box and Jenkins (1976). The **paper** series exhibits the same characteristics, reporting over French industry sales for printing and writing paper (in thousands of francs), from January 1963 to December 1972 (Makridakis et al., 1998).

The datasets with only *seasonal* components include the **deaths** and **maxtemp** series. The former is related to the number of monthly deaths and serious injuries in UK road accidents, from January 1969 to January 1983 (Harvey, 1989). The latter is a meteorological series, containing the monthly mean maximum temperature (in Celsius degrees) measured in Melbourne, Australia, from January 1971 to December 1990 (Hyndman & Akram, 2005).

Regarding the *trended* series, they were first presented in the work of Box and Jenkins (1976). The **prices** dataset contains financial data, relative to the daily IBM stock closing prices, measured from May 17, 1961 to November 2, 1962. The other set denotes temperature readings (being collected every minute) of a **chemical** process.

Both *non-trended* series were taken from the physics domain. The annual **sunspots** numbers was collected from 1700 until 1999, being commonly used to test nonlinear forecasting techniques (Tong, 1983). The **kobe** data, which is also nonlinear, contains the first 200 records of a seismograph, taken from the Kobe earthquake and recorded at

the Tasmania University, Australia on January 16, 1995, beginning at 20:56:51 GMT (Hyndman & Akram, 2004).

Finally, the last two series were artificially created by generating 200 points using chaotic formulas, namely (Peitgen, Jürgens, & Saupe, 1992):

- **quadratic** $\rightarrow x_t = ax_{t-1}(1-x_{t-1})$, where $a = 4$ and $x_0 = 0.2$; and
- **henon** $\rightarrow x_t = 1-ax_{t-1}^2+bx_{t-2}+N(0.0,0.1)$, where $a = 1.4$, $b = 0.3$, $x_0 = 0.11$ and the last factor denotes a Gaussian noise with a 0.1 standard deviation.

Table 1 presents a synopsis of the most relevant features of each of the 10 datasets, while Figure 3 shows them in a graphical perspective. For the experiments, each series was divided into a *training set*, containing the first 90% values and a *test set*, with the last 10% (column **Size** shows the series length). Only the training set is used for model selection and parameter optimization, being the test set used to compare the proposed approach with other methods.

Heuristic Approach to Model Selection

When modeling time series by neural networks, a critical issue is *model selection*; that is, what is the best sliding time window and neural structure for a given series? Both choices can perform a high impact in the forecasting performance. A small window will provide scarce information to the network, while a large number of time lags may increase the entropy, affecting the learning. For instance, the *SARIMA* models often use only the <1, 12, 13> lags for monthly seasonal trended series. Similarly, a network with few hidden neurons will have limited learning capabilities. In contrast, using an excess of hidden nodes will overfit the training data.

The statistical approach to model selection is to consider different candidate models, which are evaluated according to a generalization estimate. Several complex estimators have been developed (e.g., *K-fold cross validation*), which are computationally burdensome (Sarle, 1995). A reasonable alternative is to penalize model complexity, using a simple statistic such as the *Bayesian Information Criterion (BIC)* (Schwarz, 1978):

$$BIC = N \ln(\frac{SSE}{N}) + P \ln(N) \quad (6)$$

where N denotes the number of training cases and P the number of parameters. Although originally proposed for linear models, this criterion has also been advised for neural estimation (Faraday & Chatfield, 1998). When applied to neural networks, the number of parameters (P) is equal to the number of connection weights.

Table 1. The main features of the time series used in the experiments

Series	Type	Size	Description
passengers	seasonal & trended	144	Monthly airline passengers
paper	seasonal & trended	120	Monthly sales of paper
deaths	seasonal	169	Monthly deaths/injuries in UK roads
maxtemp	seasonal	240	Maximum temperature in Melbourne
chemical	trended	198	Chemical concentration readings
prices	trended	369	Daily IBM stock closing prices
sunspots	non-trended	289	Annual Wolf's sunspot numbers
kobe	non-trended	200	Seismograph of the Kobe earthquake
quadratic	chaotic	200	Quadratic map function
henon	chaotic	200	Henon map function

In order to draw some preliminary conclusions, a *heuristic model selection* will be tested. Four rules will be used to generate the sliding time windows, namely (Cortez et al., 1995):

1. Use all time lags from 1 to a given maximum m: <1,2,...,m> (m was set to 13, a value that was considered sufficient to encompass monthly seasonal and trended effects);
2. Use all lags containing autocorrelation values above a given threshold (set to 0.2);
3. Use the four lags with the highest autocorrelations; and
4. Use of decomposition information; that is,

   - <1, K, K+1> → if the series is seasonal (period K) and trended;
   - <1, K> → if the series is seasonal; and
   - <1> and <1, 2> → if the series is trended.

Several multilayer perceptrons, with a number of hidden nodes (H) ranging from 0 to 13, were used to explore all these windows for each series of Table 1. The *RPROP* algorithm was adopted for the training (Riedmiller, 1994), since it allows a faster convergence, being stopped when the error slope approaches zero or after 1,000 epochs. During this process, the training cases were presented in a temporal order to the network, in order to preserve the chronological sequence of the patterns, since they are not independent.

All tests reported in this work were conducted using programming environments developed in *C++* (Neves, Rocha, Rodrigues, Biscaia, & Alves, 1999). As an example, the methodology used will be explained in detail for the sunspots series (Table 2). To simplify the explanation, only some configurations are visible in this table. The results

Figure 3. The 10 time series of Table 1 in a temporal perspective

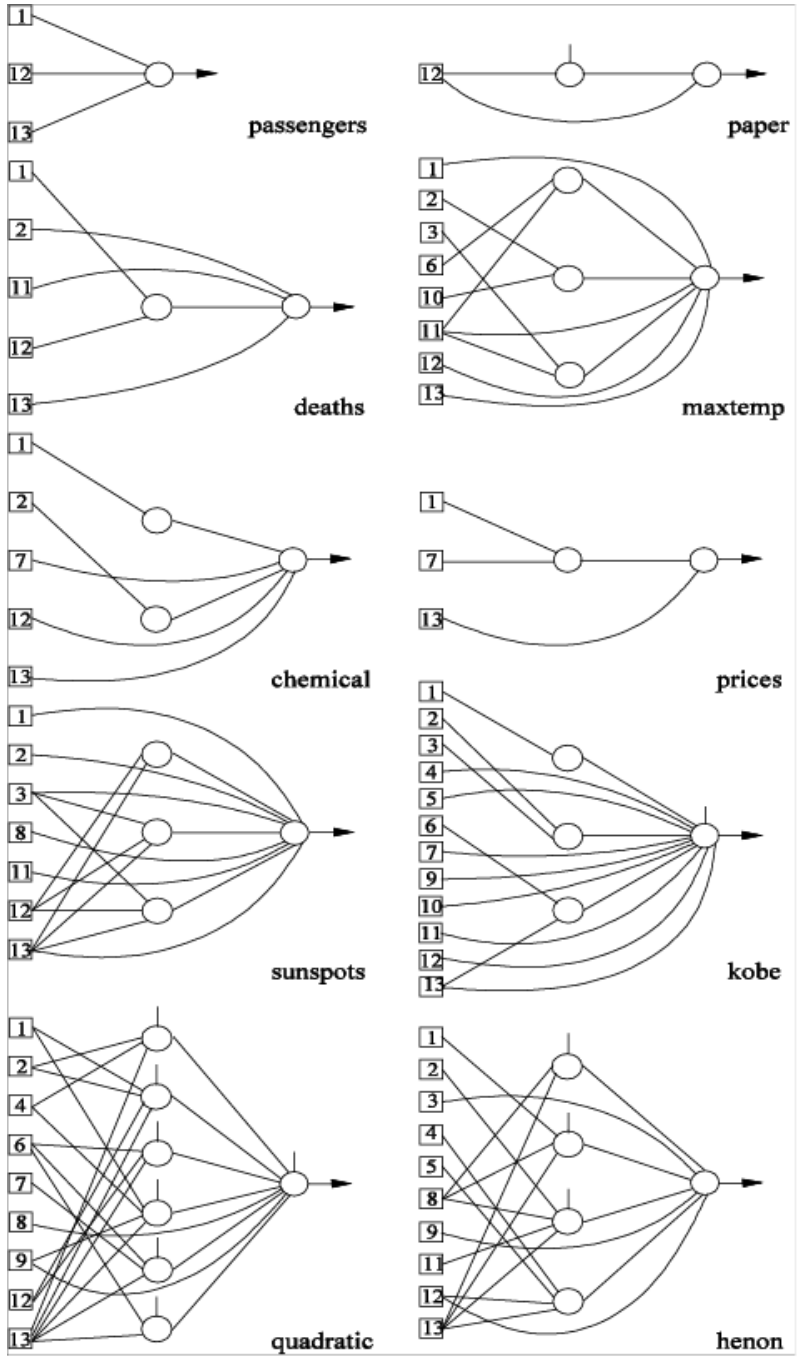


Table 2. The heuristic model selection approach applied to the sunspots series

Time Window	Hidden Nodes (H)	Weights (P)	Training $RMSE_t$	Training BIC	Forecasting $RMSE_f$
Rule 1: <1, 2, …, 13>	0	14	14.7	1404	18.2±0.1
	6	104	11.6	1784	20.5±0.7
	13	194	**9.0**	2155	20.2±0.8
Rule 2: <1, 2, 9, 10, 11, 12>	0	5	14.8	1369	17.9±0.1
	5	47	11.5	1467	**17.0**±0.6
	13	111	9.4	1717	19.0±0.8
Rule 3: <1, 2, 10, 11>	0	5	15.1	1369	18.0±0.0
	1	11	14.1	**1368**	17.8±0.3
	8	53	10.7	1464	19.4±0.5

Table 3. The best neural forecasting models given by the heuristic approach

Series	Time Window	Hidden Nodes (H)	Weights (P)	Forecasting ($RMSE_f$)
passengers	<1, 12, 13>	0	4	18.4±0.2
paper	<1, 12, 13>	0	4	51.6±0.2
deaths	<1, 11, 12, 13>	0	5	134±1
maxtemp	<1, 11, 12, 13>	0	5	0.90±0.01
chemical	<1, 2>	0	3	0.40±0.01
prices	<1>	0	2	7.49±0.01
sunspots	<1, 2, 10, 11>	1	11	17.8±0.3
kobe	<1, 2, …, 13>	0	14	557±4
quadratic	<1, 2, …, 13>	9	149	0.06±0.02
henon	<1, 2, …, 13>	2	44	0.35±0.05

of the last four columns are given in terms of the mean of 30 runs, since the initial weights were randomly generated. The 95% confidence intervals also are shown for the forecasting errors (Flexer, 1996). The statistics computed include the number of weights (P), the training estimates ($RMSE_t$ and BIC), and the forecasting error ($RMSE_f$). In this example, rule 4 is not applied since the sunspots series is neither seasonal nor trended.

The best training error ($RMSE_t$) model contains a high number of parameters and overfits. Using more hidden nodes leads to lower training *RMSE* values. Yet, when a network specializes its performance usually degrades. In contrast, the *BIC* criterion works better by selecting a network that provides one of the best forecasts (column $RMSE_f$). This behavior occurred consistently in all series.

Table 3 shows the best neural networks (with lower *BIC* values) given by the heuristic approach. As expected, the criterion suggests small time windows and linear models (with no hidden nodes) for all linear series. However, the *BIC* statistic also favors simple models (with zero or one hidden nodes) for the nonlinear series (kobe and sunspots). For

instance, the best forecasts for series sunspots are given for a network with five hidden nodes, although resulting in a higher *BIC* value, due to an excessive number of weights (Table 2).

The heuristic model selection strategy presents some limitations. In terms of the *BIC* criterion, adopting fully connected networks seems to prejudice networks with hidden nodes. Moreover, the sliding window rules are based on autocorrelation values, which only measure linear interactions, thus being not adequate for nonlinear series. In the next section, an evolutionary approach will be proposed to solve these drawbacks.

Evolutionary Approach to Model Selection

The *genetic algorithm* was first presented by Holland (1975) and since then the concept has evolved several computational variants that share a set of common features. There are a number of potential solutions (*individuals*) to a problem, evolving simultaneously (a *population*). Each individual is coded by a string (*chromosome*) of symbols (*genes*), taken from a well-defined alphabet. Each individual is assigned a numeric value (*fitness*) that stands for the solution's appropriateness. In every generation, a fraction of the individuals is replaced by the offspring, which are generated by the application of genetic operators, such as *crossover* and *mutation*, in order to create new solutions (*reproduction*). The whole process is *evolutionary*, where the fittest individuals have greater chances of surviving.

The evolutionary optimization of neural topologies has been a fertile ground of research in the last few years (Yao, 1999). The proposed models mainly differ in two issues: the *fitness* function and the neural network *representation*. Regarding the first topic, the most usual approach is to consider an error measure over an independent *validation* set. On the other hand, the representation scheme has been addressed by two alternatives, namely *direct* and *indirect* encodings. The former encodes all the topology details (weights and connections), being the most used approach, since it is more efficient and easy to implement. The latter represents the most important parameters or makes use of construction rules. One interesting example of an indirect scheme is given by the work of Gruau (1992), where a set of grammar rules are used to construct the neural structure. The indirect encoding is more scalable and biologically plausible, favoring regular networks. However, since only modular networks are considered, the search space for the best topology is restricted and this approach has revealed some drawbacks in real-world problems (Siddiqi & Lucas, 1998).

When designing multilayer perceptrons for time series forecasting, it is common to use *trial-and-error* procedures, such as testing several combinations of hidden nodes (Tang & Fishwick, 1993; Faraday & Chatfield, 1998). The use of evolutionary design corporals an alternative attempt for time series neural topology selection. Indeed, several evolutionary neural networks have been proposed, most of them using indirect parametric representation, encoding factors such as the number of input and hidden nodes, the

Figure 4. Demonstration of the encoding process for a given neural structure

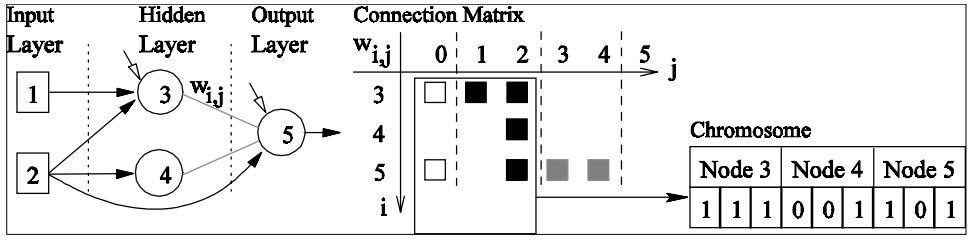
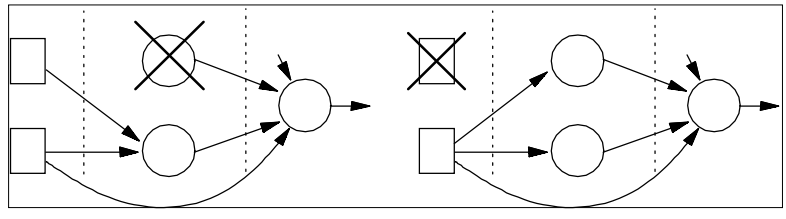


Figure 5. Example of a hidden node (left) and input time lag (right) pruning



initial weights, activation functions, or even learning rates (Falco, Cioppa, Iazzetta, Natale, & Tar, 1998; Chen & Lu, 1999). However, from a model selection point of view, it seems more important to adjust the multilayer perceptron's connectivity.

In this work, a novel evolutionary neural network approach is proposed, using a direct binary representation. Since previous experiments have shown that time series are often modeled by small networks, this encoding will prevent the topologies from growing too large, also being more efficient. Thus, each gene represents a possible connection, if its value is 1, then the corresponding connection exists, otherwise it is not considered. The connections between the hidden nodes and the output one always exist, since this strategy enhances the creation of valid networks. For demonstration purposes, Figure 4 shows how the encoding scheme works.

Assuming a maximum of I input and H hidden nodes, bias, and shortcut connections, the size of the chromosome is given by $(I+1)(H+1)$. Hidden node pruning will occur when there are no connections from input nodes, and input node pruning will occur when an input node has no outputs (Figure 5). This allows a complete representation, defining a search space which contains all neural topologies from the simplest linear perceptron to the fully connected multilayer network, and also to reach any subset of input nodes (performing feature selection).

Table 4. The best neural forecasting models given by the evolutionary approach

Series	Time Window	Hidden Nodes (H)	Weights (P)	Reduction Factor (R)	Forecasting ($RMSE_f$)
passengers	<1, 12, 13>	0	3	97%	18.2±0.3
Paper	<12>	1	4	96%	52.5±0.6
deaths	<1, 2, 11, 12, 13>	1	6	94%	132±1
maxtemp	<1, 2, 3, 6, 10, ..., 13>	3	13	88%	0.87±0.01
chemical	<1, 2, 7, 12, 13>	2	7	93%	0.36±0.01
prices	<1, 7, 13>	1	4	96%	7.49±0.01
sunspots	<1, 2, 3, 8, 11, 12, 13>	3	17	84%	17.4±0.5
kobe	<1, ...,7,9, ...,13>	3	17	84%	498±8
quadratic	<1, 2, 4, 6, 7, 8, 9, 12, 13>	6	34	67%	0.01±0.00
henon	<1, ..., 5, 8, 9, 11, 12, 13>	4	23	78%	0.24±0.03

Regarding the fitness function, using a validation set leads to information loss, since smaller training sets are considered. Furthermore, the temporal nature of the time series domain enhances this effect, once validation cases are typically the most recent in the training data. An alternative is to make use of the *BIC* criterion, which presents the advantages of requiring a simple computation while increasing the selection pressure in favor of simpler structures, following the principle of Ockham's razor.

The initial populations' genes were randomly assigned within the alphabet {0, 1}. The maximum number of input (n) and hidden nodes (H) were set to 13 and 6, since previous experiments have already favored small networks, leading to a chromosome with 98 genes. Since the genetic algorithm works as a second (high-level) optimization procedure, the setting of its parameters is not considered crucial. Backed by preliminary experiments, the population size was set to 100 individuals, being the *two-point crossover* operator responsible for breeding 80% of the offspring, while the *bit mutation* operator is accountable for the remaining ones. Under the proposed approach, the major building blocks can be identified with the set of connections that feed a hidden neuron. Thus, to avoid its disruption, the crossover cutting points were restricted (Opitz & Shavlik, 1994). The *selection* procedure is done by converting the fitness value into its ranking in the population and then applying a roulette wheel scheme. The evolutionary procedure is stopped after a convenient number of G_{Max} generations, here set to 200, once the best individuals were found in earlier generations. Finally, 30 runs are applied to the best evolved topology. The evolutionary algorithm structure is described by the following pseudo-code:

BEGIN

Initialize time (t=0).

Generate the initial neural forecasting population (P_0).

Evaluate the individuals (*BIC* computation) in P_0.

WHILE NOT ($t<G_{Max}$) DO

Select from P_t a number of individuals for reproduction.

Breed the offspring by applying to the genetic operators.

Evaluate the offspring (*BIC* computation).

Select the offspring to insert into the next population (P_{t+1}).

Select the survivors from P_t to be reinserted into P_{t+1}.

Increase the current time (t := t+1).

END

Table 4 and Figure 6 show the best models achieved by the neural genetic approach. Column *R* denotes the *reduction factor*, when comparing the number of weights of the obtained topology (*P*) with the equivalent base fully connected one (P_{fc}):

$$R = 1 - \frac{P}{P_{fc}} \tag{7}$$

The R values are within the 67 to 97% range, showing a substantial pruning by the genetic algorithm. In effect, all hidden nodes receive a short number of connections (from 1 to 4). Furthermore, the selected architectures seem to be adjusted to the series complexity, with a total of 3 to 13 weights for the linear series, 17 for the nonlinear series, rising up to 23 and 34 connections for the chaotic data. All evolved topologies present shortcut connections, attesting their usefulness; in contrast, the bias connections seem less important. As an example, Figure 7 plots the forecasts for the last 29 elements (10%) of series sunspots. In this case, the real value and forecast curves are close, revealing a good fit by the neural model.

Discussion

This section will start with a comparison between the neural and conventional time series forecasting approaches. The Holt-Winters parameters were optimized using a 0.01 grid search for the best training error (*RMSE*), which is a common practice within the forecasting field (Table 5). A different strategy was adopted for the Box-Jenkins methodology, since the model selection stage is non-trivial, requiring the use of experts. Therefore, it was decided to use the forecasting package *Forecast PRO* (Rycroft, 1993), which includes automatic model selection (Table 6). Although there are known models in the literature (Box & Jenkins, 1976), these were not considered since they only cover four of the tested series. Moreover, the literature models presented higher forecasting errors.

Figure 6. The best evolved neural forecasting topologies

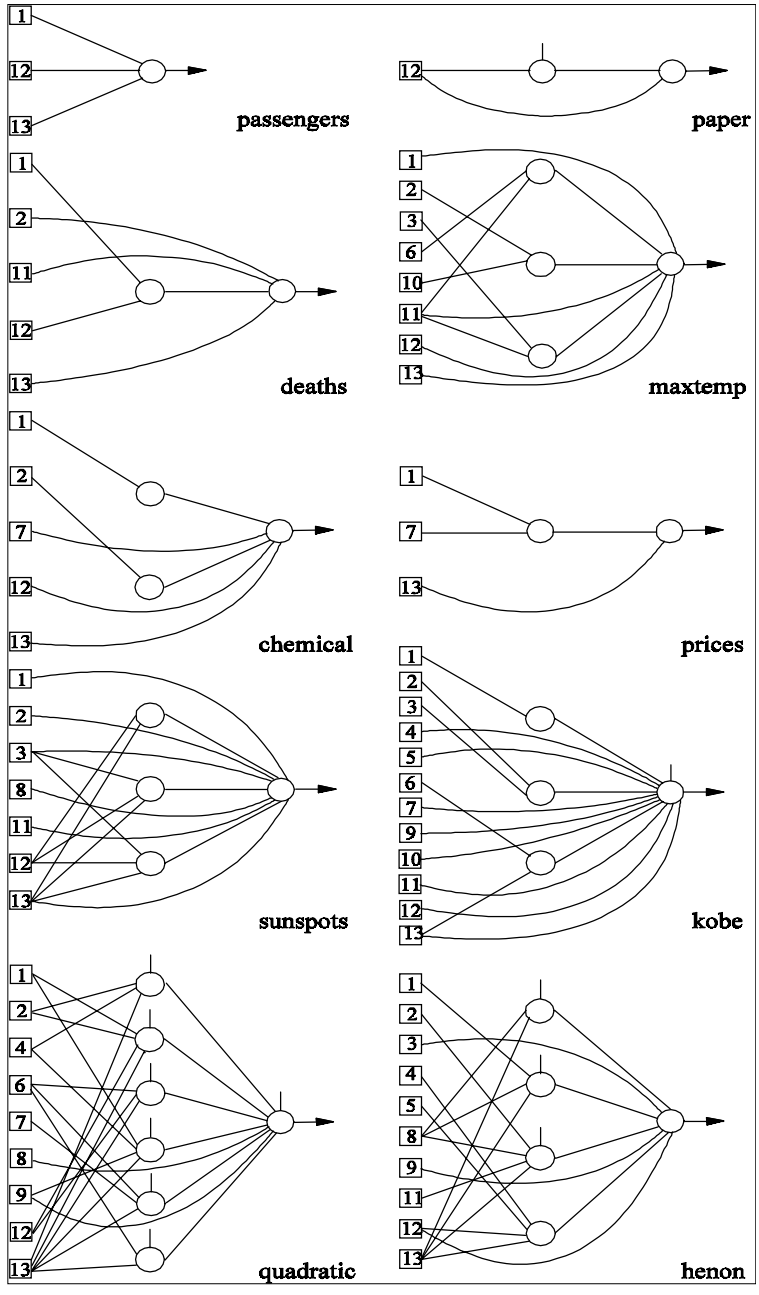


Figure 7. The sunspots' real values and the forecasts given by the evolutionary neural algorithm for the last 29 elements

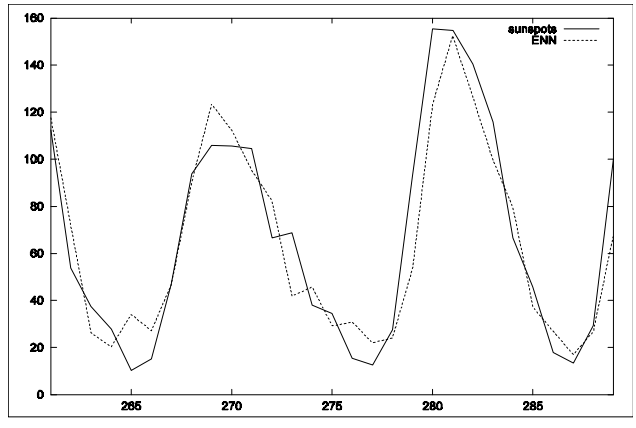


Table 5. The Holt-Winters forecasting models obtained by a 0.01 grid search

Series	α	β	γ	K	$RMSE_f$
passengers	0.29	0.03	0.95	12	16.5
paper	0.25	0.01	0.03	12	49.2
deaths	0.36	0.00	0.01	12	135
maxtemp	0.24	0.00	0.01	12	0.72
chemical	0.30	0.00	-	-	0.35
prices	1.00	0.02	-	-	7.54
sunspots	1.00	0.95	-	-	28.3
kobe	0.05	0.00	-	-	3199
quadratic	0.03	0.00	-	-	0.35
henon	0.02	0.75	-	-	0.73

In Table 7, the performance of the proposed method is compared with the conventional forecasting techniques: the *Holt-Winters (**HW**)* and *Box-Jenkins (**BJ**)*. The error values in the table are given in terms of the *Theil's U* statistic, which is defined by:

$$Theil'sU = \frac{RMSE_f}{RMSE_n} \tag{8}$$

where $RMSE_n$ stands for the forecasting error given by a naive *no change* prediction ($\hat{x}_t = x_{t-1}$). This measure takes values from the range 0 (perfect forecast) to around the unity, making the comparison among the different series and methods easier. Column

Table 6. The forecasting models obtained by the Box-Jenkins methodology

Series	μ	A_i^*	M_j^*	$RMSE_f$
passengers†•	0.0	$<1_1, 1_{12}, -1_{13}>$	$<-0.35_1, -0.62_{12}, 0.22_{13}>$	17.8
paper†	0.0	$<1_1, 1_{12}, -1_{13}>$	$<-0.87_1, -0.80_{12}, 0.70_{13}>$	61.0
deaths†	0.0	$<1_1, 1_{12}, -1_{13}>$	$<-0.66_1, -0.90_{12}, 0.59_{13}>$	144
maxtemp†	0.0	$<1_1, 1_{12}, -1_{13}>$	$<-0.88_1, -0.89_{12}, 0.78_{13}>$	0.91
chemical•	0.3	$<0.90_1>$	$<-0.56_1>$	0.35
prices	0.0	$<1_1>$	$<0.12_1>$	7.72
sunspots	14.2	$<1.39_1, -0.70_2>$	<>	28.4
kobe	3038	$<0.71_1, -0.81_2>>$	$<0.77_1, -0.21_2, -0.06_3>$	582
quadratic	0.48	<>	<>	0.35
henon	0.25	<>	$<-0.30_1, 0.14_2, -0.40_3>$	0.63

** The values denote the coefficients (A_i and M_j) and the subscripts denote the time lags (*i *and* j*).*

† SARIMA models were used for the seasonal series.

<>...The data was pre-processed by a natural logarithm transform.

BNN denotes the result obtained by the base-learner neural model; that is, a fully connected network with 13 input and 6 hidden nodes.

An analysis to Table 7 reveals that the base-learner architecture provides the worst neural performance, probably due to the excess of weights, leading to generalization loss. This shows that more sophisticated model selection strategies need to be pursued. In fact, when comparing the connectionist approaches, the best results are obtained by the evolutionary neural network (the exception is series **paper**), endorsing the use of the genetic algorithm.

A comparison with conventional methods reveals that the Holt-Winters method gives a better performance on the seasonal trended series. This is not surprising, since it was developed specifically for this kind of data. In the trended series, all techniques have difficulty to perform better than the *no change* forecast, with both the Holt-Winters and the evolutionary neural network producing comparable results. This scenario differs when considering other series, where the genetic algorithm exhibits its strength, outperforming both conventional forecasting methods, especially for the last four nonlinear series. The Box-Jenkins methodology excels the Holt-Winters method in four series (**deaths**, **sunspots**, **kobe,** and **henon**) and only outperforms the evolutionary neural strategy in two datasets (**passengers** and **chemical**).

The main drawback of the proposed approach to model selection is the increase of computational effort. The evolutionary optimization requires five times more computational processing than the heuristic model selection strategy. Therefore, if the need for accuracy is not overwhelming, the use of simpler heuristics can provide acceptable solutions. Nevertheless, time complexity could be reduced if a subset of promising models were put into the genetic algorithm*'s* initial population, although this would require the use of *a priori* information. Since most of the time series use daily or monthly data, this is not considered a major concern. As an example of the processing effort,

Table 7. The comparison (Theil's U values) between conventional (Holt-Winters and Box-Jenkins) and neural forecasting approaches (base-learner neural network, heuristic model selection, and evolutionary neural network)

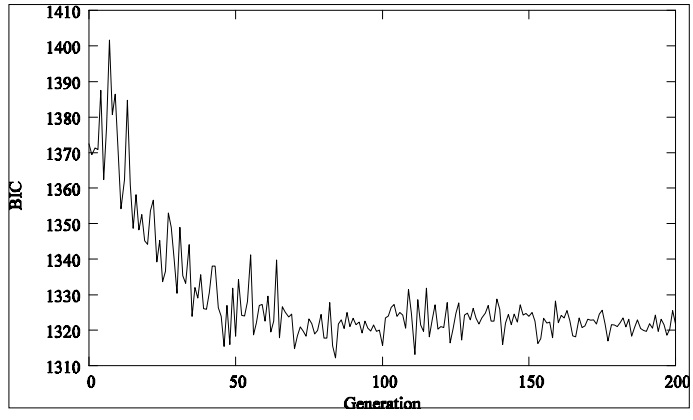
Series	*HW*	*BJ*	*BNN*	*HMS*	*ENN*
passengers	**0.104**	0.118	0.137	0.125	0.123
paper	**0.035**	0.076	0.079	0.055	0.057
deaths	0.501	0.496	0.459	0.429	**0.420**
maxtemp	0.137	0.186	0.138	0.131	**0.125**
chemical	**0.830**	0.861	0.881	1.060	0.873
prices	1.000	1.008	1.899	**0.997**	**0.997**
sunspots	0.762	0.434	0.401	0.301	**0.287**
kobe	0.823	0.027	0.031	0.025	**0.020**
quadratic	0.464	0.464	0.136	0.024	**0.000**
henon	0.417	0.326	0.174	0.110	**0.053**

Figure 8 plots the best fitness (*BIC*) obtained in each generation for the **sunspots** series. In this case, a higher convergence is observed in the first iterations, being the best solution obtained in generation 86. This behavior, which occurred in all series, shows that acceptable solutions can be obtained in earlier stages of the process.

Another important forecasting issue is related to *explanatory knowledge*. In *data mining* applications, besides obtaining a good predictive accuracy, it is also important to provide useful knowledge about what the model has learned (Fayyad, Piatetsky-Shapiro, & Smyth, 1996). However, it is difficult for humans to understand how fully connected multilayer perceptrons explain the behavior of a given series. Guided by the *BIC* criterion, the evolutionary approach performs a substantial pruning, leading to simpler models, which are more comprehensible. For instance, from the **sunspots** neural topology of Figure 6, it is possible to infer that there is a linear influence regarding the 1, 2, 8, and 11 time lags and a moderate nonlinear effect from the other ones (3, 12, and 13).

A final remark will be given to the datasets used as benchmarks. Although being small sized (Table 1), these are the common lengths found in monthly or yearly real-world time series. Indeed, the majority of the 800 series available at the *Time Series Data Library* repository contains only a few hundred data (Hyndman & Akram, 2005). However, there are also other real-world domains that require dealing with massive datasets, especially in the presence of real-time forecasting. Examples of such applications are the heart rate of intensive care patients (Weigend & Gershenfeld, 1994) or the Internet traffic (Basu & Mukherjee, 1999). In previous work (Cortez, Allegro, Rocha, & Neves, 2002), it has been shown that even large series can be modeled by small neural networks. Thus, the proposed evolutionary approach does not present scalability problems, although the computational complexity would be a major issue if real-time forecasting (e.g., in seconds) is required.

*Figure 8. The best fitness value (BIC) obtained in each generation by the evolutionary algorithm for the **sunspots** series*



Future Trends

In the last few decades, there has been an increasing effort to enhance the decision-making process. Indeed, three major factors have contributed for a stronger emphasis in the use of forecasting methods:

- the semiconductor industry growth, leading to a high computational power at low costs;
- the exponential increase in data storage and automatic acquisition devices; and
- the development of new automatic discovery tools.

Despite the nonlinear benefits provided by the artificial neural networks, the majority of the organizations still only use conventional forecasting methods (Makridakis et al., 1998), although this scenario is expected to change, as analysts and managers become more familiar with connectionist models. Nowadays, the major argued neural network forecasting hurdles are the lack of understandability, due to black-box modeling, and the complexity of use, when compared to other methods. Yet, these claims will fade as research continues in explanatory knowledge, such as the extraction of rules from trained networks (Setiono, 2003); and model selection methodologies, such as the one presented in this chapter.

Regarding the neural network architectures, multilayer perceptrons will remain a popular choice since they are available in several off-the-shelf packages. Nevertheless, other types of networks are expected to gain importance, such as radial-basis functions and recurrent networks. More recently, there has been a growing interest in use of *support*

vector machines, nonlinear techniques that present theoretical advantages (e.g., absence of local minima) over neural networks. For instance, Cao and Tay (2003) have applied support vector machines to financial series, obtaining competitive results when compared with back-propagation networks.

Another potential non-conventional approach to forecasting relies in the use of *evolutionary computation*, which is expected to increase in importance, motivated by advantages such as explicit model representation and adaptive global search. Currently, there are two promising approaches: first, the optimization of traditional forecasting methods, such as the *ARMA* model (Equation 4), with *evolutionary algorithms* based on real value representations (Cortez, Rocha, & Neves, 2004); second, the use of *genetic programming*, by building a numerical expression made of a set of time lags and operators, such as the ones taken from the alphabet {'+','-','*' and '/'} (Kaboudan, 2003).

This chapter presented a direct encoding approach to neural network design, being the connection weights set by the *RPROP* algorithm. Yet, these gradient-based procedures are not free from getting trapped into local minima (Rocha, Cortez, & Neves, 2003). One possibility to alleviate this problem would be the simultaneous evolution of architectures and connection weights (Yao, 1999). Evolutionary algorithms are attractive for weight training because they perform a multi-point global search. Furthermore, it is possible to combine both evolutionary and gradient search for neural training by using a Lamarckian optimization (Rocha et al., 2003).

In the future, will it be possible to achieve a universal method to predict any kind of time series? As pointed out by Weigend and Gershenfeld (1994), the answer is negative and it is related to Hilbert's vision of reducing all mathematics to a set of well-defined axioms, a dream that was ended with Gödel's theorem. As an alternative, Weighend and Gershenfeld propose a more realistic goal, to find the best model for a given series, where *best* means the model that requires less information to explain correctly a given phenomenon. It is hoped that future developments in *nonlinear systems*, *computer science,* and *artificial neural networks* will contribute toward solving this objective.

Conclusion

The surge of connectionist techniques, such as multilayer perceptrons, has created new possibilities for the field of forecasting. Currently, the application of these methods requires some effort from an analyst, in processes such as data analysis and model selection. In this chapter, an adaptive approach is presented, assuming no prior knowledge over each series. Moreover, the proposed system works automatically, performing model selection on the fly, being able to choose among distinct multilayer perceptrons, ranging from linear models to complex nonlinear ones. However, this added autonomy has the handicap of increasing the computational complexity.

Comparative experiments, among conventional (e.g., Holt-Winters & Box-Jenkins) and connectionist approaches, with several real and artificial series from different domains, were held. These have shown that the Holt-Winters method, although very simple,

presents a good performance on linear series with seasonal and trended components. However, when the domain gets more complex, with nonlinear behavior, the traditional methods are clearly not appropriate. The proposed neural approach shows its strength exactly in these scenarios.

On the other hand, it was possible to confirm that neural networks, such as the multilayer perceptrons, are indeed very powerful tools for regression tasks, although they rely heavily on the network design. Poor structures provide insufficient learning capabilities, while ones that are too complex lead to overfitting. One promising possibility for topology selection is to use a hybrid combination of evolutionary and neural procedures. Indeed, the genetic algorithm, here taken as the main engine, proves to be a powerful model selection tool. Regarding the evaluation of the neural forecasting generalization capabilities, the *BIC* statistic, which penalizes complexity, has revealed itself as an adequate solution. This criterion also presented the advantage of demanding a simple computation while increasing the selection pressure in favor of simpler structures, which are more comprehensible.

In the future, it is intended to apply the evolutionary neural networks to other forecasting types, such as *long-term*, *multivariate,* or *real-time* prediction, in real-world applications (e.g., *bioengineering* or *Internet traffic*). Although the proposed methodology obtained interesting results in the forecasting arena, it is potentially useful in other domains where multilayer perceptrons can be applied (e.g., *classification tasks*). Finally, it also is aimed to explore the evolutionary optimization with different neural architectures (e.g., recurrent neural networks).

References

Basu, S., & Mukherjee, A. (1999). Time series models for Internet traffic. *Proceedings of the 24th Conf. on Local Computer Networks* (pp. 164-171).

Box, G., & Jenkins, G. (1976). *Time series analysis: Forecasting and control*. San Francisco: Holden Day.

Cao, L., & Tay, F. (2003). Support vector machine with adaptive parameters in financial time series forecasting. *IEEE Transactions on Neural Networks, 14*(6), 1506-1518.

Chatfield, C. (1993). Neural networks: Forecasting breakthrough or passing fad? *International Journal of Forecasting, 9,* 1-3.

Chen, S., & Lu, C. (1999). Would evolutionary computation help in designs of ANNs in forecasting foreign exchange rates? *Proceedings of the 1999 Congress on Evolutionary Computation* (vol.1, pp. 267-274).

Cortez, P., Allegro, F., Rocha, M., & Neves, J. (2002). Real-time forecasting by bio-inspired models. In M. Hamza (Ed.), *Proceedings of the 2nd International Conference on Artificial Intelligence and Applications (AIA)*, Málaga, Spain (pp. 52-57).

Cortez, P., Rocha, M., Machado, J., & Neves, J. (1995). A neural network based forecasting system. *Proceedings of the ICNN'95 - IEEE Int. Conf. on Neural Networks*, Perth, Western Australia (vol. 5, pp. 2689-2693).

Cortez, P., Rocha, M., & Neves, J. (2004). Evolving time series forecasting ARMA models. *Journal of Heuristics*, *10*(4), 415-429.

Falco, I., Cioppa, A., Iazzetta, A., Natale, P., & Tar, E. (1998). Optimizing neural networks for time series prediction. *Proceedings of the Third World Conference on Soft Computing (WSC3)*.

Faraday, J., & Chatfield, C. (1998). Times series forecasting with neural networks: A case study. *Applied Statistics, 47,* 231-250.

Fayyad, U., Piatetsky-Shapiro, G., & Smyth, P. (1996). The KDD process for extracting useful knowledge from volumes of data. *Communications of ACM, 39*(11), 27-34.

Flexer, A. (1996). Statistical evaluation of neural networks experiments: Minimum requirements and current practice. *Proceedings of the 13th European Meeting on Cybernetics and Systems Research*, Vienna, Austria (vol. 2, pp. 1005-1008).

Grau, F. (1992). Genetic synthesis of boolean neural networks with a cell rewriting developmental process. In D. Whitley & J. Schaffer (Eds.), *Proceedings of COGANN*, Los Alamitos, CA (pp. 55-74). IEEE.

Hanke, J., & Reitsch, A. (1989). *Business forecasting*. MA: Allyn and Bancon.

Harvey, A. (1989). *Forecasting, structural time series models and the Kalman filter*. Cambridge: C. U. P.

Haykin, S. (1999). *Neural networks: A comprehensive foundation*. Prentice-Hall.

Holland, J. (1975). *Adaptation in natural and artificial systems*. Unpublished doctoral thesis, University of Michigan, Ann Arbor.

Huang, W., Xu, B., & Chan-Hilton, A. (2004). Forecasting flows in Apalachicola River using neural networks. *Hydrological Processes, 18,* 2545-2564.

Hyndman, R., & Akram, M. (2005). *Time series data library*. Retrieved from www-personal.buseco.monash.edu.au/~hyndman/TSDL/

Kaboudan, M. (2003). Forecasting with computer-evolved model specifications: A genetic programming application. *Computers & Operations Research, 30,* 1661-1681.

Kwok, T., & Yeung, D. (1999). Constructive algorithms for structure learning in feedforward neural networks for regression problems: A survey. *IEEE Transactions on Neural Networks, 8*(3), 630-645.

Lapedes, A., & Farber, R. (1987). *Non-linear signal processing using neural networks: Prediction and system modelling*. Tech. Rep. LA-UR-87-2662, Los Alamos National Laboratory.

Makridakis, S., Wheelwright, S., & Hyndman, R. (1998). *Forecasting: Methods and applications* (3rd ed.). John Wiley & Sons.

Michalewicz, Z. (1996). *Genetic algorithms + data structures = evolution programs* (3rd ed.). Springer-Verlag.

Neves, J., Rocha, M., Rodrigues, H., Biscaia, M., & Alves, J. (1999). Adaptive strategies and the design of evolutionary applications. In W. Banzhaf et al. (Eds.), *Proceedings of the Genetic and Evolutionary Computation Conference*, Orlando, FL (pp. 473-479). Morgan Kaufmann.

Opitz, D., & Shavlik, J. (1994). Using genetic search to refine knowledge-based neural networks. In W. Cohen & H. Hirsh (Eds.), *Proceedings of the 11th International Conference on Machine Learning.* San Francisco: Morgan Kaufmann.

Peitgen, H., Jürgens, H., & Saupe, D. (1992). *Chaos and fractals: New frontiers of science*. New York: Springer-Verlag.

Riedmiller, M. (1994). Supervised learning in multilayer perceptrons: From backpropagation to adaptive learning techniques. *Computer Standards and Interfaces*, 16.

Rocha, M., Cortez, P., & Neves, J. (2003). Evolutionary neural network learning. In F Pires & S. Abreu (Eds.), *Progress in artificial intelligence*, LNAI 2902 (pp. 24-28). Springer.

Rycroft, R. (1993). Microcomputer software of interest to forecasters in comparative review. *International Journal of Forecasting*, 531-575.

Sarle, W. (1995). Stopped training and other remedies for overfitting. *Proceedings of the 27th Symposium on the Interface of Computer Science and Statistics* (pp. 352-360).

Schwarz, G. (1978). Estimating the dimension of a model. *The Annals of Statistics, 6,* 461-464.

Setiono, R. (2003). Techniques for extracting classification and regression rules from artificial neural networks. In D. Fogel & C. Robinson (Eds.), *Computational intelligence: The experts speak* (pp. 99-114). IEEE Press/Wiley Interscience.

Shoneburg, E. (1990). Price prediction using neural networks: A project report. *Neurocomputing*, *2,* 17-27.

Siddiqi, A., & Lucas, S. (1998). A comparison of matrix rewriting versus direct encoding for evolving neural networks. *Proceedings of the IEEE Int. Conf. on Evolutionary Computation* (pp. 392-397). Piscataway, NJ: IEEE Press.

Tang, Z., & Fishwick, F. (1993). Feed-forward neural nets as models for time series forecasting. *ORSA Journal of Computing, 5*(4), 374-386.

Thimm, G., & Fiesler, E. (1995). Evaluating pruning methods. *Proceedings of the International Symposium on Artificial Neural Networks*, Taiwan, Republic of China (pp. 20-25).

Tong, H. (1983). Threshold models in non-linear time series analysis. *Springer lecture notes in statistics 21.* Springer-Verlag.

Ulbricht, C. (1994). Multi-recurrent networks for traffic forecasting. *Proceedings of AAAI'94 Conference*, Seattle, WA.

Weigend, A., & Gershenfeld, N. (1994). *Time series prediction: Forecasting the future and understanding the past* (pp. 3rd ed.). Addison-Wesley.

Winters, P. (1960). Forecasting sales by exponentially weighted moving averages. *Management Science, 6,* 324-342.

Yao, X. (1999). Evolving artificial neural networks. *Proceedings of the IEEE, 87*(9), 1423-1447.

Chapter IV

Development of ANN with Adaptive Connections by CE

Julián Dorado, University of A Coruña, Spain

Nieves Pedreira, University of A Coruña, Spain

Mónica Miguélez, University of A Coruña, Spain

Abstract

This chapter presents the use of Artificial Neural Networks (ANN) and Evolutionary Computation (EC) techniques to solve real-world problems including those with a temporal component. The development of the ANN maintains some problems from the beginning of the ANN field that can be palliated applying EC to the development of ANN. In this chapter, we propose a multilevel system, based on each level in EC, to adjust the architecture and to train ANNs. Finally, the proposed system offers the possibility of adding new characteristics to the processing elements (PE) of the ANN without modifying the development process. This characteristic makes possible a faster convergence between natural and artificial neural networks.

Introduction

Nature has shown to be a highly efficient method for problem-solving, so much so that countless forms of life — in the end, countless solutions for survival problem — make their way in the most complex and diverse conditions.

Big steps in several science fields have been achieved by means of the imitation of certain mechanisms of nature. An example of this might be the Artificial Neural Networks (ANNs) (Freeman & Skapura, 1993), which are based on brain activity and are used for pattern recognition and classification tasks. Another example of this symbiosis between science and nature is Evolutionary Computation (EC) (Holland, 1975; Bäck, 1996), whose techniques are based on evolution by natural selection and regarding a population of potential solutions for a given problem where both, mutation, and crossover operators are applied. EC techniques are mainly used at fitness and optimisation tasks.

ANNs are currently the most suitable of the artificial intelligence techniques for recognition patterns. Although this technique is not entirely free of problems, during several decades it has been offering solvent systems that have been successfully transferred to the industrial environment.

ANNs internal structure consists of a series of processing elements (PE) which are interconnected among them, same as biological neurons do. The ability of ANNs for problem-solving lies on, not only the type and number of PE, but also the shape of the interconnection. There are several studies about the development of PE architectures but also about the optimisation of ANNs learning algorithms, which are the ones that adjust the values of the connections. These works tackle the two main current limitations of ANNs, due to the fact that there is not mathematical basis for the calculation of the optimal architecture of an ANN, and, on the other hand, the existing algorithms for ANN learning have, in some cases, convergence problems so that the training times are quite high.

Some of the most important ANNs are those as recurrent ANNs (RANNs) (Haykin, 1999) that tackle temporal problems, quite common in the real world, and different from the classical type of problems of non-temporal classification performed by static ANNs. However, the difficulties for their implementation induced the use of tricks as delays (TDNN) or feed-forward networks from recurrent networks (BPTT) for solving dynamic problems. On reaching maturity, the recurrent ANNs that use RTRL work better with dynamic phenomena than the classical ones, although they still have some problems of design and training convergence.

With regards to the first of these problems, the architectural design of the network — in both ANN models, feed-forward and Recurrent — the existence of a vast amount of design possibilities allows experimentation but also sets out the doubt about which might be the best of combinations among design and training parameters. Unfortunately, there is not a mathematical basis that might back the selection of a specific architecture, and only few works (Lapedes & Farber, 1988; Cybenko, 1989) have shown lower and upper limits for PE number at some models and with restricted types of problems. Apart from these works, there are only empirical studies (Yee, 1992) about this subject. Due to this situation, it cannot be said for sure that the architecture selected is the most suitable one without performing exhaustive architectural tests. Besides, nowadays there is a clear

dichotomy between recurrent and nonrecurrent ANNs regarding, not only the problems for what they are used but also the way in which network training is performed in each case.

In order to palliate this problem and to make the design process automatic, several fitness methods for architecture have been developed for certain ANN models. Some of these methods are based on connection pruning (Gomm, Weerashinghe, & Willians, 1998; Setiono, 1997) and some others increment or decrement the PE number — overall at recurrent type (Martinetz, Berkovich, & Schulten, 1993; Fritzke, 1995). All these methods fall in local minimums because networks create new PE as they receive different inputs and they are quite dependent from the network initial stage and also from the order in which training patterns receive.

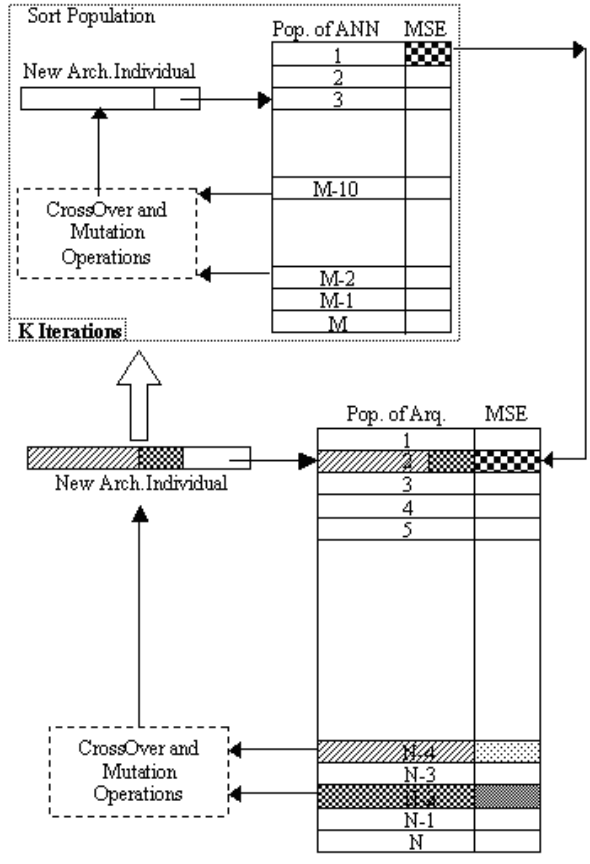
Regarding the training of the ANNs, the classical algorithms, which are based on the gradient descent, are quite sensitive to local minimums of the search space. Moreover, in order to achieve the network convergence, the designer has to configure another set of parameters that are involved in training, such as learning rate and individual parameters that belong to every algorithm. Another intrinsic problem of all learning algorithms is that they are not easily adaptable to the working modifications of both, PE and connections; therefore, this would prevent the development and the implementation of new characteristics and improvements at ANNs models. One of the possible improvements might be the incorporation of biological characteristics, similar to those of natural neural cells, for a better understanding and functionality of the artificial neurons.

A possible solution to these problems is the use of new optimisation techniques. Genetic Algorithms (GA) (Fogel, Fogel, & Porto, 1990; Yao, 1992) are easy-functioning optimisation techniques of EC that achieve good results. They have been applied, as well as other EC techniques, to the architectural adjustment for years (Robbins, Hughes, Plumbley, Fallside, & Prager, 1993), so that they represent an open field for investigation.

The works regarding ANNs training by means of GA are quite previous (Whitley, Starkweather, & Bogart, 1990). This training method has some advantages over traditional gradient-based ones because it is less sensitive to local minimums owing to its better sampling of the search space of all the weights of the network connections. The two training approaches might be mixed in a Lamarkian GA, in such a way that genetic operators are first applied to the population for obtaining the offspring of every generation, and after this, several training cycles with gradient descent algorithm are performed prior to the evaluation of every new individual (or set of weights). This approach makes good use of GA exploration and avoids the problem of local maximums by means of the use of gradient descent methods.

Finally, the training by means of GA allows modifying PE performance and connections at one's discretion with the only alteration of the evaluation function. In this case, the performance of the network elements does not affect that of the learning algorithm. This independence allows the incorporation of new characteristics, which was quite difficult when using the gradient algorithms.

Figure 1. General diagram of system performance



Proposal

This paper proposes a system design based on a multilevel design with progressive fitness that might be able to work as natural neural networks do.

The proposed scheme is based on the use of several levels of EC techniques (Holland, 1975; Bäck, 1996; Bethke, 1981; Bramlette, 1991). Specifically GA are going to be used at two levels in order to achieve, in a rapid and efficient way, the combination of parameters for both, architecture at the upper level and training at the lower level.

Another aspect that is considered is the incorporation of new characteristics to the ANNs. In order to validate that the model envisages this possibility, a modification regarding PE performance is proposed. The fact of considering PE only as a node that provides an output, which depends on the input values, is a vast simplification of the behaviour of biological neurons. In this work, with the aim of increasing ANNs potentiality, and by means of the study of biological evidences, it is proposed to add a temporal

component to the activations as a modification for PE performance. These modifications try to mimic the excitation at the natural neuron, which is induced by the action potential and subsequent refractory periods. Besides, this modification is associated with a higher complexity for ANN development and training, which is due to the addition of new parameters that influence the performance of the system.

A prediction of classical time series at statistical field was carried out in order to validate the performance of the proposed modifications. These results were compared with those obtained when prediction had been done after series characterisation with ARIMA models — *autoregressive integrated moving average* (Box, 1976; Wei, 1990).

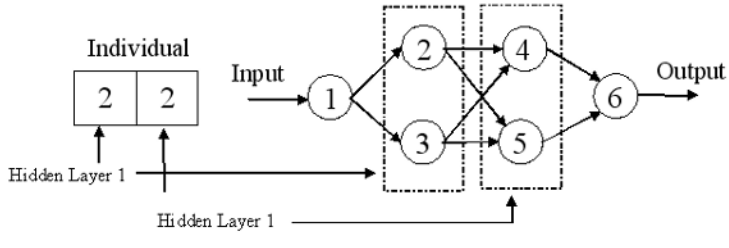
ANNs Multilevel Development with Time Attenuated Activation

Automation of Topology Development

The system shown here tries to be as much generic as possible, since it works with feed-forward ANNs as well as with RANNs. The study of the feed-forward ANNs, regards their connectivity bearing in mind different configurations of hidden layers considering their number of neurons (Whitley et al., 1990). As far as RANN are concerned, as layers do not exist, the architecture adjustment will involve the calculation of the optimal number of hidden neurons and their connectivity. The connective matrices are upper triangular at feed-forward ANNs but the whole of the matrix is stored in order to keep compatibility with RANN. It should be kept in mind that the presence of the connective matrix affects the evaluation method of ANNs, as the calculation of inputs for every element of the process has to consider the connective matrix that is associated to every network.

The system uses a GA for the design of the architecture adjustment. This GA presents marked differences with standard GA due to the special configuration of the problem to be solved (Figure 1, population of architectures). First, the information codified about

Figure 2. Structure encoding of layers and neurons at a nonrecurrent ANN



the ANN (architecture, connectivity, and PE activation functions) has no fixed length; it depends on the type of ANN. Thus, it is necessary to use a GA with individuals of variable length. Second, it is necessary to adjust three types of parameters in simultaneous form. This makes modifications in the crossover and mutation operators. They work with individuals of three different types of characteristics. Third, the evaluation of the individuals has to be realized facing the ANN with the problem trying to be solved (Figure 1, population of ANNs).

GA Configuration

Individual Encoding

It is necessary to distinguish between feed-forward networks and RANN. At the first ones, both numbers of hidden layers and of PE per layer should be configured. To achieve this, the GA codifies the individuals with an array in which every position represents the number of neurons of a hidden layer (see Figure 2). Obviously, these individuals would have variable length, as there will be individuals with a different number of hidden layers. A maximal value for number of layers will be configured in order to avoid the development of the ANN to an unbalanced size. At RANN, the individuals only have a position, which regards the number of hidden neurons.

In every individual, a connectivity matrix would be among the neurons. The matrix indicates which connections are going to be adjusted during the training period, so that "one" means that connection exists, whereas "zero" means that it does not (zero value for connection weight). Feed-forward ANNs use upper triangular matrix, which allows connections among neurons of non-consecutive layers only if these connections are toward subsequent layers.

Figure 3. Connectivity at a feed-forward ANN

PE	1	2	3	4	5	6
1	0	1	1	0	0	1
2	0	0	0	1	0	0
3	0	0	0	1	1	1
4	0	0	0	0	0	1
5	0	0	0	0	0	1
6	0	0	0	0	0	0

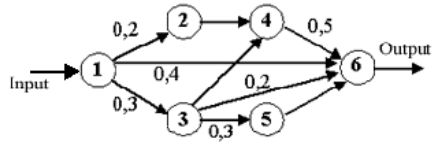
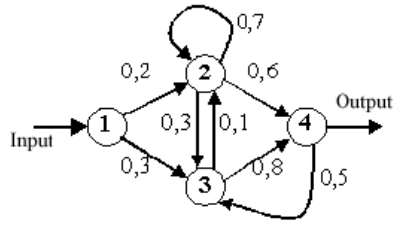


Figure 4. Connectivity at a RANN

PE	1	2	3	4
1	0	1	1	0
2	0	1	1	1
3	0	1	0	1
4	0	0	1	0



As it is shown in Figure 3, the matrix positions (1,6) and (3,6) link elements of non-consecutive layers. The positions that are attenuated at the matrix of Figure 3 are not possible as they represent recurrent connections and the network that is being used is a feed-forward one. Lastly, in those positions whose values are not attenuated, all possible combinations can be used, so the GA would eliminate those non-viable architectures without connections with both, output neurons or input ones.

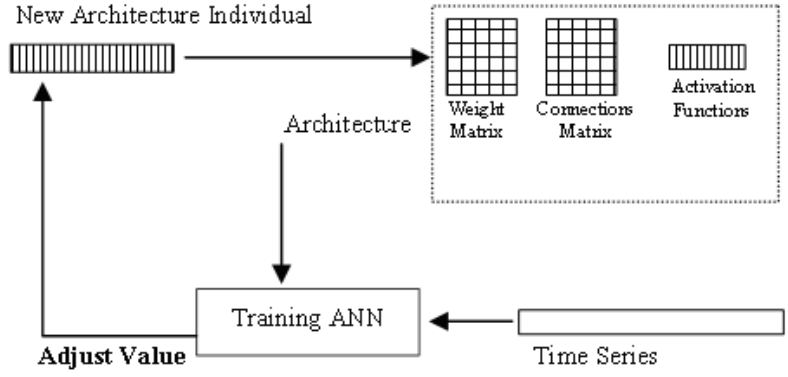
The RANN connectivity is specified by means of the complete matrix. In this case, there are not connectivity restrictions, therefore there are not connections with attenuated values. Here, the matrix allows the selection of significant connections toward precedent and subsequent neurons, as well as toward themselves (see Figure 4).

The matrix for activation functions should have as many positions as neurons in order, not only to specify the type of function that is going to be used (lineal L, sigmoid S, hyperbolical tangent H, or threshold U), but also to store those parameters referred to the lineal activation function (the slope of the straight line) and to the threshold (the threshold value).

Crossover and Mutation Operators

The individuals are composed of three parts: number of neurons, connections, and activation function for each neuron. These three parts are codified together in a chromosome of variable length. The crossover operation supposes that these three parts are independent and then it needs three cross-points in order that every descendant may inherit part of the genetic information of every progenitor in each of three concepts. So, it is similar to apply three common crossover algorithms simultaneously, one for each type of information. Each of the crossovers would be performed using one individual randomly chosen from the complete population with another from the best half part of

Figure 5. Evaluation scheme for an ANN architecture



the population. This is done in order to avoid the homogenisation of the population due to the continuous crossover of best individuals. This strategy has achieved the best results of all since it does not take advantage of best-adapted individuals and it makes good use of any individual of the population. Both individuals are evaluated during the last phase, and only the best descendant is included in the population, to be precise in the space that leaves the worse adapted individual of the origin population after its elimination.

The mutation operator applies modifications over certain number of individuals of the population in random positions. Nevertheless, in this case it is necessary to bear in mind that three parts of the chromosome use different ranges of values. In the first part, the genes represent the number of neurons and there are a maximum number of hidden neurons. In the second part, the genes represent the connectivity between the PEs, and only they can take two values, 0 or 1. Lastly, the third part represents the activation functions of each neuron and only can take one of four types implemented in the system.

Due to these peculiarities, and after randomly choosing the gene that is going to be mutated within an individual, it is necessary to check the zone where it belongs in order to generate a value within the correspondent range of values. After this, the mutated individual is reevaluated and relocated in the population, as it is usual regarding the mutation operator.

Evaluation

For the evaluation of the GA individuals, the algorithm performs the training of a RANN that fulfils the restrictions of neurons, number of PEs, connections, and activation functions of every individual of the population to verify the level of adjustment that is

produced opposite to the resolution of a concrete problem. This information codified in the individuals of the GA of the design level pass as entry to the training process by another GA. The fitness value obtained in the training process will be used as measurement of adjustment in the GA of the design level.

The training GA always simulates the same number of generations for the evaluation of all the architectures to be comparable. Once this number has been achieved, the error value obtained by the ANN during the training period is going to be the fitness level of the represented architecture.

It should be considered that trained ANNs do not need to reach a level of minimal error during the evaluation of the architectures. The ANNs' reduced training is used to achieve a representative value of the possibilities of a given architecture. In this case, it is more important to fix a finite goal, as the number of simulation cycles, for the time of training. Fixing an error level as a goal would not be operative, since the architecture might not achieve that level, and therefore the process of individual evaluation would be paralysed.

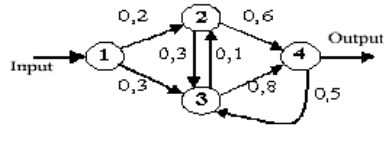
ANN Training by Means of a GA

As it has been previously commented, the developed system, in order to evaluate its individuals, needs to train ANNs. This is achieved by means of the development of another GA, which will adjust the connection weights. This GA uses a fixed ANN architecture (regarding number of layers and neurons per layer, as well as connections and activation functions) and the data of the problem.

At this point, the problem lies in the optimisation of the weights of an ANN, aiming to achieve the resolution of a given problem. Therefore, the genes of the individuals would be represented by a set of connection weights. A key point for encoding is that, as the

Figure 6. Weight matrix and the ANN that represents

PE	1	2	3	4
1		0,2	0,3	
2			0,3	0,6
3		0,1		0,8
4			0,5	



system is able to train feedforward as well as recurrent ANNs, it is necessary to choose a representation that might support both types of networks, so that the same type of genetic operators might be used with them.

Following the previous idea, every ANN is represented by an NxN matrix, where N is the total number of neurons at the ANN, no matter the layer they belong to. This matrix contains weight values only for positions that are 1 at the connection matrix, which is fixed for the training of a specific network. The cell with (i, j) position will be the weight for the connection between the output of *i* neuron and the input of *j* neuron. If two neurons (j, k) do not have direct connection because there is a zero at the connection matrix, the matrix value for the position (j, k) also would be zero.

The weight matrix for a feedforward ANN is always upper triangular because there are not connections with neurons of the same layer or to the previous ones. The RANN uses the whole of the matrix of connections to describe the forward and recurrent connections.

Crossover Operator

As every individual consist of a matrix, the crossover operator should be modified for a uniform mixture of the characteristics of the progenitors at the offspring during the crossover operation.

In cases like these, the mentioned operator has to be adapted for performing its task over a bi-dimensional matrix. A possible option for operator application would be choosing a matrix cell that might be considered as a cross point (see Figure 7). The previous positions and rows to that cell and to its row would belong to an individual of the offspring, whereas subsequent rows and positions would be for another descendant. It also might be possible to choose a row as a cross point in a more general case.

The two previous options, using a row or a position, induce a horizontal division at the matrices that are going to be crossed. As it is shown at Figure 8, the weight values of non-recurrent ANNs form groups at the upper part of their matrix due to the lack of recurrent connections and also the higher number of connections with the first hidden layer.

Due to the last and in order to perform a more homogeneous crossover, the matrix is divided into columns, whose crossover would be done as it was explained with rows (see Figure 9).

Figure 7. Crossover operator for weight matrices

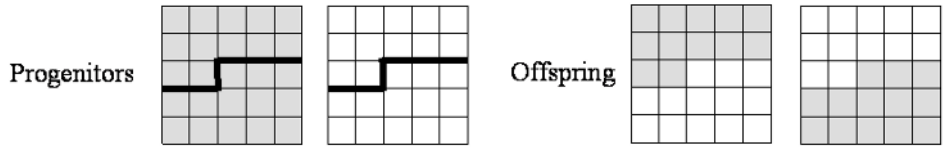


Figure 8. Example of weight matrix for non-recurrent ANN

PE	1	2	3	4
1		0,2	0,3	
2				0,6
3				0,8
4				

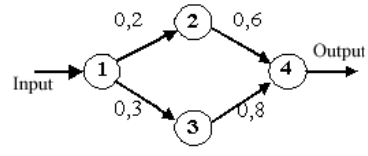
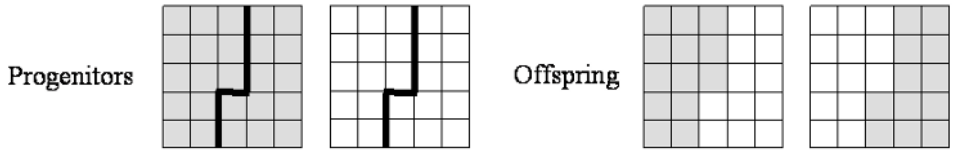


Figure 9. Vertical crossover operator for weight matrices



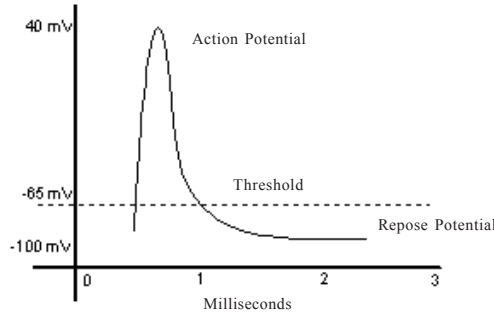
Adding Biological Characteristics

The Natural Neurone

The activation output of the neurone cell is explained from the electrical characteristics of the cell's membrane. This membrane is more permeable to the potassium ions (K+) than the sodium ions. The potassium chemical gradient leaves the potassium ions of the cell by diffusion, but the strong attraction of the organic ions tend to maintain the potassium in. The result of these opposite forces is an equilibrium where there are more sodium and chlorine ions (Cl-) out and more organic and potassium ions in. This equilibrium makes a potential difference through the cell's membrane of 70 to 100 mV (millivolts), the intracellular fluid being negative. This potential is called cell's repose potential.

The influences of the excitatory inputs that arrive to the cell from another are added in the axonal join and cause a successive depolarisation of the membrane. It is then that the potential gets inverted up to +35 mv in only 0'1 milliseconds. The resultant depolarisation alters the permeability of the cellular membrane relating to sodium ions. As a result, there

Figure 10. Unchaining of the action potential in the neurone's membrane



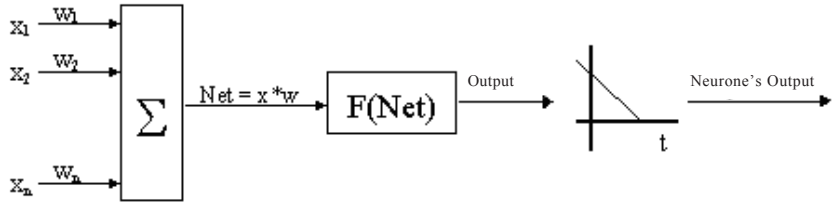
is an incoming flux of positive sodium ions that enter the cell influencing the depolarisation even more. This self-generated effect causes the action potential (Figure 10). After an action potential, the potential of the membrane becomes more negative than the repose potential for some milliseconds, up to get the ion equilibrium. To begin with the action potential, a sudden increment of approximately 15 to 30 mv is required, considering -65 mv the stimulation threshold. Within an excitable fibre, a second action potential cannot be achieved while the membrane is depolarised. The period during which another action potential cannot be launched, even with a very powerful stimulus, is called "absolute refractory period" and lasts about 0'5 milliseconds. It is followed by a "relative refractory period" that lasts from 1 to 1'5 milliseconds, where a bigger stimulus than usual is needed to achieve the action potential, because the ionic equilibrium of the repose potential has not been recovered yet.

This point cannot be excited in approximately 1 millisecond, the time that it takes to recover the repose potential. This refractory period limits the transmission frequency of the nervous impulse to about 1.000 per second.

Architecture with Time Attenuated Activation

Obviously, the behaviour of the PE in the ANN is not very similar regarding the activation phase, to what we have just described. In the traditional PEs over the input function which ponders the weights with the activation of the previous layer, an activation function is applied, being the exponential and the sigmoid most used. If the output of this function is positive, then the activation occurs. In that instant, the neurones that get the activation will have as input the product of the weight of the connection and the value of the activation. This way of propagating the activation of the neurone works in a very

Figure 11. Representation of the time attenuated activation



punctual way, without giving the idea of continuity that can be seen in the natural model where the activation flows from a maximum point (action potential) to get the "zero" (repose potential). In this work, it has been added to the classical architecture of PE, the function of the action potential, approximating it by a line that goes from the output level of the activation function to the zero level.

This model of ANN changes the conception of the connections between neurones that go from being a number (an only numerical value) to be represented by a function (a line with negative slope that is characterised by the slope value).

Now, the neurone functioning has to be explained as a time function. That is, the activation of a neurone joins a series of inputs in a point of the time. This not only affects the output neurones in that instant, it affects the following neurones in *n* next instants, too (Figure 12). The value of *n* depends on the slope of that connection.

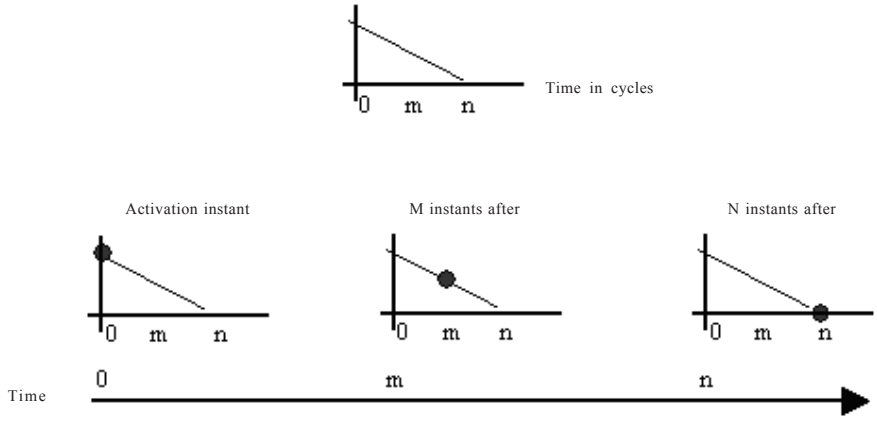
The line formula that emulates the time attenuated activation of the natural neurones is:

$$y = mx + b \tag{1}$$

In this formula, *y* represents the value of the neurone's output that is received by the following neurones. The letter *b* represents the output of the neurone's activation function. The letter *x* represents the time from the activation, and the letter m is the line slope that, as already mentioned, is always negative. At the moment when the activation occurs, the time is 0 and the output is $y = b$ from this moment and, depending on the slope, the output *y* attenuates until 0 through a time $x = t$.

When we have defined the new architecture, to apply this type of ANN to solve real problems, it is necessary for a training algorithm that adjusts the weights and, in this case, the slopes of the ANN. To this new kind of PE, the learning algorithms used until now are not valid, because we need to calculate the line slopes to each PE. GA have been used for the training of this kind of ANN.

Figure 12. Functioning of the neurone's activation



Obtaining Activation Slopes

The first step for using GA is to encode all the possible solutions as an array with **n** elements or genes, so it would represent one possible solution that is an individual within the population. In these cases, the ANN has been codified as an array of slopes or weights (recurrent or non-recurrent ANN) from the initial layer to the output one. Also, should be codified not only whether the ANN is recurrent but also all the generic parameters of the PE, such as limit weight value, limit slope value, type of activation function (lineal, threshold, exponential and tangent hyperbolic) and recurrent ANN values such as continuous training or in periods, and number of ANN iterations previous to output collection. Once the individuals have been designed, the following phase consists on generating a random population where genetic operators would be applied.

Results

In this section, the experimental results achieved are organised into three parts. First, several tests would be performed in order to check that those ANN, which use time attenuated activations, are able to train and also to generalise the information regarding the sets of both training and test. Once this goal has been achieved, the second step will involve the realisation of several manual tests for architecture adjustment, in order to compare them during the third phase by means of the automatic adjustment that is offered by the multi-level system of ANNs development.

Table 1. ANN parameters for sunspots series

Activation Time	Cycle	Activation Function	PE	Training Time	MSE
0.5	10	Exponential	4	49' 15''	**0.0165**
0.01	10	Exponential	4	49' 30''	0.0180
0.01	1	Sigmoid	4	7' 15''	0.0373
0.01	1	Exponential	7	11' 30''	0.0400

Table 2. ANN parameters for tobacco series

Activation Time	Cycle	Activation Function	PE	Training Time	MSE
0.5	10	Exponential	4	21' 30''	**0.0136**
0.01	10	Exponential	4	21' 15''	0.0206
0.01	1	Sigmoid	4	5' 15''	0.0487
0.01	1	Exponential	7	5' 00''	0.0259

Two classical time series in the statistical field were used in order to check the validity of the proposed system: the number of sunspots detected per year in the period 1701 to 1987, and the U.S. production of tobacco during the period 1871 to 1984. The first of the series has 287 values, where the last five ones are reserved for ANN test phase; the second series contains 114 cases, and the last eight ones are used for the test phase.

Time Attenuated Activations

An elitist strategy was used, where the best individual in the population is not substituted by mutation. Regarding the performance parameters, during every cycle of GA the crossover involves three pairs and the mutation affects four individuals. The number of individuals in the population is 200. To achieve these performance parameters of the GA, different options were tested with regards not only to types of individual selection for crossover, but also to crossover ratios and to mutation ones. The following ones were specifically selected due to the convergence speed of the GA.

The following tables show the parameters of the four ANNs developed to predict the values for every problem; the first two used time attenuation. The **Activation Time** field shows which cycle of ANN evaluation represents the x variable in Equation 1 (the time consumed in every cycle). The **Cycle** field represents the number of cycles that is expected since the input value is introduced until the output value can be collected. If Cycle=1, the recurrence is not considered. The PE column indicates the neuron number of the hidden layer. In the four ANNs, the upper limit value for weights is 20 and for number of training cycles is 14.000. The training time and the obtained mean squared error (MSE) represent the average of the results obtained after four different attempts using the same configuration of GA but different initial populations of ANNs.

Table 3. Comparative of real values of sunspots regarding the years 1979 to 1986 with best ANN prediction and ARIMA model

Real	ANN	% ANN Error	ARIMA	% ARIMA Error
154.6	165.88	5.93	124.7	15.73
140.4	146.76	3.34	83.33	30.03
115.9	129.62	7.22	52.71	33.25
66.6	95.44	15.17	28.21	20.20
45.9	56.59	5.62	16.09	15.68
17.9	41.47	12.40	12.68	2.74
13.4	26.55	6.92	20.99	3.99
29.3	24.86	2.33	45.24	8.39

Table 4. Comparative of real values of tobacco production regardingthe years 1980 to 1984 with best ANN prediction and ARIMA model

Real	ANN	% Error ANN	ARIMA	% Error ARIMA
1785.89	1668.13	4.99	1716.16	2.94
2063.89	1787.37	11.77	1740.14	13.77
1994.04	2003.76	0.38	1764.30	9.81
1428,90	1996.31	24.18	1788.78	15.35
1727.99	1590.33	5.84	1813.61	3.67

The training time will be used to give a general idea of the performance of each test relating to the other ones. It has no significance alone in each test, because it depends on the computer utilized for the test. In this case, the computer used is an AMD XP 1600+ with 512Mb of RAM DDR memory at 266MHz.

As can be extracted from the previous tables, in prediction problems of time series, the use or temporal attenuations in PE markedly improves ANN training, and therefore the level of error that is consequently achieved.

The fitness level of the best RANN (in bold in Tables 1 and 2) is compared with the predictions obtained by means of ARIMA models. As can be seen in Tables 3 and 4, the predictions obtained by means of RANN present higher or similar accuracy levels than those of the ARIMA models used, in this case, ARIMA(2,0,0) for sun spots and ARIMA (0,1,1) for tobacco production (Andrews & Herzberg, 1985). The error percentage is the difference, adjusted with the range of every variable, between the predicted value and the real one. Although series value prediction does not experience a drastic improvement that might definitely support RANN use in prediction of time series, the validity of its use, as is shown in the results tables, does allow approaching this problem with slight statistical knowledge.

Table 5. Training level GA parameters

Population	Crossover Rate	Mutation Ratio	Crossover Type	Substitution	Simulation
100	5%	1%	1 Point	Worst individual	25.000 iterations

Table 6. ANN parameters that achieve the best results of manual adjustment

Series	PE	Activation Function	MSE	Mean Error (ME)	Total Training Time
Sunspots	4	Lineal	0,00711	0,06208	3 min. 51 sec.
Tobacco	6	Sigmoid	0,00693	0,05995	14 min. 1 sec.

Obtaining an Optimal Architecture

Once the accurate performance of temporal activations has been checked, the results obtained with the system of automatic adjustment for architectures are presented. The result of the system would be an optimal network regarding not only number of hidden neurons and connections matrix, but also activation functions per neuron.

The manual adjustment would be intended first of all and would involve choosing a number of PE, one element of {2, 4, 6}, and one of the activation functions of {Lineal, Tangent Hyperbolic, Sigmoid}. Every combination of parameters would be tested, and the resulting architecture would be trained by means of a training module based on GA. The configuration of this GA is shown in Table 5.

For every time series, the best results of manual adjustment resulted from the combination of parameters that can be seen in Table 6.

For *sunspots* time series, the total duration of the test performed for behaviour verification of each of the combinations was **6.642 seconds** (1 h. 50 min. 42 sec.). At *Tobacco* series, **3.153 seconds** (52 min. 33 sec.) were needed for testing all the combinations.

Once the best architecture has been manually established, a new architecture would be generated by means of the procedure previously mentioned in order to compare the results obtained. The generation of this architecture would involve 200 iterations of the GA of the design level during every test. Although it is a low number of iterations, it is expected that the results might show the potentiality of the implemented system. The rest of the parameters of the design level GA that are going to be used are the ones in Table 7.

Obtaining the fitness value for every architecture will be done using the ANN training module by means of GA as evaluation function. The GA would simulate 500 cycles for every architecture for the training time to be fenced during the evaluation of each

Table 7. Design level GA parameters

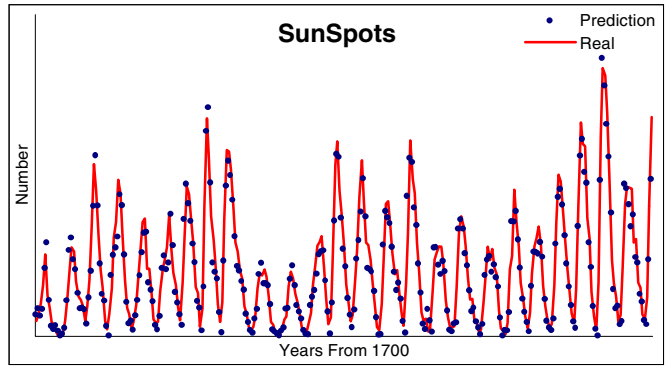
Population	Crossover Rate	Mutation Ratio	Crossover Type	Substitution
100	5%	1%	1 Point	Worst individual

individual. It is not adequate to execute the GA during a high number of cycles because it is intended to achieve an indicative value of architecture goodness, therefore, this number should be the lowest one that might allow the ANN evaluation in an acceptable period of time. However, if the value is too low, the ANN might not start the convergence process, and therefore an indicative goodness value might not be achieved. During the test performed, the starting iteration value was 100, but it was proved to not be enough because, in most of cases, no improvement was generated for that number of cycles. The value was successively incremented in 100, until there was a total number of 500 iterations per ANN. This value allows the initiation of the convergence process of the ANNs. Obviously, it is necessary to adjust this parameter for its adaptation to the problem handled in every case.

After 200 simulation cycles of the GA for the automatism of architecture design, the best individual encodes the architecture that best fits. Its fitness value corresponds to the error committed by an ANN with that architecture that has been trained during the 500 iterations already mentioned. The validity of the best architecture obtained is done through an exhaustive training that involves 25.000 iterations, in such a way that the results might be directly compared with those already achieved.

Sunspots series

Figure 13. Sunspots series versus RANN predicted values



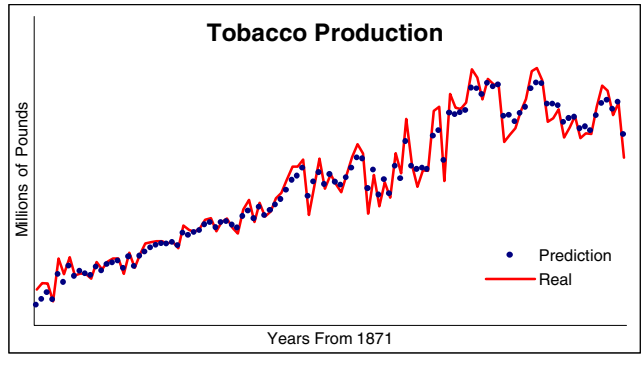
The parameters of the best-achieved solution are presented in Table 8.

Table 8. Parameters and results of the best ANN for sunspot series forecasting

Training Time	PE	Connections	Activation Functions	MSE	ME
1:23:10	6	26	L L S L L H	0,00862	0,07230

Tobacco series

Figure 14. Tobacco series versus RANN predicted values



The parameters, which determine the RANN that has achieved best results, are shown in Table 9.

Table 9. Parameters and results of the best ANN tobacco series forecasting

Training Time	PE	Connections	Activation Functions	MSE	ME
33:55	5	14	L L L L H	0,00841	0,06817

The first observation that can be done (Table 10) is that the obtained errors for both time series are slightly higher but comparable to the minimums achieved with manual configuration. The reason for this lies in the simulation period — 200 cycles — in which automatism was performed, because it is too short for a GA to entirely solve the type of problem that is being faced. However, and still with no remarkable error improvement, the GA of architecture fitness has shortened the process duration of manual configuration, which was 1 hour and 45 minutes for sunspots series and 45 minutes for tobacco series. During the simulation time of this GA, additional tasks, different from those of manual

Table 10. Comparative between manual and GA configuration

Series	Parameter	Adjust GA Parameters	Adjust Architecture	Difference
Sun Spots	Time	6.642 seconds	4.990 seconds	- 24,8%
	MSE	0,0077	0,0086	+ 11,7%
Tobacco Production	Time	3153 seconds	2.035 seconds	- 35,5%
	MSE	0,0071	0,0084	+ 18,3%

configuration, are performed such as test with higher number of neurons (between 1 and 6), connection "pruning" and automatic PE-based selection of activation functions.

Conclusions

It should be highlighted in the first place that the process involved with the design and the training of an ANN, recurrent or not, is performed using EC techniques. In that way, the designer gets out of the task of checking each of the possible combinations for the architecture as well as the different learning algorithms.

Regarding the modification of the classical structure of ANNs with the use of time attenuated activations and after the results obtained (Tables 3 and 4), it can be stated that the level of error achieved improves markedly. Therefore, it seems that the goal of helping the persistence of those internal activation states of ANN is finally achieved, which improves the flexibility of the adaptation of the ANN when solving these type of problems.

The good performance of temporal activities also shows the potentiality of this GA-based approach in order to easily incorporate the new characteristics to the performance of, not only PE or layers, but also connections, so that their functioning might be closer to that of the biological NN. The latter opens new possibilities for improving the performance of connectionist systems and also for testing new theories in ANNs' computer models.

As far as automatic design for ANN architecture is concerned, the holistic perspective presented by the system jointly approaches all the different sides of ANN development. This problem, with no supporting mathematical solution up to this moment, induces the development of non-optimised networks, due to the high amount of time required for testing all the possible network architectures. The proposed system uses a GA to adjust, not only the number of neurons and layers, but also it performs the "pruning" of the unnecessary connections and the selection of activation functions per neuron. This type of system, besides saving global simulation time, also allows the designer to distance himself or herself from the tedious task of architecture selection, so that it becomes a transparent phase of the ANN development tasks. During the test performed, apart from the automatism that the development of ANNs involves, the system also achieves results that are comparable to those obtained with ARIMA modeling of temporal series.

References

Andrews, P.E., & Herzberg, A.M. (1985). *The data: A collection of problems from statistics*. Springer-Verlag.

Bäck, T. (1996). *Evolutionary algorithms in theory and practice*. Oxford: Oxford University Press.

Bethke, A.D. (1981). *Genetic algorithms and function optimizers*. Unpublished doctoral dissertation, University of Michigan.

Box, G.E.P., & Jenkins, G.M. (1976). *Time series analysis forecasting and control*. San Francisco: Holden Day.

Bramlette, M.F. (1991). Initialization, mutation and selection methods in genetic algorithms for function optimization. In R.K. Belew & L.B. Booker (Eds.), *Proceedings of the Fourth International Conference on Genetic Algorithms* (pp. 100-107). San Francisco: Morgan Kaufmann.

Cybenko, G. (1989). Approximation by superposition of sigmoidal function. *Mathematics of Control, Signals and Systems, 2,* 303-314.

Fogel, D.B., Fogel, L.J., & Porto, V.W. (1990). Evolving neural networks. *Biological Cybernetics, 63,* 487-493.

Freeman, J., & Skapura, D.M. (1993). *Neural networks*. Boston: Addison-Wesley.

Fritzke, B. (1995). A growing neural gas network learns topologies. In G. Tesauro, D.S. Touretzky, & T.K. Leen (Eds.), *Advances in neural information processing systems* (vol. 7, pp. 625-632). Cambridge, MA: MIT Press.

Gomm, J.B., Weerashinghe, M., & Willians, D. (1998). *Pruning and extractions of qualitative fault diagnosis rules from a neural network*. Unpublished dissertation at Workshop on On-line Fault and supervision in the chemical process industries, Lyon, France.

Haykin, S. (1999). *Neural networks, a comprehensive foundation*. NJ: Prentice-Hall.

Holland, J.H. (1975). *Adaptation in natural and artificial systems*. Ann Arbor: The University of Michigan Press.

Lapedes, A., & Farber, R. (1988). How neural nets work. In D.Z Anderson (Eds.), *Neural information processing systems* (pp. 442-456). American Institute of Physics.

Martinetz, T.M., Berkovich, S.G., & Schulten, K.J. (1993). Neural-gas network for vector quantization and its application to time-series prediction. *IEEE Transactions on Neural Networks, 4*(4), 569.

Robbins, G.E., Hughes, J.C., Plumbley, M.D., Fallside, F., & Prager, R. (1993).Generation and adaptation of neural networks by evolutionary techniques (GANNET). *Neural Computing and Applications, 1*(1), 22-30.

Setiono, R. (1997). A penalty function approach for pruning feedforward neural networks. *Neural Computation, 9*(1), 185-204.

Wei, W.S. (1990). *Time series analysis. Univariate and multivariate methods*. Boston: Addison Wesley.

Whitley, D., Starkweather, T., & Bogart, C. (1990). Genetic algorithms and neural networks. Connections and connectivity. *Parallel Computing, 14,* 347-361.

Yao, X. (1992). *A review of evolutionary artificial neural networks.* Technical report Commonwealth Scientific and Industrial Research Organization.

Yee, P. (1992). *Classification experiments involving backpropagation and radial basis function networks*. Communications Research Laboratory. Report no. 249. McMaster University.

Section III

Data Mining

Chapter V

Self-Adapting Intelligent Neural Systems Using Evolutionary Techniques

Daniel Manrique, Universidad Politécnica de Madrid, Spain

Juan Ríos, Universidad Politécnica de Madrid, Spain

Alfonso Rodríguez-Patón, Universidad Politécnica de Madrid, Spain

Abstract

This chapter describes genetic algorithm-based evolutionary techniques for automatically constructing intelligent neural systems. These techniques can be used to build and train multilayer perceptrons with the simplest architecture. These neural networks are usually designed using binary-coded genetic algorithms. The authors show how the basic architectures codification method, which uses an algebra-based codification, employs a shorter string length and voids illegal architectures in the search space. The networks are trained using real number codification. The morphological crossover operator is presented and compared to other important real-coded crossover operators. The purpose is to understand that the combination of all

these techniques results in an evolutionary system, which self-adaptively constructs intelligent neural systems to solve a problem given as a set of training patterns. To do so, the evolutionary system is applied in laboratory tests and to a real-world problem: breast cancer diagnosis.

Introduction

At present, one of the main interest areas in artificial intelligence is related to the design of self-adaptive systems, which are able to transform to solve different kinds of problems or to adapt to the changing environment (Konar, 2000). Multilayer perceptrons are feedforward neural networks with one input layer, one output layer, and several hidden layers. This model is applicable within artificial intelligence to solve a wide variety of real-world problems thanks to its properties of learning by examples, generalising to unseen data, noise filtering, and pattern classification (Principe, Euliano, & Lefebvre, 2000). Despite the advantages of these intelligent neural systems, a neural architecture that does one task accurately is useless for solving another problem. A new architecture needs to be designed from scratch: structuring the network connectivity, deciding the number of hidden units, and setting the terms in the weight adjustment algorithm. This is not a straightforward job even for the most experienced practitioners, because of the size and complexity of the search space even for the best understood network models (Manrique, 2001). Under these conditions, there is no question of building a neural network-based, self-adapting intelligent system to solve any problem presented as a set of training patterns.

However, successful results have been achieved by identifying synergies between evolutionary algorithms and artificial neural networks that can be combined to design the internal structure of the network (Barrios, Manrique, Plaza, & Ríos, 2001). The way in which the neural networks that make up the search space are encoded is a crucial step in automatic network design (Hussain & Browse, 1998). Therefore, several approaches have been developed to produce efficient codifications of artificial neural networks. The first of these is the direct binary encoding of network configuration, where each bit determines the presence or absence of a single connection (Siddiqi & Lucas, 1998). This approach can be used to encode any neural architecture. Other approaches for encoding artificial neural networks are based on the binary codification of grammars that describe the architecture of the network (Hussain, 2000). The basic architectures codification method (Barrios et al., 2001) is another approach, which is based on the definition of an Abelian semi-group with a neutral element in the set of neural architectures. It rules out illegal networks codification and needs a very short encoding length, being optimum for encoding neural networks with one output. This codification encodes any kind of generalised multilayer perceptrons with one hidden layer. Another important feature is that a one-bit variation in the string that represents a network results in a very similar neural architecture, which improves the performance of the genetic algorithm. Any of these codification methods must employ an adequate crossover operator. Besides the classical operators, like the one-point, two-point, uniform, and generalised operators, the

Hamming crossover (H-X) (Barrios, Carrascal, Manrique, & Ríos, 2003), based on the concept of Hamming distance, is specialised for working with the basic architectures codification, taking advantage of its benefits to get better performance in searching for neural architectures than other binary crossover operators.

The other main drawback regarding the use of artificial neural networks is the training problem. While training feedforward neural networks with the gradient-based backpropagation with momentum factor algorithm usually yields good results, there is no guarantee that the solution it reaches is optimum. Indeed, one of the problems with this method is that both convergence speed toward the solution and the possibility of reaching a solution depend on the choice of the learning rate m and the proportionality constant of momentum a (Dai & McBeth, 1997).

Because training artificial neural networks can be seen as an optimisation problem, genetic algorithms also are a suitable option for dealing with this problem by completely or partially replacing the network learning method (Brown & Card, 2000). Michalewicz states that if a problem is real-valued in nature, then a real number genetic algorithm is faster and more precise than a binary encoded genetic algorithm (Michalewicz, 1999). This is the reason why there are different crossover operators to train artificial neural networks with real-coded genetic algorithms: Radcliffe's flat crossover (Radcliffe, 1990), BLX-a (Eshelman & Schaffer, 1993), UNDX (Ono & Kobayashi, 1997), or the morphological crossover (MX) (Barrios et al., 2003). MX has the important feature of adaptively extending (depending on progenitor values) the interval schemata defined by the parents from which the offspring are taken. It is a reinterpretation of the morphological gradient operation, very commonly used in digitalised image segmentation (D'alotto & Giardina, 1998) to get an online genetic diversity measure. This measure increases the local search speed of the genetic algorithm in the training process, which is the main drawback reported by researchers who prefer to combine genetic algorithms and gradient descent methods, while making trapping in local optima less likely.

This chapter describes evolutionary techniques for building self-adapting intelligent systems, composed of multilayer perceptrons. It is shown how, by inputting just one set of training patterns, a binary and a real-coded genetic algorithm can cooperate to output a trained multilayer perceptron with the minimum number of neurons and connections and get a very low mean square error for the given training set. Finally, comparisons are reported in terms of speed of convergence, efficiency, and accuracy between the different binary and real codification methods and the crossover operators discussed. Laboratory tests have been chosen to demonstrate the features of the different evolutionary approaches, and the breast cancer diagnosis problem is presented and solved using a multilayer perceptron generated by genetic algorithms.

Evolutionary System for Constructing Networks

This section illustrates the design of a genetic system in which a binary-coded genetic algorithm capable of designing feedforward neural architectures and a real-coded genetic algorithm that is able to train feedforward neural networks cooperate. The goal of this genetic system is to output the minimum neural network that gets a mean square error less than a previously established value for a set of training patterns that describes the problem to be solved. This set of training patterns is the system input. The binary- and real-coded genetic algorithms work in parallel (see Figure 1).

A binary codification of the neural architectures that conform the search space and a suitable crossover operator to design neural architectures will need to be chosen. The real-coded genetic algorithm that trains neural networks receives a binary-coded neural architecture from the population of the previous module for evaluation after a decoding process. This module requires a crossover operator that is capable of handling chromosomes with real numbers and finding the values of the weights for the neural connections that minimise the mean square error for the set of training patterns. The mean square error achieved after the training process is then used to calculate the fitness of the binary string that represents this neural network in the population handled by the binary genetic algorithm.

For explanatory purposes, the next two sections describe, first, different approaches to artificial neural architectures design and, then, how these architectures can be trained employing a codification with an infinite cardinal alphabet (real numbers).

Figure 1. General structure of an evolutionary system to design and train NNs

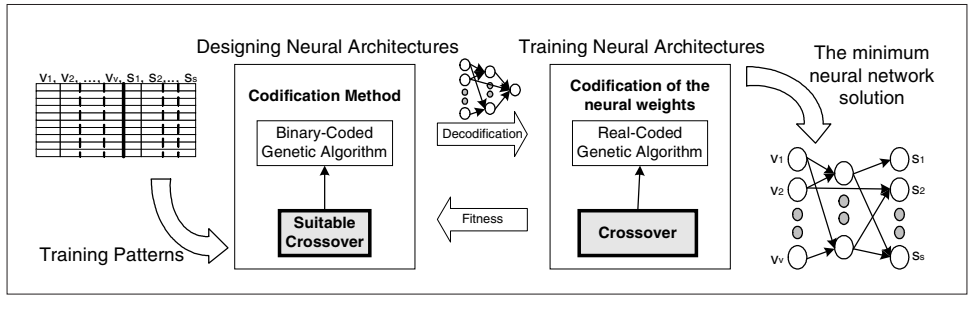
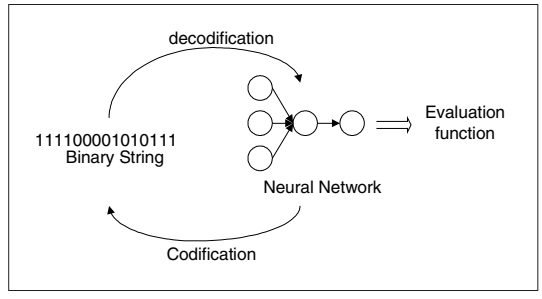


Figure 2. Basic elements required to design neural networks with genetic algorithms



Designing Neural Networks with Genetic Algorithms

The first thing to take into account when designing a neural network is that the search space has a high cardinal. This is because, depending on what problem we are dealing with, many neurons may be needed, there being, therefore, many possible combinations between them. Therefore, the search for the optimal architecture is, in principle, a very good candidate for being solved using genetic algorithms.

Using a genetic algorithm to design a neural network, like any other optimisation problem to be tackled using this technique, calls for a method of codification of the search space points, which, in this case, are neural networks. Likewise, we need the evaluation function to be optimised. It is assumed hereinafter that the optimum is the neural network that yields the minimum value of this function, which means that, given two neural networks, the network whose fitness is lower will be considered the best solution. The goal of the genetic algorithm is, therefore, to minimise the evaluation function. Figure 2 shows a diagram of these concepts, assuming that the usual binary alphabet is being used.

The fitness that is usually taken for a given string is the mean square error achieved in training the neural network output by decoding this string. This fitness value is usually weighted with other terms that take into account facts such as the complexity of the coded neural architecture. That is, individuals that represent architectures with a lot of processing elements and connections between them are penalised more than simpler architectures. This is because the simple architecture has a higher execution speed and greater generalisation capability.

Once all the individuals have been assessed, the genetic reproduction, crossover, and mutation operators are applied to output the new generation (population), repeating these steps until the population converges toward the optimum neural architecture for solving a given problem.

Evaluation Function

The choice or construction of an evaluation function that yields a fitness value for a given individual in the population that codes a neural topology is a relatively straightforward process. The most commonly used evaluation function *f* is based on calculating the mean square error of the network in the training process, weighted by other terms that take into account the number of connections in the output architecture. One of these evaluation functions is the following, which usually outputs the smallest neural architectures: $f = MSE_{it} (C_a / C_t)$, where MSE is the mean square error obtained in the training process after it iterations, C_a is the number of connections in the actual neural architecture, and C_t is the total number of possible connections permitted by the codification used.

Neural Architectures Codification Methods

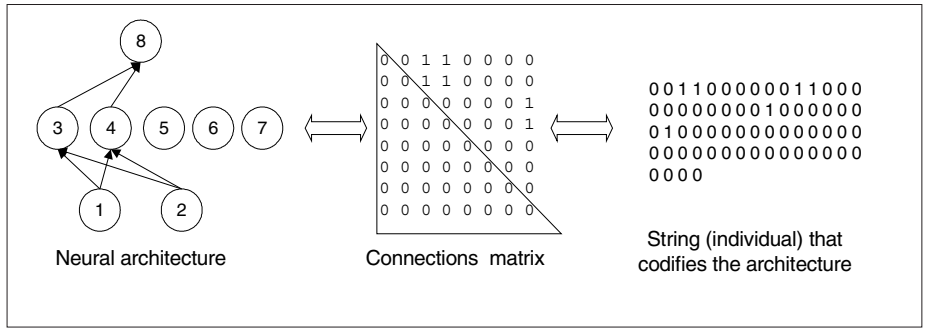
The most commonly used codification for representing the neural network architecture is the *direct codification method* (Bornholdt & Graudenz, 1992), which explicitly represents each connection by means of a binary digit. Given an architecture with N neurons, the matrix that represents the set of connections in the network is termed matrix W, of dimensions NxN:

$$W = \begin{pmatrix} w_{11} & w_{12} & \cdots & w_{1N} \\ w_{21} & w_{22} & \cdots & w_{2N} \\ \cdots & \cdots & \cdots & \cdots \\ w_{N1} & w_{N2} & \cdots & w_{NN} \end{pmatrix}$$

where $w_{ij} = \{0,1\}$, $\forall i, j = 1, 2, ..., N$. The values w_{ij} represent the presence or absence of connections. If $w_{ij} = 1$, then there exists a connection from the i^{th} neuron to the j^{th} neuron of the network. On the other hand, if $w_{ij} = 0$, then there is no connection between these two neurons. If the matrix is composed of zeros along and below the main diagonal, the neural architecture of the represented network is feedforward. If, on the other hand, there are elements other than zero, it is a feedback neural architecture. Figure 3 illustrates a feedforward neural network, its connections matrix and the binary string that represents its architecture. This codification method employs N^2 bits to codify feedback architectures with N neurons, while the number of bits required to encode feedforward architectures can be reduced to $N \cdot (N-1) / 2$.

This approach provides a very fast and easy to implement process of encoding and decoding neural networks as binary strings and vice versa. However, its drawback is that the bits required for codification is a square number. Therefore, the neural architecture to be encoded is sizeable, the connection matrixes are very large, and it takes the genetic algorithm a very long time to process them.

Figure 3. Direct codification of neural architectures



Other codifications have been designed to reduce the size of the individuals of the population that the direct codification methods generate, of which the *indirect codification methods* are very important (Kitano, 1994). They are based on the idea of using an intermediate representation between the connections matrix and the string that encodes the architecture.

A prominent member of the indirect codification methods of neural architectures is the *grammar codification method* (Kitano, 1994), which involves describing the neural architecture by means of a set of derivation rules belonging to a grammar and then encoding these rules by means of binary strings. The intermediate representation chosen from the neural architectures and the strings that encode them is, in this case, the set of grammar rules. This manages to reduce the length of the strings that form the population as compared with the direct codification method. The grammar codification method is composed of two parts: (1) construction of the grammar whose words are the set of neural architectures that are to be used, that is, the search space; and (2) the codification of the derivation rules of the grammar by means of binary strings.

For feedforward neural networks with two input neurons, five hidden neurons and one output neuron, the grammar encoding method needs 40 bits, whereas the direct codification method takes 64 bits. The drawback of this approach is that it is very tedious to define a grammar each time a new neural architecture has to be built.

The *basic architectures codification method* (Ríos, Barrios, Carrascal, & Manrique, 2001) is another important approach that can be classified as an indirect codification method. It is based on an algebraic formalisation of the set of feedforward neural architectures with one hidden layer. This algebraic description outputs a binary codification in which any neural architecture is represented by a string of bits of a shorter length than the above two encoding methods.

This codification method denotes the set of all generalised feedforward neural networks with a maximum of I input neurons, H hidden units, and O output units by $R_{I,H,O}$. Of the set of all existing neural networks $R_{I,H,O}$, we are only interested in the subset $V_{I,H,O} \subseteq R_{I,H,O}$ of all valid neural networks, as $V_{I,H,O}$ only contains points that can solve a problem and does not include illegal neural networks. Given two valid neural networks v and v', the

Figure 4. Superimposition operation

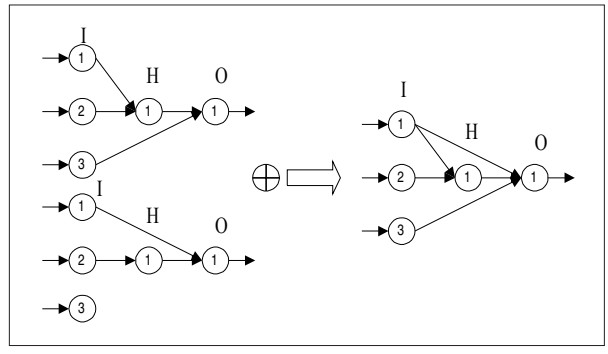


Table 1. Correspondence table for codification of neural architectures

$b_1 \rightarrow 1, 0, \ldots, 0, \ldots, 0$
$b_2 \rightarrow 0, 1, \ldots, 0, \ldots, 0$
…
$b_i \rightarrow 0, 0, \ldots, 1, \ldots, 0$; 1 is set in the i^{th} position.
…
$b_{I\cdot O(H+1)} \rightarrow 0, 0, \ldots, 0, \ldots, 1$

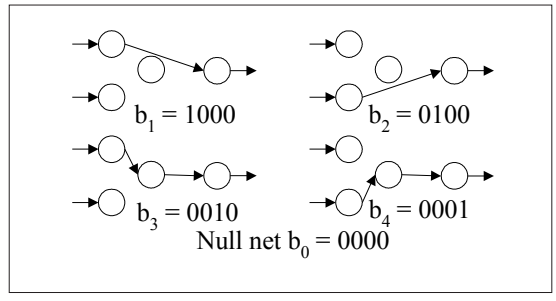
superimposition operation between them, denoted by v⊕v', is the union of the connections from v and v'. Figure 4 illustrates the superimposition operation.

A valid neural network b∈ $V_{I,H,O}$ is also called basic, and so, b∈ $B_{I,H,O}$, if and only if b has only one direct connection from the input to the output layer; or there is one connection from the input to one hidden unit and another from this unit to the output neuron. The subset $B_{I,H,O} \subseteq V_{I,H,O}$ of all basic neural networks has important features because any valid generalised feedforward neural architecture can be built from these basic structures.

We now calculate the cardinal of set $B_{I,H,O}$. There is a total of I·O artificial neural networks with no hidden units, plus I·H·O nets where one output is connected to one input through one hidden neuron. Thus, the cardinal of $B_{I,H,O}$ is I·O·(H+1).

The binary codification of neural architectures using this approach is as follows: There are I·O(H+1) basic neural networks that can be superimposed to build more complex structures. Thus, the set of basic neural networks $\{b_1,\ldots,b_{I\cdot O(H+1)}\}$ can be combined in $2^{I\cdot O(H+1)}$ different ways to build valid neural networks. So, there exists a one-to-one correspondence between the set of all possible superimpositions of all valid neural architectures and the set $V^b_{I,H,O}$ of all binary strings that encode each of them. With all these premises, the codification of all the points in $V_{I,H,O}$ will be based on the codification of the set of basic neural architectures $B_{I,H,O}=\{b_1, b_2, \ldots, b_i, \ldots, b_{I\cdot O(H+1)}\}$ with binary strings of length I·O(H+1) bits as shown in Table 1.

Figure 5. Binary codification chosen for $V_{2,1,1}$



b_1, ..., $b_{I\cdot O(H+1)}$ can be ordered in any way, but should then preserve the same sequence as has been chosen. The null net, a neural network without any connections, is always encoded as a string of $I\cdot O(H+1)$ zeros. Figure 5 shows an example of this table of correspondences between basic neural networks and their codification for the set $V_{2,1,1}$.

Once all basic neural networks have been encoded, any valid neural network included in $V_{I,H,O}$ can be built by applying the binary-OR operator ($\vee$) to the encoded basic nets. This outputs all binary strings included in the set $V^b_{I,H,O}$. It is easy to encode any neural architecture $v \in V_{I,H,O}$ by finding the basic neural networks that have to be superimposed to obtain v and, starting from a string of $I\cdot O(H+1)$ zeros, switching the i^{th} bit to 1 if the i^{th} basic neural net is needed to superimpose to give v. When this codification is used, the set $V^b_{I,H,O}$ has two important features. First, the search space defined with the proposed codification yields only possible solutions to the problem, that is, there are no illegal neural networks. Second, there exist several binary strings that codify the same valid ANN because the set $V^b_{I,H,O}$ encodes the set of all decompositions of all valid neural networks and any valid neural architecture has at least one decomposition. This is a very positive feature, as genetic algorithms will find the best artificial neural network that solves a problem faster, because, generally, several codifications of the best neural architecture are spread in the search space.

Binary Crossover Operators

The crossover operator can be introduced to renew the population. The chosen codification plays a very important role in this operator. It now works with codes of individuals so that each point in the search space is transformed into a vector or string of length *l,* which, in the case of the codification of neural architectures, contains zeros and ones, that is, works with a finite alphabet of cardinal 2. Crossover acts according to a given probability, outputting new individuals, called offspring, with some characteristics "inherited" from the original individuals. n parents are usually used to generate new offspring of two individuals. Some of the most important crossover operators are described in the following.

The *one-point crossover* operator (Larrañaga, 1996) randomly takes two parents from the actual population and cuts their strings at a point, called crossover place, picked at random from all the possible 1s. This determines two substrings in each of the strings of the pair that will be used to create the two new strings that form the offspring. The first offspring is formed by concatenating the first substring of one of the parts with the second of the other parent, and the other two substrings lead to the second offspring.

The *two-point crossover* operator has been found from research into the behaviour of the multiple-point-based crossover operator to be an improvement upon the one-point operator. However, adding more crossover points is of no benefit to algorithm behaviour (Davis, 1991).

The *uniform crossover* (Syswerda, 1991) is based on the definition of a crossover mask composed of a random string of bits of the same length as the individuals. The operator generates two offspring for each pair of parents as follows: The i^{th} position of the string of the first offspring will be the i^{th} gene of the first parent if there is a value 0 at the i^{th} position of the crossover mask and will be the i^{th} gene of the first parent otherwise. The mask is applied inversely for the second offspring. The term uniform crossover stems from the fact that the crossover mask is calculated at random, and there is the same probability of there being a 1 or a 0 in each of the mask positions.

The *generalised crossover* operator starts from two binary strings to output two new strings as the offspring. Let $S = \{0,1\}$ be the set of all possible strings of length l. Let g: $S \rightarrow \{0, 1, ..., 2l-1\}$ be the conversion function that can be used to get its usual decoding as an integer. As g is a bijective function, the equivalent integer set can be defined as the set g(S). Therefore, the search space or the equivalent integer set can be used indistinctly. Given two binary strings a and b Î S, the offspring obtained from the generalised crossover operator will be the randomly chosen points a' and b' such that: $a' \in g\text{-}1([g(a \wedge b), g(a \vee b)]); b' = g\text{-}1(g(a) + g(b) - g(a'))$, where $\wedge$ is the logical operator *and* bit to bit, and $\vee$ is the logical operator *or* bit to bit.

The *Hamming crossover* operator (Carrascal, Manrique, Ríos, & Rossi, 2003) takes into consideration the Hamming distance between a set of progenitor strings in order to generate two offspring. The main feature of this operator is the capability of generating offspring that can be more or less similar to their parents (in terms of Hamming distance), according to a measure of the genetic diversity present in the population, hence dynamically acting on the algorithm's exploitation versus exploration capabilities ratio. The Hamming crossover operator works as follows: Let *l* be the length of the chromosomes and $d_H(s_i,s_j)$ the Hamming distance between two strings s_i and s_j. A set of *n* progenitor chromosomes $G = s_1, \ldots, s_n$ is chosen from the population. The maximum Hamming distance *h* between two progenitors s_{min} and s_{max} is then calculated such as $\forall s_i, s_j \in G$, $d_H(s_i,s_j) \leq d_H(s_{min},s_{max})$. Then, the genetic diversity measure of G is calculated as $g = h/l$. The genetic diversity *g* adaptively guides the behaviour of the Hamming operator, generating two offspring for each set G of progenitor strings. The two offspring will be more or less similar to their parents as a means of decreasing or increasing the diversity in the population of the next generation: A function $\varphi(g)$ is used to determine how many bits $m = \varphi(g)$ of the offspring should be different from their parents'. Function $\varphi(g)$ works as follows: If the values of *g* are close to zero (low diversity), the diversity of the population must be increased in order to make trapping in a local optima less likely, that

is, we need more exploration. On the contrary, if the values of *g* are high, diversity is high, and it could need to be lowered, that is, to concentrate the search (more exploitation).

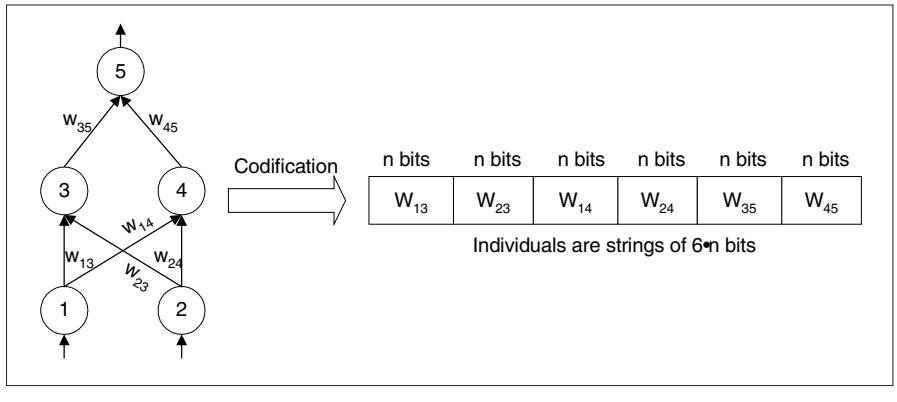
Training Neural Networks with Genetic Algorithms

The usual way to train neural networks using genetic algorithms is to build, starting from a neural network and a set of training patterns, a genetic algorithm that is able to find out the values for the weights of the connections that minimise the mean square error or ensure that it is at least lower than what is considered to be acceptable.

We now describe the construction and generic operation of the genetic algorithm for feedforward neural networks. Given a neural network that is ready to be trained, the initial population of individuals (chromosomes) is generated, each of which codifies a set of values for the weights of the connections and biases of the neural network. Then the fitness of each individual is evaluated, which entails allocating the values of the weights and biases codified by the individual to assess the network connections and calculating the mean square error. The set of training patterns are used to do this calculation. The error value is the individual's fitness. After this initialisation stage, the genetic reproduction, crossover, and mutation operators are usually applied to output new generations until there is convergence toward a population of individuals that encode the set of weights that minimise the network error.

The weights can be encoded in several ways. The most common way is to binary encode the value of each network weight and build the individual by concatenating these binary numbers in a string of bits as shown in the example in Figure 6. In this respect, Gray's codification is very often used to reduce the possibility of small changes in bits leading to big variations in the values that they represent.

Figure 6. Binary codification of the weights of connections of a neural network



Neural network training schemata as shown in Figure 6 and encoding methods as described have been used in several research projects which have achieved acceptable results for problems such as the implementation of the XOR function by means of neural networks, the two-bit adder, or the 4-2-4 coder / decoder problem.

Training artificial neural networks can be seen as an optimisation problem, where the mean square error must be minimised by adjusting the values of the weights of the connections. Evolutionary algorithms should therefore be a suitable option for dealing with this problem. This problem is real-valued in nature as the weights of the connections of a multilayer perceptron are real numbers. Therefore, genetic algorithms using real numbers codification to represent the weights of the neural network can be expected to yield the best results (Michalewicz, 1999). The main problem arising when using this type of codification in genetic algorithms is the issue of crossover operators, as crossover operators based on recombining the genes from the parents cannot be used to generate the offspring. Suppose we have two real-number strings $a = a_1, a_2, ..., a_l$ and $b = b_1, b_2, ..., b_l$ with $a_i, b_i \in \Re$, $\forall i \in \{1, ..., l\}$. Recombining the genes from the parents in any manner would lead to strings of length l of type $c = c_1, c_2, ..., c_l$, where $c_i \in \{a_1, ..., a_l, b_1, ..., b_l\}$, $\forall i \in \{1, ..., l\}$ as offspring. That is, the same genes (real numbers) that appeared in the original population would be used, albeit in different positions, throughout the evolution process. In this setting, the only operator capable of generating new genes would be the mutation operator, which means that, given the low probability of its use generally, the algorithm would not converge.

A recent technique to deal with real-coded genetic algorithms is the *mathematical morphology crossover,* which has recently been developed to work on real-coded genetic algorithms (Barrios et al., 2001). For training artificial neural networks with real-coded genetic algorithms, this crossover operator improves the speed of convergence and is less likely to be trapped in local optima than when using the backpropagation with momentum factor training algorithm (Barrios, Carrascal, Manrique, & Ríos, 2000). Let $D_{\Re}$ be a point in the search space, defined by the string $s=(a_0, a_1, ..., a_{l-1})$, where $a_i \in \Re$. This operator works with each gene in the parents independently to get the respective gene in the two descendants. Let $s_1, ..., s_n$ be an odd number of strings chosen from the actual population to be crossed, the n by l progenitors matrix is defined as:

$$G = \begin{pmatrix} a_{10} & a_{11} & ... & a_{1l-1} \\ a_{20} & a_{21} & ... & a_{2l-1} \\ ... & ... & ... & ... \\ a_{n0} & a_{n1} & ... & a_{nl-1} \end{pmatrix}$$

where $s_i = (a_{i0}, a_{i1}, ..., a_{i,l-1})$, $i = 1, .., n$. The crossover operator works with each column $f_i = (a_{1i}, a_{2i}, ..., a_{ni})$ in matrix G obtaining genes o_i and o_i'. The result of applying the operator to matrix G is, therefore, two new descendants $o = (o_0, o_1, ..., o_{l-1})$ and $o' = (o'_0, o'_1, ..., o'_{l-1})$. The procedure employed by this crossover to generate the new offspring strings o, $o' \in D_{\Re}$ from the parents $s_1, ..., s_n$ in matrix G is as follows:

1. The morphological gradient operator, $g_b(f_i): D_{fi} \rightarrow \Re$, is applied on each vector f_i, i = 0, 1, .., l-1, with a structuring element b:$D_b \rightarrow \Re$ defined as:

   $b(x) = 0, \forall x \in D_b, D_b = \{-E(n/2), ..., 0, ..., E(n/2)\}$.

   E(x) being the integer part of x; g_i is calculated as the value:

   $g_i = g_b(f_i)\,(E(n/2)+1)\;\; i \in \{0, 1, ..., l-1\}$

   Just as the morphology gradient applied to images returns high values when sudden transitions in grey-level values are detected, and low values if the pixels covered by the window (structuring element) are similar, g_i gives a measure of the heterogeneity of gene i in the individuals picked for crossing. If value g_i is high, the population is scattered, while if it is low, the values of that gene are converging.

2. Let $\varphi: \Re \rightarrow \Re$ be a given function. The maximum gene is defined as $g_{imax} = \max(f_i) - \varphi(g_i)$. Likewise, the minimum gene is defined as $g_{imin} = \min(f_i) + \varphi(g_i)$. These values determine the crossover interval $C_i = [g_{imin}, g_{imax}]$, from where the desired value o_i is randomly taken. The i^{th} gene for the other descendant o'_i is calculated from inside the crossover interval using the following formula: $o'_i = g_{imax} + g_{imin} - o_i$.

   The objective is now to get a rule that can dynamically control the range of the crossover interval to make trapping in local minima less likely and to get a high convergence speed. When the individuals to be crossed are diverse (which implies a high value of the gradient), the crossover interval is made narrower according to the values $\max(f_i)$ and $\min(f_i)$, and the interval can be searched for the optimum much faster. On the other hand, if the individuals to be crossed are very similar (gradient close to zero), which means that the population is converging, then it is advisable to expand the interval $[\min(f_i), \max(f_i)]$ to allow exploration of new points in the domain, thus avoiding the possible convergence to a local optimum. This possibility of expanding or narrowing the crossover interval depending on the value of the gradient g_i is given by the function φ, which is shown in Figure 7. It is defined in domain [0,1], so the population has to be normalised in the same range. The application of this function in the described crossover operator is very efficient, as each pair of new individuals can be generated by only one multiplication.

$j(g_i)$ is positive when g_i is strictly greater than 0.2, making the crossover interval narrower in this case. The greatest narrowing takes place at j(1)=0.4. On the other hand, the values of $j(g_i)$ are negative if g_i £0.2, that is, the crossover interval is expanded, and the maximum expansion takes place at j(0.2)= –0.05. As of this point the values of the function are smaller and smaller, as the gradient value falls, until they reach the value –0.01 at g_i=0.

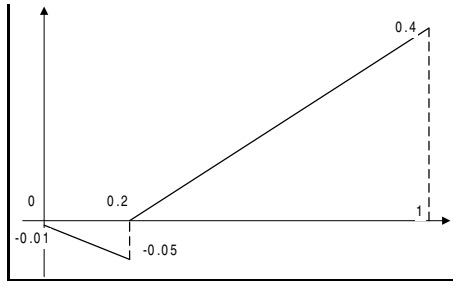
The Evolutionary System for Designing and Training Neural Networks

This section compares the results of a set of experimental tests that have been run with the neural architectures codification methods, binary crossover operators, and real-coded crossover operators explained earlier. The goal of this section is to pick the best operators for building the evolutionary system that can used to construct self-adapting intelligent neural systems. The experimental results are related to the convergence speed and size of the resulting networks. The basic architectures codification method with the Hamming crossover, employed in the neural architectures design module within the evolutionary system, has been compared to the direct and grammar codification methods. The performance of the Hamming crossover operator is compared to one-point, two-point, generalised, and uniform crossovers. The basic architectures method was used in all cases. Finally, the performance of morphological crossover for training artificial neural networks has already been compared to backpropagation with momentum factor, Radcliffe's flat crossover, and blended crossover. The results reported clearly show the superiority of morphological crossover in terms of speed of convergence and lower likelihood of trapping in local optima (Barrios et al., 2000).

Similar genetic settings have been chosen for these experiments. Thus, we used a proportional reproduction strategy in which reproduction probability is decided according to the fitness of each individual. The probability of mutation was set at 0.05, and the crossover operator was used with a probability of 0.6. For each experiment, we ran 100 trials of 1,000 generations and calculated the mean value. The fitness of each neural network in the population is calculated using formula (1). Morphological crossover was employed as the learning algorithm.

The coder/decoder problem was examined to show the speed of convergence and the size of the neural architectures' output as solutions to this problem using the basic architectures codification method as compared with the direct and grammar encoding methods.

Figure 7. Function j used by the mathematical morphology crossover



$$\varphi(g_i) = \begin{cases} -(0.2 \cdot g_i) - 0.01 & \text{if} \quad g_i \le 0.2 \\ (0.5 \cdot g_i) - 0.1 & \text{otherwise} \end{cases}$$

Figure 8 shows the results, in terms of convergence speed, of searching for the best network that solves the encoder/decoder problem, encoding architectures of a maximum of four input and output units and up to 8 hidden neurons (4-8-4). The fitness of the individuals was calculated after it = 50,000 learning iterations had been run and the average MSE of the 10 best individuals of the population was plotted against the generation.

Apart from the fact that the basic architectures codification method clearly outperforms the other methods, it should be noted that, in these experiments, the evolution line of our model tends to converge much faster after several iterations have occurred. Table 2 shows the final solutions output by each method. From this table, we find that the basic architectures codification method not only converges faster, but also gets smaller neural networks. Using this codification approach, 4-0-4 and 4-1-4 architectures with direct connections from the inputs to the outputs are yielded in 89% of the executions. This is computationally very desirable when these networks are working as part of an intelligent system to solve the problem for which they were designed.

The Hamming crossover operator can be used in the binary genetic algorithm (Figure 1) to design neural architectures, increasing the convergence speed and making trapping in local optima less likely with respect to other crossover operators when employing basic architectures codification.

Test results for solving the coder/decoder problem are shown, encoding architectures of a maximum of four input and output units and up to eight hidden neurons. The genetic parameters are similar to the ones used previously. Figure 9 shows the convergence process of the binary genetic algorithm toward the optimum neural network using the Hamming, one-point, two-point, uniform, and generalised crossover operators. For each operator, the average fitness after it=50,000 learning iterations of the 10 best individuals of the population is plotted against the generation.

Figure 8. Convergence process in the coder/decoder problem

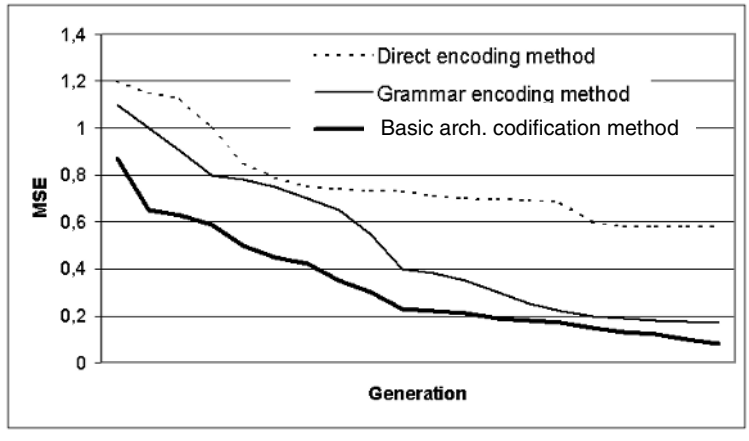


Table 2. Final solutions of the coder/decoder

	Direct encoding	Grammar encoding	Basic Arch. Codification
4-0-4	42%	58%	74%
4-1-4	41%	32%	15%
4-2-4	17%	10%	11%

Table 3 lists the neural networks output to solve the coder/decoder problem by each of the five crossover operators used with the basic architectures codification method. The Hamming crossover clearly outperforms the other operators. Eighty-four percent of the executions run with the Hamming crossover yielded 4-0-4 and 4-1-4 neural networks. The generalised crossover, which is the second-best operator after the Hamming operator, gets the best two architectures in 68% of the cases.

From the results achieved and theoretical considerations previously described, we choose the basic architectures codification method with the Hamming crossover as the binary genetic algorithm for building neural architectures to construct the evolutionary system for designing and training feedforward neural networks. Artificial neural networks are trained to get their fitness using a real-coded genetic algorithm that employs the morphological crossover.

Application: Breast Cancer Diagnosis

The configuration of the evolutionary system chosen in the previous section has been applied to a real-world problem: the diagnosis of suspicious masses present in breast tissue that could be a carcinoma. This application is part of a larger project for automatically detecting and diagnosing breast pathologies. The input for the system built so far is a complete set of views of digitalised mammograms from both patients' breasts, which it uses to search for microcalcifications and suspicious masses, the two main abnormalities that can be detected by mammography. There are two subsystems that work in parallel: the microcalcifications' detection subsystem and the masses' detection subsystem. The output is a set of characteristics for each abnormality found. We focus here on the masses' detection subsystem. The output it gives for each detected mass consists of a vector of 10 real values: first, the radius of the mass (mean of distances from centre to points on the perimeter), the texture (standard deviation of grey-scale values), the perimeter, the area, compactness ($\text{perimeter}^2 / \text{area} - 1$), concavity (severity of concave portions of the contour), symmetry of the mass, standard deviation of the distances taken from the centre of the mass to the perimeter, the biggest distance, and the most concave portion of the perimeter. These features were chosen after having held several interview sessions with expert radiologists working on the project.

The evolutionary system takes a training set of 330 patterns with 10 real-valued inputs (one for each mass characteristic) and one output with two possible values: -1 if the mass

Figure 9. Convergence process in the coder/decoder problem

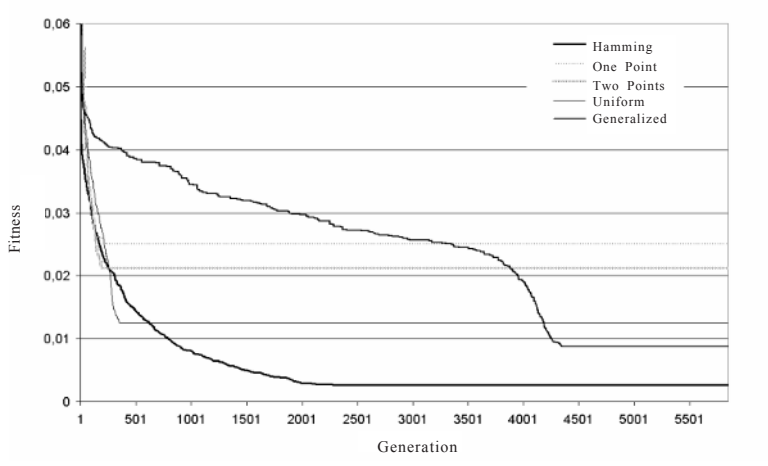


Table 3. Final solutions of the coder /decoder for each of the binary crossover operators

	Hamming	One-point	Two-point	Uniform	Generalised
4-0-4	74%	0%	0%	0%	38%
4-1-4	15%	0%	0%	0%	30%
4-2-4	11%	0%	0%	6%	18%
4-3-4	0%	0%	0%	13%	12%
4-4-4	0%	19%	23%	26%	2%
Others	0%	81%	77%	55%	0%

is considered as benign, and +1 if it is considered as malignant. The intelligent neural system must give the minimum artificial neural network trained to solve this problem. The results given by the evolutionary system in terms of speed of convergence and size of the resulting neural networks are given. The genetic parameters used are the same as in the previous section. Figure 10 shows the progress made by the evolutionary system converging toward the optimum neural network. The average fitness of the 10 best individuals of the population is plotted against the generation. Architectures of a maximum of 10 input units, 1 output and up to 100 hidden neurons have been encoded. This means, in the case of basic architectures codification, that the string length of the individuals is about 1,000 bits.

Table 4 shows the number of connections of the resulting neural networks for the 100 trials run. In the third row of this table, the interval [25-30] represents the resulting neural

networks having between 25 and 30 connections. The high performance and quality of the solutions output by the proposed evolutionary system is very clear from this experiment.

The minimum neural network that solves the problem, which was output in 62% of the trials run, has 24 connections as shown in Table 4. It is important to note that direct connections from the inputs to the output were not needed and that the evolutionary system eliminated four inputs. Therefore, the proposed system was implicitly running a sensitivity analysis of the problem, eliminating all variables that had no or very little importance for the output. The system owes this important feature to the basic architectures codification method.

In this case, the input variables eliminated were: variable 1 (the radius of the mass), variable 3 (the perimeter), variable 7 (the symmetry of the mass), and, finally, variable 8 (the standard deviation of the distances from the centre to the perimeter). These deleted variables bear a clear relationship to each other: The radius and the perimeter represent the size of the mass, while the other two variables represent the shape of the border of the mass. The radiologists working on the project agreed that the size of a mass is not very relevant, although it can sometimes give additional information. The system has not completely eliminated the information related to the size of the mass, because the area, which has not been deleted, is a substitute for the radius and perimeter. A similar thing applies to the other two deleted variables: the symmetry and standard deviation of the radiuses, which can be substituted by concavity.

The resulting neural network was tested using 240 testing patterns that were not used to train the network, yielding an error rate of 3.3%. Of the 240 testing patterns, 180 were malignant masses and the other 60 were benign. The network incorrectly classified only one of the 180 malignant cases, while the network made seven mistakes regarding benign cases. For radiologists the probability of incorrectly classifying malignant cases should be lower than for benign cases. This is because it is much more dangerous to tell a patient that the mass found in her breast tissue is benign, when it is really a cancer than the other way round.

Future Trends

The evolutionary techniques presented in this chapter could be adapted to most neural architectures and topologies today, making it possible to choose the neural model that best fits the type of problem to be solved. Recently, research work within connexionist artificial intelligence is focusing on a specific type of neural architectures: radial basis function networks, which are a type of network that are very useful for pattern classification problems (Yampolskiy, Novikov, & Reznik, 2003). This is because, unlike the multilayer perceptrons, the output of a radial basis function network is mainly influenced by the hidden layer neuron whose centre is closer to the input pattern. Therefore, these neural networks are local approximators, whereas multilayer perceptrons are global approximators. Despite the advantages of radial basis function networks, the design of

Figure 10. Convergence process for the evolutionary system for cancer diagnosis

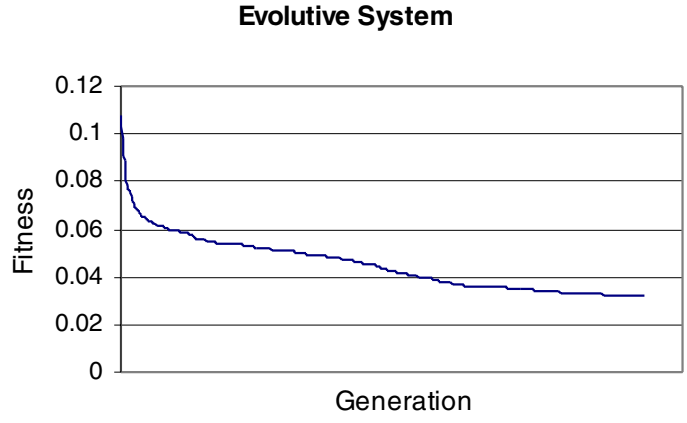


Table 4. Number of connections of the neural networks solution

Number of connections	Evolutionary System
24	62%
[25 – 30]	31%
[31 – 40]	7%
More than 40	0%

an optimal architecture to solve a particular problem is, again, far from being a straightforward matter. Additionally, radial basis function networks trained according to conventional gradient descent-based methods have been shown to be more likely to get trapped in local optima, and they are, therefore, less accurate than when applied to multilayer perceptrons. This is because a set of weights for connections that best solve the problem needs to be found, and the centres of the radial basis functions of the hidden layer neurons also have to be searched.

To surmount the problem of trapping in local optima and improve the accuracy of the results of applying gradient backpropagation to RBFN, research has been conducted aimed at building hybrid training algorithms (Cohen & Intrator, 2002). These algorithms combine an unsupervised search of the hidden layer neuron centres using clustering algorithms, such as k-means, or improved versions, like moving k-means, with gradient backpropagation to get the weights of the connections (Mashor, 2000). However, these approaches are still beset by the very same weakness of local optima trapping, because they use derivatives-based optimisation methods. Research work is now showing that the application of real-coded genetic algorithms with the morphological crossover outperforms backpropagation and hybrid training algorithms for radial basis function networks. Likewise, a new and promising line of research is opening in which the basic

architectures codification method and the Hamming crossover are being adapted to automatically construct intelligent systems using this kind of neural networks, taking advantage of their benefits from the viewpoint of convergence speed and low probability of trapping in local optima. Longer-term efforts are looking at developing an evolutionary system capable of selecting the best suited neural architecture for a given problem and constructing the intelligent system that solves this problem.

Conclusions

This chapter has presented several evolutionary techniques for constructing self-adapting intelligent systems with feedforward neural networks, nowadays one of the most commonly used models. From the results reported after running several tests, it can be stated that the better results are yielded by combining the basic codification method with the Hamming crossover for the binary genetic algorithm and the morphological crossover for the real-coded genetic algorithm. The basic codification method does not codify illegal neural structures, so it requires a shorter codification length because the search space is smaller. Another important feature of this codification is that a one-bit variation of the binary string results in a very similar neural architecture. This feature is enhanced by employing the Hamming crossover, which generates offspring that are near to progenitors in terms of the Hamming distance definition. The morphological crossover is able to adequately balance the exploration and exploitation capabilities of the real-coded genetic algorithm, allowing high-speed searching and avoiding trapping in local optima when adjusting the weights of the connections of the neural network to be trained.

The application of the evolutionary system within a complete application to solve the real-world problem of detecting and diagnosing breast pathologies shows its capability of removing all the connections that are not needed for the output. So, the evolutionary system gets smaller neural architectures, which offer a higher generalisation and lower run-time response speed.

References

Barrios, D., Carrascal, A., Manrique, D., & Ríos, J. (2000). Neural network training using real-coded genetic algorithms. *Proceedings of V Ibero American Symposium on Pattern Recognition,* Lisboa, Portugal (pp. 337-346).

Barrios, D., Carrascal, A., Manrique, D., & Ríos, J. (2003). Cooperative binary-real coded genetic algorithms for generating and adapting artificial neural networks. *Neural Computing and Applications, 12*(2), 49-60.

Barrios, D., Manrique, D., Plaza, M.R., & Ríos, J.(2001). An algebraic model for generating and adapting neural networks by means of optimization methods. *Annals of Mathematics and Artificial Intelligence, 33,* 93-111.

Bornholdt, S., & Graudenz, D. (1992). General asymmetric neural networks and structure design by genetic algorithms. *Neural Networks, 5,* 327-334.

Brown, A.D., & Card, H.C. (2000). Cooperative coevolution of neural representations. *International Journal of Neural Systems, 10*(4), 311-320.

Carrascal, A., Manrique, D., Ríos, J., & Rossi, C. (2003). Evolutionary local search of fuzzy rules through a novel neuro-fuzzy encoding method. *Evolutionary Computation, 11*(4), 439-461.

Cohen S., & Intrator, N. (2002). A hybrid projection-based and radial basis function architecture: Initial values and global optimization. *Pattern Analysis and Applications, 5*(2), 113-120.

D'alotto, L.A., & Giardina, C.R. (1998). *A unified signal algebra approach to two-dimensional parallel digital signal processing.* New York: Marcel Dekker.

Dai, H., & McBeth, C.(1997). Effects of learning parameters on learning procedure and performance of a BPNN. *Neural Networks, 10*(8), 1505-1521.

Eshelman, L.J.,& Schaffer, J. D. (1993). Real-coded genetic algorithms and interval-schemata. *Foundations of Genetic Algorithms, 2,* 187-202.

Hussain, T.S., & Browse, R.A. (1998). Including control architecture in attribute grammar specifications of feedforward neural networks. *Proceedings of the 1998 Joint Conference on Information Sciences: 2nd International Workshop on Frontiers in Evolutionary Algorithms* (vol. 2, pp. 432-436).

Hussain, R.A. (2000). Evolving neural networks using attribute grammars. *IEEE Symposium on Combinations of Evolutionary Computation and Neural Networks,* San Antonio, TX (pp. 37-42).

Kitano, H. (1994). Neurogenetic learning: An integrated meted of designing and training neural networks using genetic algorithms. *Physica D., 75,* 225-238.

Konar, A. (2000). *Artificial intelligence and soft computing*. Boca Raton, FL: CRC Press.

Manrique, D. (2001). *Neural networks design and new optimization methods by genetic algorithms.* Ph.D. thesis, Universidad Politécnica de Madrid, Spain.

Mashor, M.Y.(2000). Hybrid training algorithms for RBF network. *International Journal of the Computer, 8*(2), 50-65.

Michalewicz, Z. (1999). *Genetic algorithms + data structures = evolution programs.* New York: Springer-Verlag.

Ono, I., & Kobayashi, S. (1997). A real-coded genetic algorithm for function optimization using unimodal normal distribution crossover. *Proceedings of the 7th International Conference on Genetic Algorithms,* East Lansing, MI (pp. 246-253).

Principe, J.C., Euliano, N.R., & Lefebvre, W.C. (2000). *Neural and adaptive systems, fundamentals through simulations.* New York: Wiley & Sons.

Radcliffe, N.J. (1990). *Genetic neural networks on MIMD computers.* Ph.D. thesis, University of Edinburgh, UK.

Ríos, J., Barrios, D., Carrascal, A., & Manrique, D. (2001). ADANNET: Automatic design of artificial neural networks by evolutionary techniques. *Proceedings of the 21st*

SGES International Conference on Knowledge Based Systems and Applied Artificial Intelligence, Brighton, UK (pp. 67-80).

Siddiqi, A., & Lucas, S. (1998). A comparison of matrix rewriting versus direct encoding for evolving neural networks. *Proceedings of the 1998 IEEE International Conference on Evolutionary Computation,* Piscataway, NJ (pp. 392-397).

Syswerda, G. (1991). Schedule optimization using genetic algorithms. *Handbook of genetic algorithms*. New York: Van Nostrand Reinhold.

Yampolskiy, R., Novikov, D., & Reznik, L. (2003). *Experimental study of the neural networks for character recognition.* Rochester, NY: Department of Computer Science, Rochester Institute of Technology.

Chapter VI

Using Genetic Programming to Extract Knowledge from Artificial Neural Networks

Daniel Rivero, University of A Coruña, Spain

Miguel Varela, University of A Coruña, Spain

Javier Pereira, University of A Coruña, Spain

Abstract

A technique is described in this chapter that makes it possible to extract the knowledge held by previously trained artificial neural networks. This makes it possible for them to be used in a number of areas (such as medicine) where it is necessary to know how they work, as well as having a network that functions. This chapter explains how to carry out this process to extract knowledge, defined as rules. Special emphasis is placed on extracting knowledge from recurrent neural networks, in particular when applied in predicting time series.

Introduction

Through the use of artificial neural networks (ANN), satisfactory results have been obtained in the most diverse fields of application, including the classification of examples, the identification of images, and processing of natural language.

However, in some fields there is still some reticence toward their use, mainly as a result of one single issue: the fact that they do not justify their answer. An ANN extracts information from a training series (formed by input-output pairs, or simply input if unsupervised training is used). Based on the information obtained, it will be able to offer output for input not previously seen during the learning process. Even if the answer provided is correct, the ANN does not provide any information about why one particular solution has been chosen: it behaves like a "black box". This is unacceptable in certain fields. For example, any system used in medical diagnosis must not only reach the correct conclusions, but also be able to justify on what it has based its decision. For this reason, expert systems are commonly used in medicine.

Expert systems (ES) are able to explain the solution or response achieved, which is their main core and also their guarantee of success. Therefore, this chapter attempts to develop a system that carries out an automatic extraction of rules from previously trained ANN, thereby obtaining the knowledge that an ANN obtains from the problem it solves.

Different rule-extraction techniques using ANN have been used to date, always applied to multi-layer ANN, as they are easier to handle. These networks also have a limited capacity with regard to the knowledge that can be distributed among their connections.

As may be inferred, the extraction of rules and expressions from recurrent ANN is more complicated, due to the fact that past states intervene in neural activation, and that their distributed knowledge capacity is considerably higher than that of multi-layer ANN, as there are no restrictions to neural connectivity. Also, if recurrent ANNs are used in dynamic problems where certain time characteristics such as the prediction of time series intervene, the task of extracting using the methods developed so far becomes harder, if not impossible for most of them.

However, if ANN provides good results, why reject them? It would be enough to find a method that justifies the output offered by ANN based on the input values. This method would have to be able to be applied to networks of any type, meaning that they would have to comply with the following characteristics (Tickle, 1998):

- **Independence of the architecture.** The method for extracting rules should be able to be applied independently from the architecture of the ANN, including recurrent architectures.
- **Independence of the training algorithm.** The extraction of rules cannot depend on the algorithm used for the ANN's learning process.
- **Correction.** Many methods for extracting rules only create approximations of the functioning of the ANN, instead of creating rules as precisely as possible.

- **Eloquence.** The language used to describe the rules extracted must represent the knowledge obtained from the ANN as eloquently as possible.

State of the Art

Genetic Programming

Some believe that Cramer (1985) and Fujiki (1987), who published on evolving programs in 1985 and 1987 at the very first ICGA conference, are the pioneers of genetic programming (GP). However, others think that Friedberg, who from 1958 to 1959 evolved machine language programs (Friedberg, 1958, Friedberg, 1959), is really the pioneer.

John Koza (1992) devised the term used as the title of his book *Genetic Programming*. This book formally establishes the basis of GP used nowadays. Later, the same author published *Genetic Programming I* (Koza, 1994), and, recently, *Genetic Programming III* (Koza, 1999). Both explore the new possibilities of GP.

Different fields are derived from GP. One of the most promising with regard to knowledge discovery (KD) is that of "fuzzy rules" (Fayaad, 1996; Bonarini, 1996). This field derives from the union between fuzzy logic and systems based on rules (SBR). Fuzzy rules can be obtained through evolutionary computation (EC) with the technique known as automatically defined functions (ADF) (Koza, 1994), an evolution of the concept known as "classical genetic programming."

GP works through the evolution of a population. In this population, each individual is a solution to the problem we are trying to solve. This evolution occurs by selecting the best individuals (although the worst also have a small probability of being selected) and combining them to create new solutions. This process is carried out using selection, mutation, and crossover algorithms. After some generations, the population is expected to contain a good enough solution to the problem.

In GP, the codification of solutions is carried out in the shape of "trees." As a result, the user must specify which terminals ("leaves") and functions can be used by the algorithm. With them, complex expressions can be produced that are either mathematical (i.e., involving arithmetical operators), logical (involving Boolean or relational operators), or even more complex.

ANN Rule Extraction

The extraction of rules representing the knowledge stored in an ANN is an NP-complete problem. However, this has not been an obstacle to the development of a large number of systems that carry out this task, more or less successfully (Rabuñal, 2004).

Andrews (1995, 1996) identifies three techniques for discovering rules: "decompositional," "pedagogical," and "eclectic." The first refers to discovery at neural level, focusing on dealing with each of the ANN's neurons, particularly those in the hidden and output layers. Therefore, rules are extracted from each neuron and its relation to the other neurons. This makes the methods related to this technique totally dependent on the architecture of the ANN to be dealt with, meaning their applicability is generically limited. The second treats the ANN as a "black box," where by applying inputs, only the relations between these inputs and the ANN outputs are analysed. The main goal of this technique is to obtain the function that is computed by the ANN. The third technique uses the ANN's architecture and the input-output pairs as a complement to a symbolic training algorithm. This technique combines elements from the first two.

"Decompositional" and "pedagogical" rule-extraction techniques may be combined. An example is DEDEC (Tickle, 1996), which uses trained ANN to create examples from which the underlying rules may be extracted. However, an important difference is that it extracts additional information from the trained ANN by subjecting the resultant weight vectors to further analysis (a "partial" decompositional approach). This information is then used to direct the strategy for generating a (minimal) set of examples for the learning phase. It also uses an efficient algorithm for the rule extraction phase.

Rabuñal (2004) contains an introduction to the most important techniques for extracting rules from neuronal networks. Rabuñal (2004) discusses methods such as the following: those described in Jang (1992), Buckley (1993), and Benítez (1997), which use diffuse rules; Towell (1994) who analysed the network's connections; using lineal programming as in Thrun (1995); GA as in Keedwell (2000); other algorithms such as RULENEG (Pop, 1994) and TREPAN (Craven, 1996a, 1996b), and others based on them (Chalup, 1998; Visser, 1996).

One of the most remarkable aspects in every discovery process is the optimization of the rules obtained from the analysis of the ANN. It should be noted that the rules discovered may have redundant conditions; many specific rules may be contained in general ones, and many of the logical expressions that have been discovered may be simplified if they are written another way. Therefore, rule optimization consists of the simplification and elaboration of symbolic operations with the rules. Depending on the discovery method and the type of rules that have been obtained, various optimization techniques may be applied. Therefore, they may be classified into two main groups: *a posteriori* methods and implied optimization methods. The first group usually consists of a syntactic analysis algorithm which is applied to the rules discovered in order to simplify them. For instance, Duch (2001) uses Prolog as programming language in order to carry out a post-processing of the rules obtained. In this way, optimal linguistic variables are achieved, which help to discover simplified rules which use these variables. Implied optimization methods are techniques used in rule-discovery algorithms that intrinsically cause the algorithm to produce better and better rules.

Knowledge Extraction Systems

Development Premises

When faced with the design of a discovery algorithm, the most important task is that of deciding its modus operandi. As previously explained, discovery techniques may be classified into three main groups: "decompositional," "pedagogical," and "eclectic."

In the pedagogical method, the rules are created based on an empirical analysis of the pairs (input, output) presented to the ANN. This is an approximation based on the "black box," as it completely ignores the internal functioning of the ANN. The basic idea consists of seeing the rule extraction process as a learning task in which the concept sought is the function calculated by the ANN, and the input characteristics are the inputs of the ANN. Techniques of this kind will attempt to extract rules that directly transform these inputs into the outputs.

On the contrary, in the decompositional method, the rules are determined by inspecting the "weight" of each of the ANN neurones and the relationship between one neurone and the rest. This technique has several inconveniences. First, by independently analysing the nodes, the results wholly depend on the architecture of the ANN analysed. Second, the complexity of the rule extraction grows exponentially as the size of the ANN increases, meaning that techniques of this kind are not valid for use with complex ANN.

The third "eclectic" method blends together elements from the first two. In this category, techniques may be classified that use knowledge of the internal architecture or the vectors of ANN weights to complement a learning algorithm.

The combination of CE techniques with traditional inductive algorithms has shown to be capable of offering better results than by applying the previous techniques on an individual basis.

However, the individual application of developmental algorithms reduces the complexity of the method of rule extraction, and produces results equally as good as those that would be obtained in combination with other methods. An example of this is the GABIL system, which carries out an incremental search for a group of classification rules represented by a sequence of fixed-length bits.

The application of CE techniques is to some extent similar to applying an ANN to solving problems. Both are based on the translation of biological processes to computing. They are highly recommendable techniques when working on problems that grow exponentially as the number of variables have to be considered increase. In these situations, which make it unviable to apply deterministic search processes, they behave with tremendous efficiency.

Amongst the CE techniques, opting for the GA as a rule-extracting algorithm presents a series of difficulties. The most important of these is that in order to store the rules extracted in the chromosome, it is necessary to use a GA with variable length individuals, as it is not possible to previously establish the number of rules that will be extracted, which considerably increases the complexity involved. Furthermore, codifying the

problem of rule extraction as an easily understood and interpretable chain of values is a highly complex task.

A large number of these inconveniences are solved with GP. Firstly, the codification of solutions as a tree does not predetermine a fixed size for the solution, or a specific number of rules. Furthermore, the tree is the traditional method of representation used by lexical and syntactical analysers that have always been used for analysing programmes, which may be seen as a generalisation of the rules extracted from an ANN. For these reasons GP was chosen for use as the inductive algorithm for rule extraction.

In this article, GP is used as an algorithm for constructing syntactic trees that represent rules. To do so, a process of symbolic regression has been used, applied to input-output patterns. The input patterns are the input sequences applied to the ANN and the outputs desired are those produced by the ANN. Therefore, the rules returned by GP will imitate the functioning of the ANN. This type of technique may be considered as "pedagogical," in which the ANN is treated as a "black box." The advantage of this is that it is not necessary to know anything about the internal functioning of the ANN. However, it may be desirable to some knowledge of the internal structure and functioning of the ANN with the aim of largely reducing the search area of the rules. Systems of this type, for which there is some information about how they function, are known as "grey boxes." By using GP, it is possible to enter information about the ANN through the configuration of the GP algorithm for the task designated for the ANN. For example, if it is known that the ANN carries out classification tasks, the type of rules wished to be obtained may be limited to Boolean values, avoiding floating point values. This eliminates a large number of mathematical expressions in which to carry out the search, meaning that the system will be more efficient. If a certain level of knowledge of the ANN is used in the system, the technique may be considered as "eclectic" instead of "pedagogical."

System Description

A pedagogical methodology was chosen for use, with the aim of obtaining the greatest independence of the ANN architecture and the learning algorithm used. In this way, the ANN is considered as a system capable of abstracting the peculiarities of a problem, from which it provides outputs to the input patterns used. When extracting rules that explain the functioning of the ANN, it will be of interest that these capture the generality of the resolution process, and not the individual contribution of each of the elements of the network. Therefore, as mentioned previously, the ANN may be treated as if it were a black box or a function — albeit complex — that transforms input patterns into output patterns.

As mentioned previously, the process to be carried out consists of discovering the ANN's rules based on input-output patterns. Inputs are applied to the ANN and, by using the outputs produced, the "inputs_training_set - outputs_ANN" patterns are built. The inductive algorithm (GP) will use these patterns in order to quantify the adjustment achieved by the rules obtained (fitness) with the pair "outputs (ANN) - outputs_obtained (rules)" (Rabuñal, 2004).

Once the ANNs have been designed and trained, the same training and test values may be used in order to generate a second data pattern which will be used to search for the

Figure 1. Rule extraction process of ANN

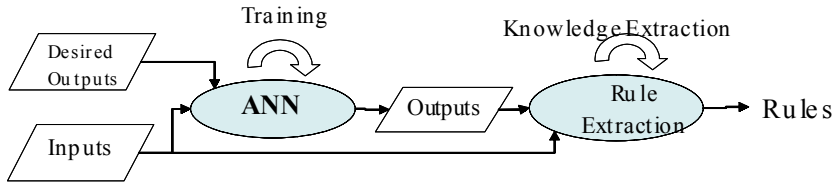
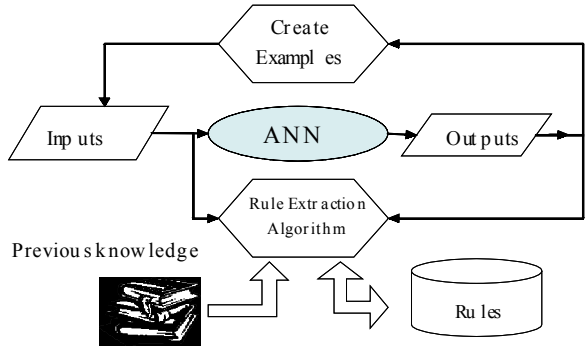


Figure 2. Rule extraction process



rules that the ANN has acquired during the training process. Figure 1 shows a general chart of what the process would be like.

This previous knowledge would include the information about the type of data used by the ANN, or the type of problem trying to be solved. For example, if it is known that the ANN is working exclusively on Boolean values, it may be possible to prevent the extraction algorithm attempting to seek out rules that include continuous operators or variables. In this way, it is possible to reduce the search area that the algorithm has to explore, with the subsequent improvement in efficiency, depending on the quantity of knowledge held about the network and the type of problem being dealt with.

Figure 2 shows the discovery process diagram followed for the development of the rule discovery system.

Discovering the Generalization Capacity

Another aspect to be taken into account when designing the discovery mechanism is that it may be expanded in order to determine an important feature presented by ANNs, which has been one of the main reasons for their success: the generalisation capacity. The purpose is to extract the knowledge which the ANN obtains from the training patterns and from test cases that have not been provided during the learning process.

For this purpose, we must design an algorithm to create new test cases which are going to be applied to the ANN, allowing its behaviour to be analysed when faced with new situations, and also to discover its functional limitations. This aspect is of particular relevance when applying ANNs to certain fields with a high practical risk, such as monitoring nuclear power plants and diagnosing certain medical conditions, and it has not been clearly used in the scientific literature about ANNs and rule discovery until now.

The ideal situation to be able to extract knowledge about the generalisation ability of the ANN would be to analyse the results produced by the ANN when faced with all of the possible combinations of input values. However, this becomes unviable as the search area increases, meaning that it will be necessary to provide a mechanism that makes selections depending from which input-output combinations (the group of examples) the extraction of knowledge will be made. One alternative would involve selecting a subgroup of these combinations randomly and in a uniform distribution. This leads to obtaining a relatively low number of patterns, meaning that the rule extracting process does not require excessive computing power. However, it does have the inconvenience that it is not possible to guarantee that all of the combinations have been selected, making it possible to extract all of the generalisation capacity of the ANN.

As regards discovering the generalisation capacity, an algorithm known as the "algorithm for the creation of new patterns" has been designed. New test cases may be created using this algorithm, making it possible to discover rules and expressions of those parts of the ANN's internal functioning which have not been explored by the learning patterns. We are not sure if the behaviour of the ANN matches the behaviour of the external reality; we only hope so. The use of new test cases has the sole purpose to better mimic the model of the ANN; it cannot necessarily be extended to the point that the rule set will better mimic the problem being considered. For this purpose, we start with the learning patterns, applying to them the rule-discovery algorithm that has been previously used through GP. The solution used is based on the premise that all ANNs obtain the capacity for generalisation from the implicit knowledge of the training files. This means that the example-creating algorithm will have to make use of these files. Additionally, it will be provided with a new set of input combinations, selected at random and based on the inputs of the previous files.

Having determined the existence of this group of new inputs, its size will have to be established. If it is small, its influence will barely be noticeable in the final result. On the contrary, if it is too high, it will make the solutions tend toward the information provided by these new inputs, minimizing the contribution of the original training patterns. Once a good adjustment is obtained, the algorithm for creating new patterns is applied, thus generating a series of new input patterns which are applied to the ANN, obtaining the relevant outputs. These new patterns "new_inputs - outputs_ANN" are added to the training patterns, renewing the rule-discovery process. It should be noted that in order to continue with the discovery process, the first step is to reassess each and every individual in the population in order to calculate its new adjustment. Once this has been done, the discovery process may continue.

As the rule extraction process advances, it is possible to eliminate the examples that the rules obtained so far have captured from the group of new inputs. When there is a new test case in which the ANN's output coincides with that produced by the best combi-

nation of rules obtained, then that test case is eliminated, as it is considered that the rules obtained have acquired the knowledge represented by that test case. It is possible to add new examples at the same time, making it possible to explore new regions of the search space. The examples provided by the training files will stay constant throughout the process, meaning they will always be presented to the rule extraction algorithm.

Therefore, the process is carried out in an iterative manner: A number of generations or simulations of the rule-discovery algorithm are simulated, new test cases are generated once it is finished, and those whose result coincides with that of the ANN are eliminated. Once all the relevant test cases have been eliminated, new cases are created in order to fill in the gaps left by those that have been deleted. Each new case is compared with the existing ones in order to determine if it is already present in the test cases. In this case, it is eliminated and a new one created in the same way. Then the process starts again, running the discovery algorithm N generations, once again repeating the elimination and creation of test cases. This process will be complete when the user determines that the rules obtained are those desired.

The parameters used by the algorithm for the creation of new patterns are described in detail in Rabuñal (2004), together with a more formal description of the algorithm.

In order to generate a new input pattern from those already existing, a number of different alternatives may be applied: creating all of the new values randomly, or, starting out with some of the original patterns, modifying them in some way in order to generate a new entry. Using the first method, it is possible to generate patterns that vary greatly from each other, and which do not correspond to the areas of the search space in which the ANN work, which may reduce the validity of the rules obtained. However, this may be interesting in the event of wishing to check how the ANN responds in extreme situations. In the case of the second alternative, one of the training patterns is chosen at random, and by slightly modifying its values, a new example case is constructed. In this way, inputs are generated that are not too far apart in the search space of the training cases. Therefore, the tendency will be to produce new input sequences from which all of the possible output cases are generated for which the ANN has been trained.

Although the algorithm for generating examples is totally independent from the architecture of the ANN, the fact of working with a feedforward or recurrent network means that the algorithm is totally different.

When working with recurrent networks, it is not enough to compare the output offered by the RANN with that offered by the rules for an input pattern to determine if the rules have captured the correct knowledge or not. This is due to the fact that the functioning of an RANN is not based on individual input patterns, but instead on sequences of inputs. This means that the behaviour of the RANN can only be said to have been captured when the rules offer the same outputs as the RANN itself for all of the values of a sequence of entries.

The solution adopted in the particular case of the RANN uses the training files as its initial base, in order to generate a new starting sequence which, when applied as input to the ANN, will produce the subsequent input-output combinations. For example, if an RANN is used for predicting a short-term time series, it will work as a feedback system: The value is given at moment t and the network predicts the value at the moment $t+1$, and this value at moment $t+1$ is provided as input to predict the value at moment $t+2$, and so on.

A sequence of inputs will be eliminated when the rules simulate the behaviour of all of the values of the series with an error below a pre-established threshold — when the sequence of output values for the ANN obtained from the new value created coincides with the sequence obtained applying the rules extracted about this value. When this occurs, a new input value must be created, from which a new sequence of inputs will be generated so that the rule extraction process can start again.

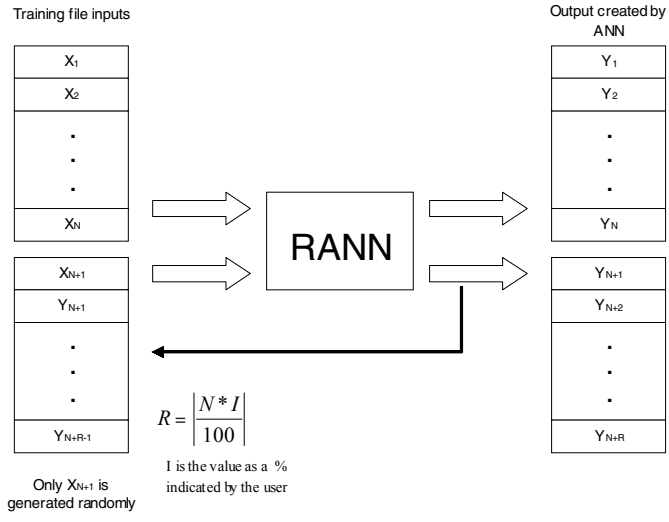
In the case of working with RANN, the system will have a series of additional parameters. In total, it will have the following parameters:

- **Percentage of new patterns**. How many new patterns will be used apart from the training patterns. In the case of not working with time series, these new patterns are checked so that they are different from the training patterns. If working with a time series, these new random patterns are added at the end of the series used for training.
- **Probability of change**. The process for creating new test cases focuses on generating new sequences from the training data. A sequence of inputs is chosen at random from the training file. For each input, the same value is maintained or a new one is generated at random, following a certain probability. This technique is similar to the EC mutation process.
- **Minimum error for survival**. Once the process has started to extract rules, and as these are generated, they reflect the patterns with which they are being trained. Once the rules extracted sufficiently reflect the behaviour of the ANN with a given pattern, this pattern is no longer necessary, and so it is discarded and a new one created. This parameter indicates when a pattern should be discarded, when the error value obtained for the rules in this specific pattern is below a certain threshold.
- **Number of feedback outputs**. When working with recurrent networks, the aim is also to study the behaviour of the network by taking the outputs it returns and using them as inputs. This parameter indicates the number of patterns (the group of patterns whose size is indicated by the parameter "percentage of new patterns") that will not be generated at random, but which instead will be the result of simulating the network, taking each input as the output in the previous moment.

Figure 3 is a graphic representation of the idea. The way of eliminating the sequence of examples also will vary. In this case, only a random value is created, and this is what creates all of the following cases. The criteria for elimination will be when the *R* new sequences created have a comparison error with those produced by the RANN of zero, or when the rules assimilate the knowledge that exists in all of the subseries of new cases. As a result, all of them will be eliminated. The next step will be to create a new random value, use it to construct the new subseries, and start the extraction process again until the adjustment is achieved, and so on.

As may be seen in Figure 3, an overlapping occurs in the sequence of inputs created between the inputs (X) and the outputs, as a result of the first randomly generated value. In this way, it is possible to analyse the behaviour of the RANN faced with a multitude of possible future sequences, and to abstract the generalisation knowledge resident in its recurrent connections.

Figure 3. Initial process for crating examples in cases of RANN and Temporary ANN



Here, it is important to note that in order to obtain an output, the RANN not only takes into account the input value at the actual moment, but that the outputs from previous moments also have an influence. This means it makes use of the ability for *internal memory* provided by the feedback, and therefore that every effort should be made to ensure that this information stored in the ANN does not affect the outputs associated with the new patterns created by feedback. To do so, before generating a new example sequence, a series of randomly generated values are entered into the ANN so that the internal memory is not determined by previous examples.

Rule Extraction with RANNs for Time Series Forecasts

When working with temporary RANN to predict series, the data used during the training process are usually values or measurements taken from a real process. For example, in the prediction of sunspots, the quantity of sunspots produced each year are represented, and these values are cannot be altered. In this and other similar cases, the unknowns are the future values and the behaviour of the ANN in order to predict these values.

In the case of short-term prediction, the aim will be for the ANN, or otherwise the rules extracted from its behaviour, to predict the output value at the moment t – Y(t) — using the input values X(t) and the outputs of the previous moments Y(t-1), …, Y(t-n).

The process is slightly different for long-term prediction. In this case, the input values are known up to moment t – X(1), …, X(t) — and the aim is to predict the output value at a future moment, for example at t=N. In this case, the outputs provided by the ANN in a given moment will be re-entered as inputs in the following cycle. This means that the value of Y(N) may be obtained from the values X(1), …, X(t), Y(t), Y(t+1), …, Y(N-1).

Rule extraction will be the same in both cases; what will vary is adapting these rules depending on whether greater importance is to be given to prediction in the short or long term. This adaptation will be determined by comparing the output values provided by the ANN and the rules themselves. The difference lies in how the output for the rules is obtained in the case of the examples created by feedback (patterns X(i+1), …, X(i+n) in Figure 3).

This means that in order to check how the rules adjust the behaviour of the ANN, it is necessary to compare, from a given moment t=i, the output provided by the ANN– $Y\_ANN_{t=1}$ — with that obtained using the rule — $Y\_ Rule_{=i}$. In the case of short-term prediction, in the following cycles the rule will use that provided by the ANN in the previous moment as the input:

$$X\_Rule_{t=i+1} = Y\_ANN_{t=i}$$

Here the aim is to favour rules that best adjust the behaviour of the ANN at a given moment, only using information from the previous outputs of the ANN.

On the contrary, in case of long-term prediction, the rule will use as an entry value at moment i+1 the feedback output from the rule at the moment immediately beforehand. In this way, the output from the moment t+n (long-term prediction) will be a direct consequence of the outputs provided by the rules, not by the ANN, from moment i.

$$X\_Rule_{t=i+1} = Y\_Rule_{t=i}$$

Optimization of Rules Obtained

Whatever the methodology chosen to extract the rules, one of the most important steps involves the optimization or post-processing of the rules obtained. This optimization is necessary for several reasons:

- Multiple specific rules may be summarized in a single, more general rule.
- Redundancies may appear amongst the rules obtained.
- Many rules may be simplified just by rewriting them in a different way.

Also, the optimization process includes the simplification and/or creation of symbolic operations between the rules.

Various optimization techniques may be applied, depending on the extraction method used and the type of rules obtained. These techniques may be divided into two groups: a posteriori methods and implicit optimization methods.

The first group is generally formed by syntactic analysis algorithms. One example is the use of the Prolog language to carry out the simplification and validation of the new rules obtained. To do so, a series of linguistic variables are generated, which are then used to

find the simplified rules that use these variables the implicit optimization methods are formed by a group of techniques used in rule discovery algorithms, which intrinsically lead to the algorithm discovering improved rules.

As mentioned, GP was chosen as the rule extraction algorithm. One of the classic configuration parameters of GP, parsimony or penalisation for profundity, may be used in the process of optimizing the rules extracted.

In conceptual terms, when the level of adaptation of a genetic individual is evaluated — in this case each individual or tree will be the representation of a rule — its "goodness" value is reduced by a certain amount depending on the number of terminal and non-terminal nodes it presents. The larger the number of nodes, the greater the amount by which its goodness value is reduced. This method is used to benefit the creation of simpler rules which are therefore easier to interpret.

GP Parameter Configuration

The next step in discovering rules is the numeric calculation of the parameters involved in the algorithm.

Experimentation is used to adjust the GP-related parameters. These are the implemented parameters:

Table 1. Possible GP parameters and their values

Parameter	Options
Population creation algorithm	Full Partial Ramped Half&Half
Selection algorithm	Tournament Roulette Stochastic remainder Stochastic universal Deterministic sample
Mutation algorithm	Subtree Punctual
Elitist strategy	YES-NO
Crossover rate	0-100 %
Mutation rate	0-100 %
Non-terminal selection probability	0-100 %
Population size	Number of individuals
Parsimony level	Penalty value

When specifying terminal and non-terminal operators, a type must be specified: Each node will have a type, and non-terminal nodes will demand a particular type from their offspring (Montana, 1995). This guarantees that the trees generated in this way satisfy the grammar specified by the user. Also, both sets of specified operators must fulfill two requirements: closure and sufficiency; it must be possible to build correct trees with the specified operators, and to express the solution to the problem (the expression we are searching for) using these operators.

An empirical analysis must be carried out in order to adjust the ideal parameters, trying different combinations in order to adjust the various values progressively until the best results are achieved.

The following parameter adjustment is related to the terminal and non-terminal nodes that will take part in the execution of the algorithm. The knowledge we already possess about the ANN, together with the types of problems for which it was designed, will be of consequence in this case.

These are the parameters to be selected:

Table 2. Possible elements of groups of terminals and functions

Parameter	Options
Logical Operators	AND OR NOT
Relational Operators	>, <, =, <>, >=, <=
Arithmetic Functions	+, -, *, % (protected division)
Decision Functions	IF–THEN– ELSE over real values IF–THEN– ELSE over Boolean values
Constants	Random rank selection or manually established (indicating rank or one by one)
Input Variables	Determining one by one whether it is real or Boolean
Type of Outputs	Real Boolean Turning Real into Boolean (specifying the threshold)

As may be seen, a wide range of configurations is possible. The more information we have about the ANN, the easier it is to establish the values of the parameters detailed above. Otherwise, experimentation is required in order to determine which are the best for each ANN.

Results

Redes Feedforward for Classification

Rabuñal (2004) provides an extensive description of how to apply ANNs in different types of problems, as well as the subsequent extraction of rules from the networks obtained using the technique described in this article. The problems resolved in this case are the classification of iris flowers (Fisher, 1936) and poisonous mushrooms (Blake, 1998) and the diagnosis of breast cancer, hepatitis, and appendicitis (Blake, 1998). The results obtained in solving these problems using genetic programming and ANN, as well as their comparison against other techniques, may be found in Rabuñal (2004).

Having trained the network for each problem, the rule extraction algorithm is applied only using the training patterns as inputs, and using the algorithm for creating new patterns. Table 3 shows a comparison of the adjustments obtained by using only the learning patterns, and those achieved by using the algorithm for creating new patterns. In order to create this table, an analysis file was built with all of the possible values for the input variables, taking regular intervals (normalized inputs). By using these intervals, it is possible to analyse the whole possible range of classifications that the rules carry out. Table 3 shows the adjustment values produced by the rules obtained with both methods when the analysis file patterns are evaluated.

Time Series Forecast

RANN architectures must be used for the prediction of time series and for modeling problems of this type. In this section we will show that the system proposed in this chapter can also successfully perform knowledge extraction from RANNs. The extraction of rules from ANNs with recurrent architecture presents an additional hurdle, as these ANN are characterised by their immense capacity for representation and distributed knowledge among their connections. This can be specifically applied to time and dynamic problems. The problem to be solved will be the prediction of a classic chaotic laboratory time series: the Mackey-Glass series (Mackey, 1977). The following results show that the rules to be obtained from this ANN should incorporate mechanisms for treating time values. Therefore, non-terminal nodes representing mathematical and trigonometrical operations will be used, together with input variables at previous n moments (X_n). Most of these time series cases are structures with a single input and a single output. The input corresponds to a number value at the moment t, while the system's output is the prediction of the number value at t+1. The Mackey-Glass equation is an ordinary differential delay equation (3).

$$\frac{dx}{dt} = \frac{ax(t-\tau)}{1+x^c(t-\tau)} - bx(t)$$

Choosing $\tau = 30$, the equation becomes chaotic, and only short-term predictions are feasible. Integrating the equation (3) in the rank [t, t + τ t] we obtain:

$$x(t+\Delta t) = \frac{2-b\Delta t}{2+b\Delta t}x(t) + \frac{\alpha\Delta t}{2+b\Delta t}\left[\frac{x(t+\Delta t-\tau)}{1+x^c(t+\Delta t-\tau)} + \frac{x(t-\tau)}{1+x^c(t-\tau)}\right]$$

The first step is to obtain an RANN which emulates the behaviour of the time series. The RANN we used has three neurons with tangent hyperbolic activation function with total interconnection. The training files used correspond to the first 200 values of the time series (Figure 5). The RANN resulting from the training process which yielded the least mean square error (MSE=0.000072) is shown in Figure 4.

Once we obtained the RANN, we attempted to obtain the rules and the expressions which direct its functioning by using symbolic regression. In this case, we used a test file containing the first 1,000 values of the time series. These 1,000 values were transferred to the RANN, obtaining the corresponding outputs. Using the input-output file, we ran the GP algorithm.

Different combinations of terminal and function elements and GP parameters were tried, and the following used:

- *Arithmetic functions*: +, -, *, % (protected division)
- *Constants*: 10 random values in [0,1] *Variables*: X_n, X_{n-1}, X_{n-2}, X_{n-3}
- *Selection algorithm*: Tournament *Population size*: 1,000 individuals
- *Crossover rate*: 95% *Mutation rate*: 4%
- *Parsimony level*: 0.0001

Table 3. Adjustment values obtained in the process of extracting rules, using only training patterns and the algorithm for creating new examples

Method	*Problem*						
	Iris Flower			*Breast Cancer*	*Hepatitis*	*Poisonous Mushrooms*	*Appendicitis*
	Iris setosa	*Iris versicolor*	*Iris virginica*				
Without dynamic creation	63.34%	56.97%	57.23%	64.56 %	59.95 %	77.44 %	78.93 %
With dynamic creation	95.45%	88.81%	77.88%	71.91 %	73.34 %	82.51 %	85.97 %

Also, the elitist strategy was used, not permitting the loss of the best individual. The rule expressed as a mathematical function is as follows:

$$((X_n * ((((((X_n * (X_n * X_{n-2})) \% X_{n-2}) * (((0.9834 * (((((X_n * (X_{n-2} \% X_{n-3})) * X_{n-3}) * X_{n-3}) \% X_{n-3}) \% (X_n \% X_{n-3}))) \% X_{n-3}) \% ((X_{n-2} * X_{n-2}) \% X_{n-3}))) \% X_{n-3}) * X_{n-2}) \% X_{n-2})) \% ((X_n * X_{n-2}) \% X_{n-3}) * 0.9834))$$

This function obtains a fitness value (normalised) on the 1,000 values produced by the RANN of 0.0029. Figure 5 compares the values produced by the function forecast (4) and the RANN forecast.

Considering Figure 5, it may be concluded that the rules adjust the short-term prediction of the ANN in a satisfactory manner. But what happens in the case of long-term prediction? To verify this, it is necessary to introduce the feedback of the outputs obtained in the immediately previous moment as input values to the rule.

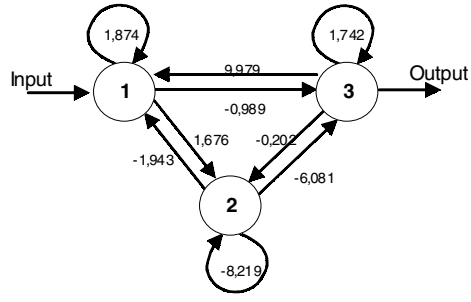
The results show how the behaviour of the rule clearly differs from that of the ANN. While the ANN normally tends toward the middle values of the series, the outputs of the rules will tend toward one value or another, depending on the previous tendency — meaning if the series is on an ascending or descending slope at the moment when the feedback of the outputs begins. If the slope is descending (see Figure 6) the values provided by the rules tend toward 0 or $-\infty$, whereas if the slope is ascending (see Figure 7) the values tend toward 1 or $+\infty$.

Here it is possible to see how the output provided by the rules starts to grow in an undefined manner (specifically, at moment 1,000 it has a value of $1.94*10^{10}$).

Therefore, it seems apparent that the rule extracted has not apprehended the generalization capacity of the RANN as successfully as required. In order to solve this problem, the rule extracting process must be carried out again, this time with the improvement referred to. In this case the suitability of a rule will be obtained based on the feedback of its outputs, and not on the real values of the series itself or the outputs provided by the RANN. The scheme used to configure the rule extraction process includes the first 200 values from the series, 20 random values (to wipe the internal memory of the RANN) and 60 feedback entries, using the following parameter configuration:

- *Percentage of new patterns:* 40% (Total of 200 + 80 patterns)
- *Probability of change:* 20%
- *Minimum error for survival:* 0.01
- *Nº of feedback outputs:* between 10 and 60

Figure 4. RANN that emulate the Mackey-Glass function



At first, 10 values were used for this last parameter, and while the algorithm was being executed it was modified until it ended with 60 values. The reason for doing it this way is that if the process starts with a high value, the behaviour of the RANN tends toward the average of the last former values, and as a result the extraction process becomes stuck in a local minimum situated in the rule: Output = X_n (the output is the entry value $\rightarrow X_{n+1} = X_n$). Therefore, in order to prevent this from happening, the random creation of new values is favoured over feedback outputs, and once the local minimum is exceeded, feedback outputs are favoured. This must be done manually, as the algorithm does not know if it has fallen in a local minimum or not, and by observing the type of rules obtained, the online parameters are readjusted.

In the case of the parameters for the extraction algorithm, the same were used as in the previous case, as the type of problem is the same, and also the expressions obtained and the results they produce may be compared more significantly in the same case scenarios.

Figure 8 shows an example for the creation of the initial values obtained by the algorithm for the dynamic creation of examples.

After one day and four hours of execution the following equation was obtained as a result of the extraction process:

$$((((((X_{n-1} - X_{n-3})*(-((X_n * X_n) - 0.7554)))*(-2.3241))*(-0.2474))+X_n) - ((((X_{n-1}*(-((X_n * X_n) - 0.7554)))*(-2.3241))*(-0.2474))*((((X_n - (-0.2457)) - X_{n-2})*(X_{n-3} + ((X_n * X_n)-0.7554))) \% ((X_{n-3} - (0.2979 * X_{n-4})) \% 0.0752))))$$

Figure 9 shows the adjustment obtained from the previous equation with the learning patterns configured as 200 training values, 20 random values, and 60 feedback inputs. It shows in greater detail the behaviour of the RANN when faced with the feedback of its outputs.

Figure 5. Comparison between the RANN forecast and the function forecast (4)

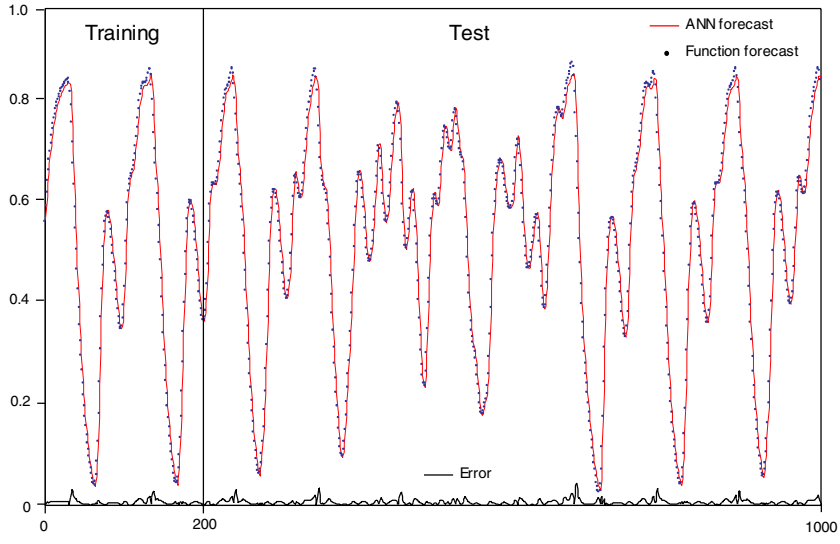


Figure 6. Mackey-Glass series: Adjustment obtained with 200 feedbacks

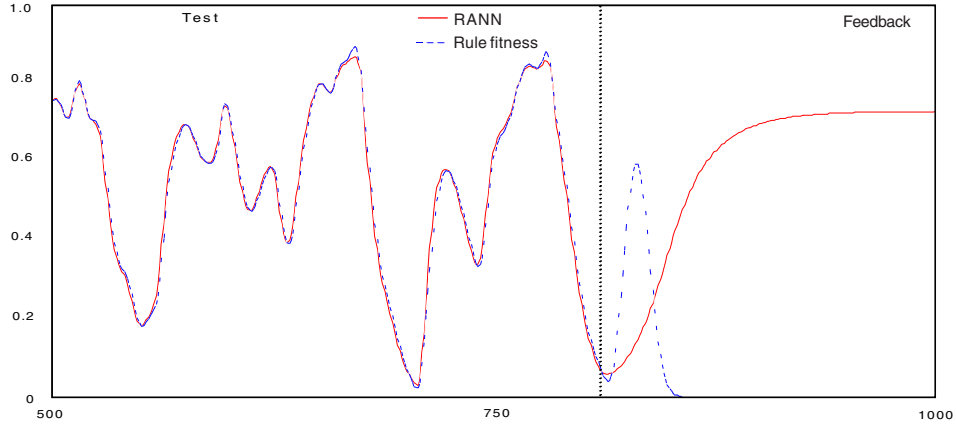


Figure 7. Mackey-Glass series: Adjustment obtained with 150 feedbacks

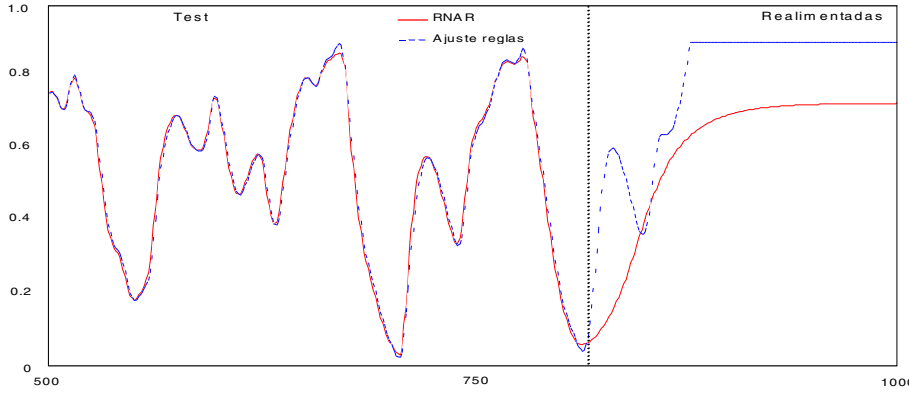


Figure 8. Learning pattern created by the example creation algorithm

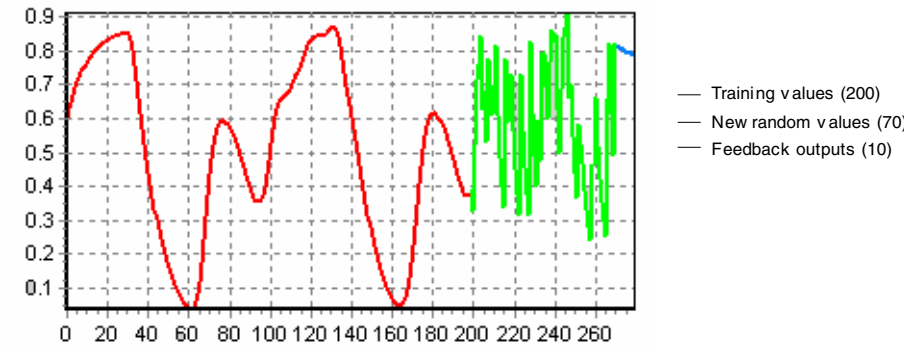


Figure 9. Adjustment obtained

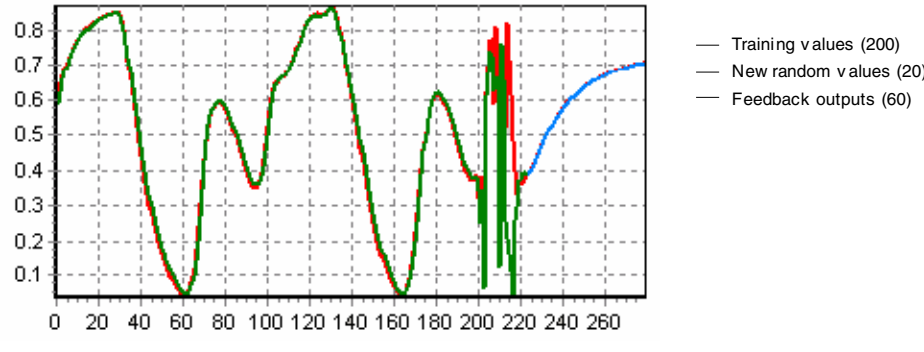
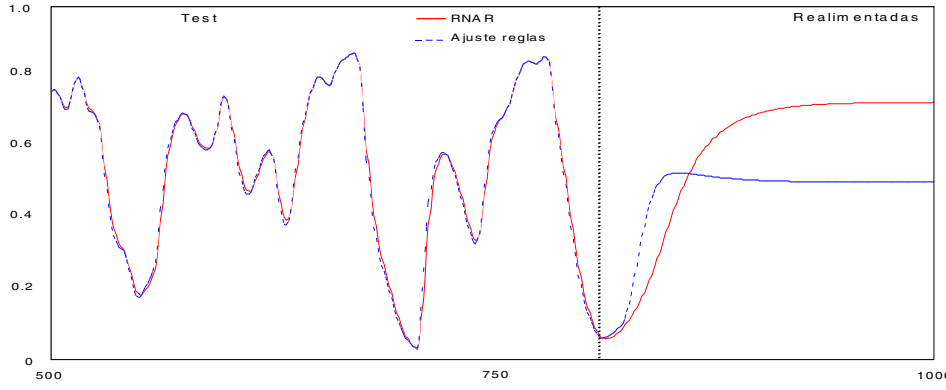


Figure 10. Mackey-Glass series: Adjustment obtained with extraction of generalisation



In Figure 9, it is possible to start to see how this rule behaves for long-term prediction. Here we may see how the outputs of the expression are profiled within the range [0.1], the same presented by the original signal. Furthermore, as may be seen in Figure 10, the behaviour of the output provided by the rules is very similar to that provided by the RANN with an initial zone with rising values that are then stabilized toward the midway values of the signal, and with a highly satisfactory prediction in the initial moments in which there is output feedback.

Conclusions

As mentioned in the introduction to this chapter, any system that attempts to extract rules from knowledge stored in an ANN has to comply with a series of features: It must be independent from the architecture of the network, independent from the learning algorithm used, have a suitable level of correction, and generate highly expressive rules.

Thanks to the proposal presented here it is possible to treat the ANN as if it were a "black box," using an EC technique like GP, by only considering the inputs and outputs produced by the ANN. In this way, it is possible to comply with the requirements that the architecture and learning algorithm are independent.

Also, the rules generated approximate the functioning of the ANN in the same way as the adjustment value produced by the extraction algorithm (GP), meaning that correction is guaranteed. Furthermore, these rules provide the semantic value of the trees of expressions with which the genetic individuals of the extraction algorithm are codified.

Therefore, it may be said that extraction algorithms based on EC techniques meet the requirements demanded of algorithms of this type, as well as producing similar, if not

superior results to those provided by methods that are more specific and dependent on the problem.

The results of the tests carried out are presented in support of these ideas. The example chosen (the Mackey Glass series) is frequently included and studied in the literature from different scientific areas (mathematics, statistics, etc.) and is sufficiently complex in order to determine if the extraction algorithms are efficient at carrying out their task. A detailed study of these results reveals how the rules obtained by the algorithm proposed obtain results that are as good as, if not better than, several of the other methods specifically designed for each of the problems proposed.

Moreover, we may conclude from the results obtained from discovering the generalisation capacity, that a rather accurate simulation of the ANN's behaviour has been achieved with regard to the possible combinations of values that may occur in the inputs. Therefore, we obtained a high fidelity of ANN behaviour, seen in the high success rate of rule extraction from the ANNs. We may also state that the knowledge treasured by the ANN was obtained in an explicit and comprehensible way from a human's point of view. It was also possible to express the ANN's generalisation capacity using a symbolic rule.

Future Studies

A future development will be the analysis of the different parameters that intervene in the correct functioning of the algorithm, depending on the type of problem solved by the ANN. The ANN should be treated not as a "black box," but as a "grey box," in which, for instance, the activation function of the ANN is known, being incorporated as one of the mathematical operators of GP, and analysing which rules are extracted by this operator.

In order to accelerate the rule-extraction process, a network may be used with several computers so that the search is conducted in a distributed and concurrent manner, exchanging rules (sub-trees) between each.

References

Andrews, R., Cable, R., Diederich, J., Geva, S., Golea, M., Hayward, R., Ho-Stuart, C., & Tickle A.B. (1996). *An evaluation and comparison of techniques for extracting and refining rules from artificial neural networks.* Queensland University of Technology, Neurocomputing Research Centre. QUT NRC Technical report.

Andrews, R., Diederich, J., & Tickle, A. (1995). A survey and critique of techniques for extracting rules from trained artificial neural networks. *Knowledge Based Systems, 8,* 373-389.

Andrews, R., & Geva, S. (1994). Rule extraction from a constrained error backpropagation MLP. *Proceedings of the Australian Conference on Neural Networks,* Brisbane, Queensland (pp. 9-12).

Benítez, J.M., Castro, J.L., & Requena, I. (1997). Are artificial neural networks black boxes? *IEEE Transactions on Neural Networks,* 8(5), 1156-1164.

Blake, C.L., & Mertz ,C.J. (1998). *UCI repository of machine learning databases.* University of California, Department of Information and Computer Science. Retrieved from www-old.ics.uci.edu/pub/machine-learning-databases

Bonarini, A. (1996). Evolutionary learning of fuzzy rules: Competition and cooperation. In W. Pedrycz (Ed.), *Fuzzy modelling: Paradigms and practice.* Norwell, MA: Kluwer Academic Press.

Browne, C., Düntsch, I., & Gediga, G. (1998). IRIS revisited: A comparison of discriminant and enhanced rough set data analysis. In *Rough sets in knowledge discovery* (vol. 2, pp. 345-368). Heidelberg: Physica Verlag.

Buckley, J.J., Hayashi, Y., & Czogala, E. (1993). On the equivalence of neural nets and fuzzy expert systems. *Fuzzy Sets Systems, 53,* 29-134.

Chalup, S., Hayward, R., & Diedrich, J. (1998). *Rule extraction from artificial neural networks trained on elementary number classification task.* Queensland University of Technology, Neurocomputing Research Centre. QUT NRC Technical report.

Cramer, N.L. (1985). A representation for the adaptive generation of simple sequential programs. *Grefenstette: Proceedings of First International Conference on Genetic Algorithms.*

Craven, M.W. (1996). *Extracting comprehensible models from trained neural networks.* Ph.D. thesis, University of Wisconsin, Madison.

Craven, M.W., & Shavlik, J.W. (1996). Extracting tree-structured representations of trained networks. *Advances in neural information processing systems* (vol. 8). Cambridge, MA: MIT Press.

Duch, W., Adamczak, R., & Grbczewski, K. (2001). A new methodology of extraction, optimisation and application of crisp and fuzzy logical rules. *IEEE Transactions on Neural Networks, 12,* 277-306.

Engelbrecht, A.P., Rouwhorst, S.E., & Schoeman, L. (2001). A building block approach to genetic programming for rule discovery. In R. Abbass & C. Newton (Eds.), *Data mining: A heuristic approach.* Hershey, PA: Idea Group Publishing.

Fayyad, U., Piatetsky-Shapiro, G., Smyth, P., & Uthurusamy, R. (1996). *Advances in knowledge discovery and data mining.* AAAI/MIT Press.

Fisher, R.A. (1936). The use of multiple measurements in taxonomic problems. *Annals of Eugenics, 7,* 179-188.

Friedberg, R.M. (1958). A learning machine: Part I. *IBM Journal of Research and Development, 2*(1), 2-13.

Friedberg, R.M., Dunham, B., & North, J.H. (1959). A learning machine: Part II. *IBM Journal of Research and Development, 3*(3), 282-287.

Fujiki, C. (1987). Using the genetic algorithm to generate lisp source code to solve the prisoner's dilemma. *International Conf on Gas,* 236-240.

Halgamuge, S.K., & Glesner, M. (1994) Neural networks in designing fuzzy systems for real world applications. *Fuzzy Sets and Systems, 65,* 1-12.

Jagielska, I., Matthews, C., & Whitfort, T. (1996). The application of neural networks, fuzzy logic, genetic algorithms and rough sets to automated knowledge acquisition. *4th Int. Conf. on Soft Computing, IIZUKA'96,* Japan (vol. 2, pp. 565-569).

Jang, J., & Sun, C. (1992). Functional equivalence between radial basis function networks and fuzzy inference systems. *IEEE Transactions on Neural Networks, 4,* 156-158.

Jankowski, N., & Kadirkamanathan, V. (1997). Statistical control of RBF-like networks for classification. *7th International Conference on Artificial Neural Networks,* Lausanne, Switzerland (pp. 385-390).

Kasabov, N. (1996). *Foundations of neural networks, fuzzy systems and knowledge engineering*. Cambridge, MA: MIT Press.

Keedwell, E., Narayanan, A., & Savic, D. (2000). Creating rules from trained neural networks using genetic algorithms. *International Journal of Computers, Systems and Signals (IJCSS), 1*(1), 30-42.

Koza, J. (1992). *Genetic programming: On the programming of computers by means of natural selection.* Cambridge, MA: MIT Press.

Koza, J. (1994). *Genetic programming II: Automatic discovery of reusable programs (complex adaptive systems).* Cambridge, MA: The MIT Press.

Koza, J., Bennett, F.H., Andre, D., & Keane, M.A. (1999). *Genetic programming III: Darwinian invention and problem solving.* Morgan Kauffman.

Mackey, M., & Glass, L. (1977). Oscillation and chaos in physiological control systems. *Science, 197,* 287.

Martínez, A., & Goddard, J. (2001). Definición de una red neuronal para clasificación por medio de un programa evolutivo. *Revista Mexicana de Ingeniería Biomédica, 22*(1), 4-11.

Montana, D.J. (1995). Strongly typed genetic programming. In *Evolutionary computation* (pp. 199-200). Cambridge, MA: MIT Press.

Nauck, D., Nauck, U., & Kruse, R. (1996). Generating classification rules with the neuro-fuzzy system NEFCLASS. *Proceedings of Biennal Conference of the North American Fuzzy Information Processing Society (NAFIPS'96),* Berkeley, CA.

Pop, E., Hayward, R., & Diederich, J. (1994). *RULENEG: Extracting rules from a trained ANN by stepwise negation.* Queensland University of Technology, Neurocomputing Research Centre. QUT NRC Technical report.

Rabuñal, J.R. (1999). *Entrenamiento de Redes de Neuronas Artificiales mediante Algoritmos Genéticos*. Graduate thesis, Facultad de Informática, Universidade da Coruña.

Rabuñal, J.R., Dorado, J., Pazos, A., Pereira, J., & Rivero, D. (2004). A new approach to the extraction of ANN rules and to their generalization capacity through GP. *Neural Computation, 16*(7), 1483-1524.

Shang, N., & Breiman, L. (1996). Distribution based trees are more accurate. *International Conference on Neural Information Processing* (vol. 1, pp. 133-138). Hong Kong.

Ster, B., & Dobnikar, A. (1996). Neural networks in medical diagnosis: Comparaison with other methods. *Proceedings of the International Conference on Engineering Applications of Neural Networks, EANN'96* (pp. 427-430).

Thrun, S. (1995). Extracting rules from networks with distributed representations. In G. Tesauro, D. Touretzky, & T. Leens (Eds.), *Advances in neural information processing systems (NIPS)* (vol. 7). Cambridge, MA: MIT Press.

Tickle, A.B., Andrews, R., Golea, M., & Diederich, J. (1998). The truth will come to light: Directions and challenges in extracting the knowledge embedded within trained artificial neural networks. *IEEE Transaction on Neural Networks, 9*(6), 1057-1068.

Tickle, A.B., Orlowski, M., & Diedrich J. (1996). *DEDEC: A methodology for extracting rules from trained artificial neural networks*. Queensland University of Technology, Neurocomputing Research Centre. QUT NRC Technical report.

Towell, G., & Shavlik, J.W. (1994). Knowledge-based artificial neural networks. *Artificial Intelligence, 70,* 119-165.

Visser, U., Tickle, A., Hayward, R., & Andrews, R. (1996). Rule-extraction from trained neural networks: Different techniques for the determination of herbicides for the plant protection advisory system PRO_PLANT. *Proceedings of the rule extraction from trained artificial neural networks workshop,* Brighton, UK (pp. 133-139).

Weiss, S.M., & Kulikowski, C.A. (1990). *Computer systems that learn.* San Mateo, CA: Morgan Kauffman.

Wong, M.L., & Leung, K.S. (2000). *Data mining using grammar based genetic programming and applications.* Kluwer Academic Publishers.

Chapter VII

Several Approaches to Variable Selection by Means of Genetic Algorithms

Marcos Gestal Pose, University of A Coruña, Spain

Alberto Cancela Carollo, University of A Coruña, Spain

José Manuel Andrade Garda, University of A Coruña, Spain

Mari Paz Gómez-Carracedo, University of A Coruña, Spain

Abstract

This chapter shows several approaches to determine how the most relevant subset of variables can perform a classification task. It will permit the improvement and efficiency of the classification model. A particular technique of evolutionary computation, the genetic algorithms, is applied which aim to obtain a general method of variable selection where only the fitness function will be dependent on the particular problem. The solution proposed is applied and tested on a practical case in the field of analytical chemistry to classify apple beverages.

Introduction

The main goal of any classification method is to establish either the class or the category of a given object, which is defined by some attributes or variables. Nevertheless, not all those attributes give the same quality and quantity of information when the classification is performed. Sometimes, too much information (in this chapter, this term will include both useful and redundant information) can cause problems when assigning an object to one or another class, thus deteriorating the performance of the classification.

The problem of variable selection involves choosing a subgroup of variables from an overall set of them that might carry out the finest classification. Some advantages obtained after a selection process are:

- **Cost reduction for data acquisition:** If less data are required for sample classification, the time required to obtain them would be shorter.
- **Increased efficiency of the classifier system:** Less information also requires less time for its processing.
- **Improved understanding of the classification model:** Those models that use less information to perform the same task will be more thoroughly understood. The simpler the formulation of the classifier, the easier the extraction of the knowledge and its validation.
- **Efficacy improvement:** Sometimes, too much information might deteriorate the generalization ability of the classification method.

Variable Selection

A generic process of variable selection can be formalized by means of the following definitions:

If A is a set of n objects:

$$A = \{x_i, i=1...n\}$$

Each object x_i is described by a set of d variables, V, each one can be either quantitative or qualitative:

$$V = \{V_j, j=1...d\}$$

If C is the overall set of classes, from which discrimination is to be done, and C_k is the class to which object x_i belongs to, then any object x_i can be completely described by the set

$$x_i = \{ V_{ij}, C_k \}, j=1...d$$

The main goal of any variable selection procedure should be to obtain the smallest subset of variables, S ($S \subset V$), that still performs a satisfactory classification task.

Traditionally, this problem has been approached by several statistic techniques, including principal components (Gnanadesikan, 1997), cluster analysis (Jain, Murty, & Flynn, 1999), potential functions (Forina, Armanino, Learde, & Drava, 1991; Tomas & Andrade, 1999), maximum likelihood methods as SIMCA (soft independent modelling of class analogy) (Wold, Johansson, Jellum, Bjørnson, & Nesbakken, 1997), and so forth.

Evolutionary Computation

Evolutionary computation (EC) is composed of a group of techniques inspired on the biological world. As they mimic the evolutionary behaviour of the living species, they work by evolving a group of potential solutions to a given problem until the optimum solution is reached.

More formally, the term EC involves the study of the basis and application of some heuristic techniques that are based on fundamentals of natural evolution (Tomassini, 1995). This group of heuristics can be sorted into four main categories that are included in the evolutionary equation that is shown in Figure 1.

Biological Fundamentals

As it has been stated, evolutionary algorithms, in origin, tried to mimic some of the processes that take place in natural evolution. Some of the most remarkable ones are the survival of the best individuals and natural selection, both contributed by Charles Darwin (Darwin, 1859). Although the details of the natural evolutionary process remain

Figure 1. Evolutionary equation

Evolutionary Computation	=	Genetic Algorithms	+	Evolutionary Strategies	+	Genetic Programming	+	Evolutionary Programming

unknown and are not completely understood yet, the events that EC take as its basis have strong experimental evidences:

- Evolution is a process that deals with chromosomes rather than with organisms. The chromosomes might be considered as organic tools that code life or, from a computational viewpoint, a problem's solution.
- Natural selection is the mechanism that links chromosomes with efficiency, bearing in mind the entity that they represent. It provides the individuals best adapted to the environment with a higher number of opportunities for reproduction.
- The evolutionary processes take place during the reproductive stage.

There are numerous mechanisms related to reproduction, the most common ones being crossover or recombination, which combines the chromosomes of the parents in order to produce the offspring, and mutation, which randomly induces the genetic material of the offspring to be different from that of the parents.

Basic Operation of an Evolutionary Algorithm

An evolutionary algorithm is a recurrent and stochastic process that operates with a group of potential solutions to a problem known as genetic population (Goldberg, 1989).

Initially, this population is generated randomly and the solutions are evolving continuously after consecutive stages of crossovers and mutations. Every individual in the population has a value that is associated to its adjustment or fitness, in accordance to its adequacy to solve the problem. This value, which has to be obtained individually for each potential solution, is the quantitative information the evolutionary algorithm will use to guide the search. The process will continue until a predetermined stopping criterion is reached. This might be the achievement of a particular threshold error for the solution or a certain number of generations. This process is described in Figure 2.

Techniques of Evolutionary Computation for Variable Selection

Among the strategies that may be followed to select a subset of variables, the EC technique known as genetic algorithms (GA) stands out. Here, only those aspects related to the encoding of the problem will be commented. As it can be observed on the pseudocode shown in Figure 2, it is necessary to establish a mechanism to determine to what extent an individual of the genetic population yields a good solution. This means that an evaluation function should be provided, although the particular one to be used will depend on the specific problem at hand.

Figure 2. Pseudocode of a basic evolutionary algorithm

```
Generate initial population P(0)
t ← 0
While stop criterion is not fulfilled
    Evaluate P(t)
    P'(t) ← Select (P(t))
    P''(t) ← Reproduce (P'(t))
    P(t+1) ← Substitute for (P(t), P''(t))
    t ← t + 1
EndWhile
Return the best solution at P(t)
```

In the next section, the problem of variable selection in the field of IR (infrared) spectrometry will be approached by means of a practical example in which the evaluation function has to be described for each different variable selection approach.

Pruned Search

In this first approach, a GA will be used to determine which variables contribute most to the determination of the class of each object and how many they are. The GA will steadily reduce the amount of variables that characterise the objects, until getting an optimal subset that allows an overall satisfactory classification.

In this approach, each individual in the genetic population is described by n genes, each representing one variable. The binary encoding each gene might be either 0 or 1, indicating whether the gene is active or not and, therefore, if the variable should be considered for classification.

The evaluation function has to guide the pruning process in order to get individuals with a low number of variables. To achieve this, the function should help those individuals that, besides classifying accurately, make use of fewer variables. An option is described by Equation 1. It defines fitness according to two parameters: the number of variables used by the individual to classify the samples and the quality of the classification performed employing them. The best individuals will be those with fitness closer to 0 because the evaluation function will guide the search of the GA along two parallel directions. The solutions should minimize the difference between the number of classification errors obtained using all the variables and the number of errors when only the GA-selected variables are employed. As the solutions that use fewer variables are preferred, it can be stated that the lower these two aspects of the function, the better the solution codified by the individual will be.

Equation 1.

Pruned search. Generic evaluation.

$$fitness_i = f(classification_i) + f(SelectedVars)$$

The metrics to be used when evaluating the quality of the classification will depend on the problem at hand, among them the percentage of objects that have been correctly classified (using only the selected variables), the number of errors, and so forth. There are two options when the subset of variables that have been selected for classification is to be considered: the use of the cardinality of this subset and its percentage regarding the whole of the existing variables.

If the function that evaluates the quality of the classification can be delimited by an upper value (e.g., the total number of errors) it would be possible to weight the importance of each of the members of the equation. The fitness function then would be like Equation 2, where α represents a weighting factor.

Equation 2.

Pruned search. Generic evaluation with weighting.

$$fitness_{Ind_i} = \alpha \cdot f(classification_i) + (1-\alpha) \cdot f(SelectedVars) \quad \alpha \in [0..1]$$

Fixed Search

In this second approach, the GA will only determine which are the outstanding variables for classification, as the encoding of the individuals will determine in advance how many they are. In order to get this previous knowledge, a Procrustes rotation method might be used to determine the minimal number of variables needed for a satisfactory classification (Krzanowski, 2001). This number will define the length of the genetic individual from where the configuration of the GA can be performed. As the genotype determines how many variables are used for classification, the GA will only be focused on identifying them.

Quite distinct from the previous approach, this one uses a real encoding for the chromosome of the genetic individuals. Here, every gene represents the index of the variable that is considered representative for classification.

Using this alternative, the evaluation function will depend only on the first term, which is dependant on the quality of the classification, because the cardinality of the subgroup of variables is predefined.

Multimodal Search

In some occasions, the solution of a problem is not represented by a unique local optimum but for several ones. In some other cases, it is preferable to provide several solutions,

so that the expert in charge of the implementation may choose, maybe not the best one (mathematically, either the absolute maximum or minimum) but the one that provides less complexity in practice with equivalent results. In such cases, the problems are known as multimodal problems.

A multimodal problem represents two challenges for the performance of a basic GA (Harik, 1995). First, this type of problem usually has a high number of variables, deteriorating the performance of a basic GA. Second, a traditional GA tends to focus the search along the area where the most promising solution is, therefore eliminating less promising areas as generations advance. Some of the modifications that address these limitations are the messy GA (Whitley, 1997), that appears to be stronger when working with a high number of variables, and the multimodal techniques as crowding (DeJong, 1975) that seek new schemes for the generation of individuals or for their substitutes in order to avoid the homogenization of the population. Another modification is fitness sharing (Sareni, 1998), which tries to keep the diversity of the population during the exploration of multiple peaks.

Our own approach to use GA to solve multimodal problems is a hybrid two-population GA (HTP-GA). Briefly, this solution adds a new population — named as genetic pool — to a traditional GA. Each individual on this new population represents a sub-region of the searching space (the overall space is constituted by all the possible solutions; in the present case, all the combinations of variables). These individuals have the property that their genes can vary only within a restricted range of values, here a limited set of variables (e.g., variable ranges [1..4], [5..8], etc.). This forces the distribution of the genetic individuals along the entire search space, something that protects the diversity of the population as generations go by.

As a consequence of the homogeneous distribution of the individuals along the searching space, the GA may find — and keep into the genetic population — valid solutions, distancing each other within the search space, whereas when using a traditional GA, it tends to focus the search within a restricted region as generations advance.

The HTP-GA approach can be detailed in the next subsection.

Hybrid Two-Population Genetic Algorithm

The proposed GA is based on a traditional GA that is augmented with a new population (the *genetic pool*) in order to force a homogeneous search along the entire space of solutions.

In addition to the genetic pool, another population (the *secondary population*) is also used. The secondary population evolves, guided by a classical GA in connection with the individuals of the genetic pool. The secondary population will be in charge of giving the solutions, whereas the genetic pool will act as an "engine" for the search of these solutions, by keeping the searching space homogeneously explored.

Depending on the complexity of the problem, the user will have to test a different number of sub-regions he or she wants on the search space (or the number of individuals on the

genetic pool or the length of the sub-ranges). Note that a traditional GA considers only one sub-region (the entire search space) which becomes successively reduced.

Description of the Populations: Genetic Pool

Since its individuals represent sub-regions within the global searching space, they have the same structure (or gene sequence) than a traditional GA.

The difference lies in the values the genes may have. While in a traditional GA any value within the search space may be valid, the range of values is restricted for the genes of the individuals of the genetic pool. The total range of values (e.g., the total number of variables) is divided into as many ranges as individuals on the genetic pool. Accordingly, each individual on the pool will have its own sub-range of values assigned. Then, each gene of the genetic pool individuals can only take values contained on its associated range.

Each individual on the genetic pool has to know which of its genes are included into the best solution of the secondary population (this has to be updated for each generation). This knowledge is given in the form of Boolean values which avoid modification of the genes of the best solution.

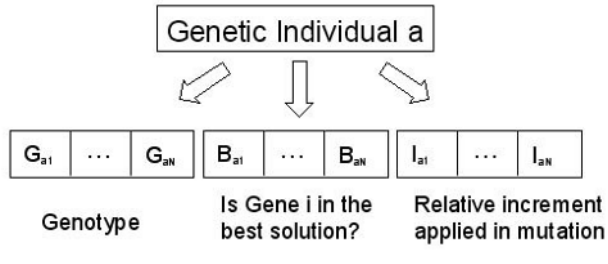
Each gene of each genetic pool individual has an associated I value which indicates the relative increment that will be applied to the gene during a mutation. It is obvious that this increment cannot make the value of the gene to exceed the range in which this gene is allowed to vary. The structure of the genetic pool individuals is shown in Figure 3.

As these individuals do not represent global solutions to the problem that has to be solved, evaluating their fitness is not compulsory.

Description of the Populations: Secondary Population

Unlike the genetic pool, the genes of the individuals on the secondary population can take any value throughout the whole space of possible solutions. In this way, the secondary population offers global solutions to the problem, which none of the

Figure 3. Chromosome of the individuals in the genetic pool



individuals at the genetic pool can do because their genes are restricted to different sub-ranges.

The evolution of the individuals at this population will be carried out by the traditional GA rules but using a different crossover since the two parents will not belong to the same population. Hence, the genetic pool and secondary population are combined instead.

Genetic Operators in HTP-GA

As in classical GA, the development of solutions in HTP-GA is carried out applying crossover and mutation operators. The unique differences are in crossover (as already discussed) and, less, in the mutation operator.

Crossover

It combines an individual from the secondary population with a representative of the genetic pool. We propose the former to be chosen at random. Therefore, there is no need for the secondary population to be ordered by the fitness of the individuals, and in consequence, there will be an improvement on the execution time. A genetic pool representative is constructed in such a way that all the genes of the genetic pool are considered. Each gene of the representative is randomly and independently chosen among all the individuals (see Figure 4). The representative is, unlike the genetic pool individuals, not a partial but a global solution.

Then, a uniform crossover is performed. Its implementation yields only one individual (because only one individual was extracted from the secondary population), whose genes were randomly selected from its parents. Insertion into the secondary population has been done by means of a nondestructive replacement algorithm in such a way that the insertion is solely performed when the fitness of the offspring is higher than that for the parent of the secondary population.

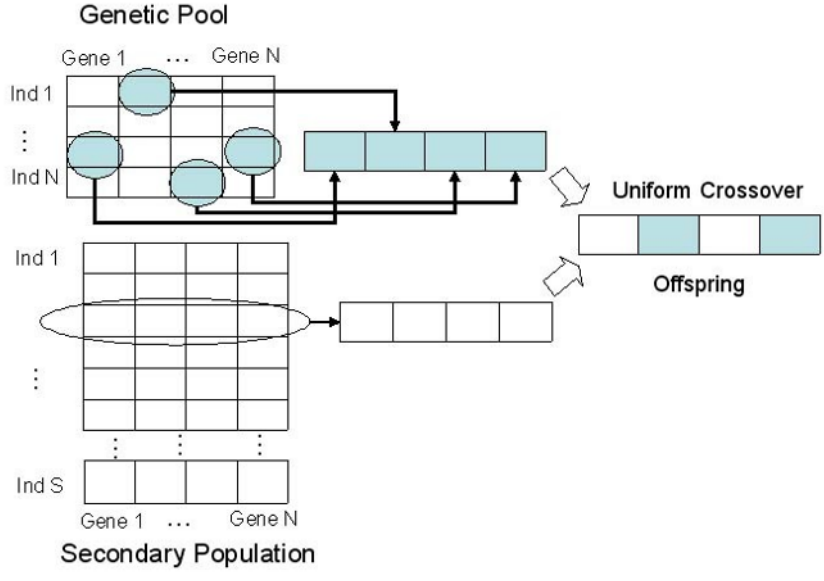
If crossover leads to a better solution than those already stored, the original genes taken from the genetic pool should not change in the next generation and, so, their corresponding "control" or Boolean genes (B_{ij} in Figure 3) change accordingly.

Mutation

Application of the mutation operator involves a random selection of one individual from the genetic pool in order to mutate one of its genes. The new value results from increasing (decreasing) the old value by the increment specified on its corresponding "incremental gene" (labeled as I in Figure 3), see pseudo-code in Figure 5. Genes considered as the best solutions of the secondary population should not mutate, as explained.

If the increment makes the value of the gene to exceed the upper limit of the sub-range in which the gene is allowed to vary (LIM_SUP_IND), it is changed to the lowest value of the sub-range (LIM_INF_IND). Furthermore, the incremental value that might be applied on subsequent mutations of that gene is also reduced (controlled by a "Delta" value). This tries to do a more exhaustive search through all the values that a given gene

Figure 4. Crossover



may have as generations advance. The Delta value is initialised according to Equation 3. Mutation, therefore, will be applied only to individuals belonging to the genetic pool.

Equation 3.

Delta initialization, IND_POOL = number of genetic pool individuals

$$\frac{(LIM\_SUP\_IND)-(LIM\_INF\_IND)}{IND\_POOL}$$

Practical Application: Selection of IR Spectral Variables to Classify Apple Beverages

As consumption of fruit juices has exploded over the past 10 years, associated fraudulent techniques aimed at achieving higher profits (e.g., inappropriate labeling or sugar addition) increased as well. In an attempt to stop this trend, new European Union (EU) directives established compulsory informative data to be printed out on the individual containers such as fruit origin and amount of pure juice.

Figure 5. Pseudocode for mutation

```
IF (not Bi)
  Gi = Gi + Ii
  IF (Gi > LIM_SUP_GEN)
   Gi = LIM_INF_GEN
   Ii = Ii – Delta
  ENDIF
ENDIF
```

The addition of sugar is the most common and simple method for juice adulteration. Other methods may be the addition of several substances as water, colorants, lower quality juices, or fruit pulp (Saavedra, García, & Barbas, 2000).

There are different techniques for the detection of these irregularities, such as HPLC (Yuan & Chen, 1999), gas chromatography (Stöber, Martin, & Pepper, 1998), or isotopic methods (Jamin, González, Remaud, Naulet, & Martin, 1997), that characterize a fruit juice from a qualitative and quantitative viewpoint. Their main problem is that all of them are complex analytical techniques that also require a lot of time and money.

Another methodology that can be used is the infrared spectroscopy (IR), which might measure the full spectrum of the sample within some seconds the samples. Spectroscopy is based on differential light reflection and/or absorption of the objects. A sample will respond differently to distinct wavelengths. The differences are due to the chemical composition and the structure of the molecules that are analysed. Therefore, the discrimination of samples with different concentrations of natural juice might be a previous step for the detection of adulterated juices.

The quantity of information extracted from a sample by IR is huge. In the present practical application, the small spectral range that was measured (wavelengths from 1,250 cm^{-1} to 900 cm^{-1}) provides 176 absorbances (variables that measure light absorption). Figure 6 shows a typical example of an apple juice. It is quite complex to establish whether all those data provide the same amount of information for sample differentiation, thus the application of mathematical and/or computational methods takes quite a lot of time. Accordingly, the spectral characterisation of fruit juices becomes an appropriate case for the use of variable selection techniques, which are intended to decrease the time required for the experimental measurement, for the application of mathematical methods, and also to extract chemical conclusions regarding those wavelengths that provide a higher amount of information for sample characterisation.

Samples

Previous to variable selection construction of datasets for both model development and validation is mandatory. Thus, samples with different amounts of pure apple juice were

Figure 6. Typical spectrum of an apple juice

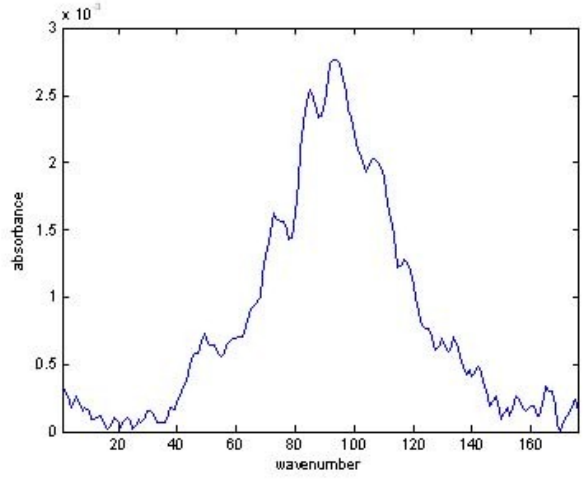


Table 1. Number of samples with low concentrations of pure juice

Low Concentrations 2% - 20%			
Concentration	*Training*	*Validation*	*Commercial*
2%	19	1	0
4%	17	1	0
6%	16	13	0
8%	22	6	0
10%	21	6	1
16%	20	6	1
20%	19	6	0
Total	**134**	**39**	**2**

prepared in the laboratory. Twenty-three apple juice-based beverages sold in Spain were purchased and analysed (their declared amount of juice was printed on their labels). The samples were distributed in two ranges: samples containing less than 20% of pure juice (Table 1) and samples with more than 20% of pure apple juice (Table 2). IR spectra were obtained using a Perkin Elmer 16PC FTIR spectrometer, equipped with a horizontal ZnSe ATR plate (50 scans, 4 cm^{-1} resolution, strong apodization, baseline correction between 1,250 to 900 cm^{-1}, DTGS detector). The spectra were digitized (1 datum/2 cm^{-1}).

Independent different datasets were used to extract the rules (ANN training) and to validate. The commercial samples were used to further check the performance of the

Table 2. Number of samples with high concentrations of pure juice

High Concentrations 20% - 100%			
Concentration	*Trainning*	*Validation*	*Commercial*
20%	20	6	0
25%	19	18	2
50%	16	13	0
70%	14	1	0
100%	17	6	19
Total	86	41	21

model. It is worth noting that if a predicted value does not match that given on the labels of the commercial products it might be owing to either a true wrong performance of the model (classification) or an inaccurate labeling of the commercial beverage.

Evaluation of the Solution

Previous studies (Gestal et al., 2004; Gestal et al., 2005) showed that ANN classify apple juice beverages according to the concentration of natural juice they contained and that ANN have advantages over classical statistical methods, such as faster model development and easy application of the methodology on R&D laboratories.

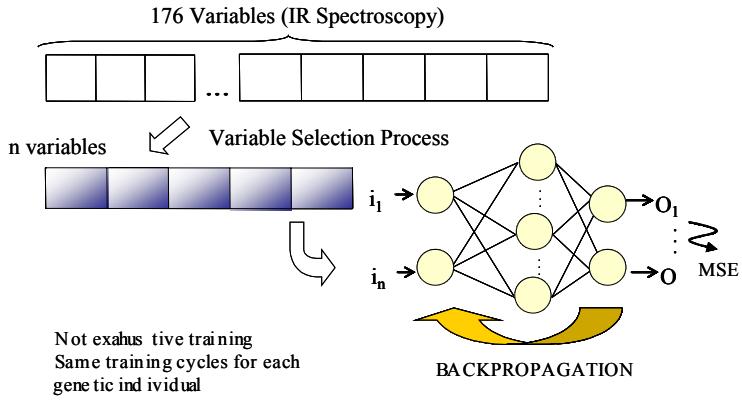
Here, GA have been considered to guide the search of the minimum number of variables by evaluating the prediction capabilities of different ANN models developed employing different sets of IR variables. In the previous sections, three alternatives were described to select variables (namely, pruned search, fixed search, and multimodal HTP-GA search) and they will be tested hereinafter (see Figure 7 for a representation of the overall process).

The problem to be addressed is to find out a small set of IR variables that, when combined with an ANN model, are capable of classifying apple juice-based beverages properly (according to the amount of pure apple juice they contain).

Once a subset of IR variables is proposed, their patterns are inserted into the ANN (a pattern consists of the absorbance values that are associated to the variables).

The ANN considers as much input processing elements (PEs) as variables. The output layer has one PE per category (six for the lower range and five for the higher concentrations). After several previous trials considering several hidden layers (from 1 to 4), each with different PEs (from 1 to 50), a compromise was established between the final fitness level reached by the ANN and the time required to its training. This compromise was essential because although the better results were obtained with more hidden layers, the time required for training was much higher as well. Then, it was decided not to extensively train the net but to get a good approximation to its real performance and elucidate whether the input variables are really suitable for the accurate classification of the samples. The

Figure 7. Scheme to evaluate genetic individuals



important point here is that all the ANN have the same structure in order to be able to compare their results.

The goal is to determine which solutions, among those provided by the GA, represent good starting points. Therefore, it would be enough to extend the ANN learning up to the point where it starts to converge. For this particular problem, convergence started after 800 cycles; in order to warrant that this step is reached, 1,000 iterations were fixed.

In order to avoid the influence of local minima as much as possible during training, every training period is repeated three times and the worst results are discarded.

The mean square error (MSE) statistic obtained from the best training will be used to establish the fitness of the genetic individual. The overall fitness value will depend on the strategy of variable selection, as it is shown.

Pruned Search

When fitness of an individual has to be determined, to evaluate the successful classification ratio of the ANN model is only part of the problem (which can be addressed by using the MSE). The other part is to "help" those individuals that perform a satisfactory classification with fewer variables.

Hence, the fitness of a genetic individual would be formulated as Equation 4.

Equation 4.

Evaluation function for a pruned search approach.

$$fitness_{Ind\,i} = MSE(ANN_i) + \frac{\#1's\ genotype\ Ind_i}{176}$$

Fixed Search and Multimodal Searches

Here, the number of variables is fixed in advance so that the second term in Equation 4 is not needed any more (Equation 5).

Equation 5.

Evaluation function for both, fixed and multimodal searches approaches.

$$fitness_{Ind\,i} = MSE(RNA_i)$$

Refining the GA Searches by Focusing on Significant Variables

As it is well-known, the GA searches are powerful searching techniques capable of good mathematical solutions even in very complex situations. This, in turn, carries out a disadvantage, which is that many times the end user of the methodology cannot interpret the results. This is an important issue because they need to understand why a given sample is not properly classified or, worst, if the classification given by the control laboratory do not agree with that declared on the label. In this particular case study, the original IR spectral variables can be chemically associated to different carbohydrates (mainly sugars) and, so, there is a need to stress chemical understanding of what-is-going-on or — at least — the final result. This arose more relevant when the fixed and multimodal searches were used.

Figure 8. Selected variables and areas chemically related to main carbohydrates

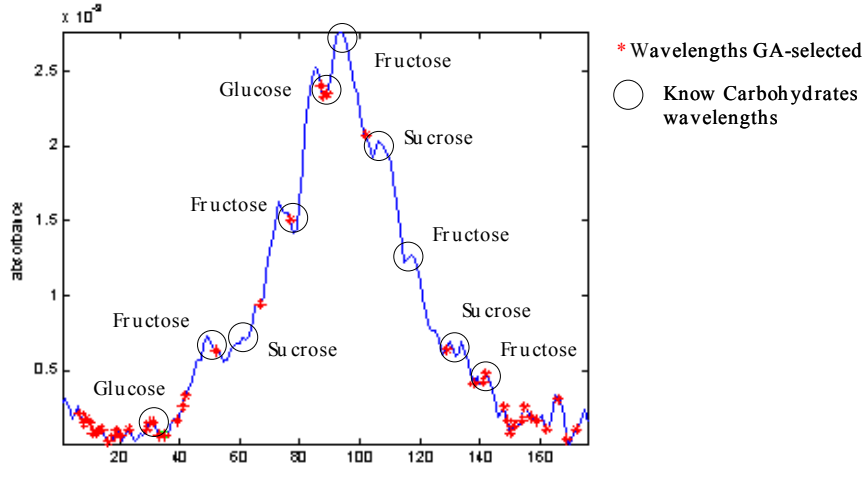


Figure 9. Pseudo-code for the mutation operator

```
nCarbohydrates= 10   // 10 carbohydrates variables
   Ind_i[1..nGenes]    // Individual to be evaluated
   CPos[1.. nCarbohydrates] // Carbohydrates_Position

   minDistance = MAX_INT

   MSE ← ANN Classification Error for Ind_i

   For every g gene      // g∈Ind_i
    For every position p   // p∈Carbohydrates_Position
                Dist_gp = |Ind_i[g]-cPos[p]|
                IF (Dist_gp < minDistance)
                THEN minDistance  = Dist_gp
  END FOR
   END FOR

   Fitness_i = MSE + minDistance
```

Accordingly, a new term was included on the fitness evaluation in order to achieve this objective.

The variables that are associated with major carbohydrates (glucose, fructose, or sucrose) are known in advance and, so, the GA will be informed about 10 carbohydrates positions (Figure 8). The 10 variables that identify the carbohydrates are stored at the carbohydrates position vector (no change during the tests). Now, the fitness of an individual can be evaluated not only by the MSE but for the distance between each gene of the individual and the closest "theoretical" positions of the carbohydrates. Those individuals for which any of these combinations is minimal, will be benefited. The main goal is that, as long as the quality of the classification remains good, the chemical interpretation gets highly simplified. As fitness is evaluated by a minimization function, the preferred individuals would be those minimizing both the MSE and the distance between, at least, one of the selected variables and one of the "theoretical" variables at the carbohydrates position vector, as shown in Figure 9.

Results and Discussion

As a reference situation is needed to perform comparisons, the results of the classification achieved using the 176 original variables are presented first (Table 3). They were obtained optimizing an ANN model as usual (testing the number of hidden layers, the number of neurons on each layer, learning rate, etc.).

Table 3. Classifications using all the 176 original variables

Concentration	Training	Validation	Commercial
	PLS		
2%-20%	#LV = 2, n=134, #errors= 29	n=39, #errors = 11	n=2, #errors=0
25%-100%	#LV = 2, n=86, #errors=11	n=44, #errors = 5	n=21, #errors=1
	SIMCA		
2%-20%	# factors for each class: 2%: 1 4%: 2 6%: 3 8%: 4 10%: 2 16%: 3 20%: 4 n=134, #errors=19	n=39, #errors=14	n=2, #errors=0
25%-100%	# factors for each class: 20%: 2 25%: 2 70%: 3 100%: 4 n=86, #errors=15	n=44, #errors=12	n=21, #errors=1
	POTENTIAL CURVES		
2%-20%	factors subspace: PC1-PC2 n=134, #errors=4	n=39, #errors=9	n=2 , #errors=0
25%-100%	factors subspace: PC1-PC2 n=86, #errors=4	n=44, #errors=6	n=21 , #errors=0
	ARTIFICIAL NEURAL NETWORKS		
2%-20%	ANN topology =176/50+80/7 lr=0.0005 , mseThreshold=5 n=134, #errors=0	n=39, #errors=4	n=2 , #errors=0
25%-100%	ANN topology= 176/8+5/5 lr=0.001 , mseThreshold =1 n=86, #errors = 0	n=44, #errors=1	n=21 , #errors=0

n=number of samples; #=cardinality; lr=learning rate; LV=latent variables; mseThreshold: mse threshold used to stop training (if it does not stop after 500.000 cycles)

In order to decide the minimum number of variables that can be consider, an external criterion was needed. Here, a Procrustes rotation algorithm was employed (Krzanowski, 2001) and two principal components were considered good enough to describe the dataset (99.76% of the total variance). Hence, two original variables should be the minimum to be reached.

Classification Considering All the Original Variables

Different classification methods typically used in analytical chemistry were performed (see Table 3). It was concluded that the set of samples with low percentages (2 to 20%)

Table 4. Classification results after pruned search

Concentration	Training	Validation	Commercial
Selected Variables: [42 77]			
2%-20%	topology ANN=2/10+60/7 lr=0.001, mseThreshold=5 n=134, #errors = 5	n=39, #errors=16	N=2 , #errors=1
25%-100%	topology ANN=2/10+60/5 lr=0.001, mseThreshold=5 n=86, #errors= 3	n=44, #errors=9	n=21 , #errors=0
Selected Variables: [52 141]			
2%-20%	topology ANN=2/10+60/7 lr=0.001, mseThreshold=5 n=134, #errors=19	n=39, #errors=17	N=2 , #errors=2
25%-100%	topology ANN=2/10+60/5 lr=0.001 , mseThreshold=5 n=86, #errors=4	n=44, #errors = 8	N=21 , #errors=1
Selected Variables: [102 129]			
2%-20%	topology ANN=2/10+60/7 lr=0.001, mseThreshold=5 n=134, #errors=10	n=39, # errors =14	N=2 , #errors=2
25%-100%	topology ANN=2/10+60/5 lr=0.001, mseThreshold=5 n=86, #errors=3	n=44, #errors = 10	N=21 , #errors=0

n=number of samples; #=cardinality; lr=learning rate; mseThreshold: mse threshold used to stop training (if it does not stop after 500.000 cycles)

of juice are far more complex and difficult to classify than the samples with higher concentrations (25 to 100%). Indeed, the number of errors was higher for the 2 to 20% range both in calibration and validation.

Classification of the commercial samples agreed quite well with the percentages of juice declared in the labels, but for a particular sample. When that sample was studied in detail it was observed that its spectrum was slightly different from the usual ones. This would suggest that the juice contained an unusually high amount of added sugar(s).

If the information obtained by the different classification methods is compared, it can be stated that the best results were achieved using ANN, which is very useful to address the processes of variable selection employing GA with fitness functions based on ANN.

Classification After Pruned Search

Table 4 shows some classification results obtained from different ANN models using sets of two variables selected by the Pruned Search-GA strategy. Although several trials were performed, Table 4 shows only the three most satisfactory ones. They are worse than

Table 5. Classification results after fixed search

Concentration	Training	Validation	Commercial
Selected Variables: [12 159]			
2%-20%	topology(*)= 2/10+60/7 lr=0.001 , me = 5 n=134, #errors = 9	n=39, #errors=16	n=2 , #errors=2
25%-100%	topology(*)= 2/10+60/5 lr=0.001, me =5 n=86, #errors = 4	n=44, #errors=12	n=21 , #errors=2
Selected Variables: [23 67]			
2%-20%	topology ANN=2/10+60/7 lr=0.001 , mseThreshold=5 n=134, #errors=5	n=39, # errors=16	n = 2 , #errors=1
25%-100%	topology ANN=2/10+60/5 lr=0.001 , mseThreshold=5 n=86, #errors=3	n=44, # errors=12	n = 21 , #errors=2
Selected Variables: [102 129]			
2%-20%	topology ANN=2/10+60/7 lr=0.001 , mseThreshold=5 n=134, #errors=10	n=39, #errors=14	n = 2 , #errors=2
25%-100%	topology ANN=2/10+60/5 lr=0.001 , mseThreshold=5 n=86, #errors=3	n=44, #errors=10	n = 21 , #errors=0

n=number of samples; #=cardinality; lr=learning rate; mseThreshold: mse threshold used to stop training (if it does not stop after 500.000 cycles)

those from Table 1, especially for the validation datasets. Nevertheless, the classification of commercial samples remains satisfactory.

Classification of the same datasets was also performed by PLS, SIMCA, and potential curves employing the same pairs of variables. Although they are not detailed here, the ANN models were superior (Gestal et al., 2005), quite probably because the GA uses an ANN to evaluate the fitness of each solution and, therefore, select the best variables to precisely minimize the classification using an ANN afterwards. Accordingly, we will only present results obtained using ANN.

Classification After Fixed Search

Three good solutions are summarized in Table 5. As in the previous section, not all the trials that were performed are shown. It can be observed that ANN yielded results considerably worse than those obtained by pruned search. The same occurred when other classification methods were attempted (more details are given in Gestal et al., 2005). It was concluded, therefore, that the fixed search approach is not a satisfactory way to find out the best subset of variables (in this case, two variables).

Table 6. Classification results after multimodal search

Concentration	Training	Validation	Commercial
Selected Variables: [89 102]			
2%-20%	Topology ANN=2/10+60/7 lr=0.001 , mseThreshold=2 n=134, #errors=7	n=39, #errors=10	n = 2 , #errors=2
25%-100%	Topology ANN=2/10+60/5 lr=0.001 , mseThreshold=2 n=86, #errors=3	n=44, #errors=8	n = 21 , #errors=0
Selected Variables: [87 102]			
2%-20%	Topology ANN=2/10+60/7 lr=0.001 , mseThreshold=2 n=134, #errors=4	n=39, #errors=11	n = 2 , #errors=2
25%-100%	topology ANN=2/10+60/5 lr=0.001 , mseThreshold=2 n=86, #errors=3	n=44, #errors=7	n = 21 , #errors=0
Selected Variables: [88 89]			
2%-20%	topology ANN=2/10+60/7 lr=0.001 , mseThreshold=2 n=134, #errors=14	n=39, #errors=10	n = 2 , #errors=2
25%-100%	topology ANN=2/10+60/5 lr=0.001 , mseThreshold=2 n=86, #errors=4	n=44, #errors=4	n = 21 , #errors=0

n=number of samples; #=cardinality; lr=learning rate; mseThreshold: mse threshold used to stop training (if it does not stop after 500.000 cycles)

The fixed search approach considering chromosomes with two genes has the disadvantage that as any crossover operator will use the unique available crossing point (between the genes) only half of the information from each parent would be transmitted to its offspring. This converts the fixed search approach into a random search when only two genes constitute the chromosome. Indeed, this type of search might not be able to join two variables, which not being good solutions by themselves, yield a good solution once they become combined. In that sense, this approach works as a deceptive problem (Goldberg, 1987).

Classification After Multimodal Search

As discussed previously, there are situations where the final model is extracted after analysing different solutions of the same problem. An example is given in Table 6. After repeating the analysis several times (c.a. 20), it was observed that the solutions concentrated along specific spectral areas. Positively enough, the results were clearly superior to those obtained with the previous alternatives.

Table 7. Classification results after multimodal search with points of interest

Concentration	Training	Validation	Commercial
Selected Variables: [88 92]			
2%-20%	topology ANN=2/10+60/7 lr=0.001 , mseThreshold=2 n=134, #errors=7	n=39, #errors=9	n = 2 , #errors=2
25%-100%	topology ANN=2/10+60/5 lr=0.001 , mseThreshold=2 n=86, #errors=3	n=44, #errors=8	n = 21 , #errors=0
Selected Variables: [88 102]			
2%-20%	topology ANN=2/10+60/7 lr=0.001 , mseThreshold=2 n=134, #errors=4	n=39, #errors=10	n = 2 , #errors=2
25%-100%	topology ANN=2/10+60/5 lr=0.001 , mseThreshold=2 n=86, #errors=3	n=44, #errors=7	n = 21 , #errors=0
Selected Variables: [85 102]			
2%-20%	topology ANN=2/10+60/7 lr=0.001 , mseThreshold=2 n=134, #errors=0	n=39, #errors=11	n = 2 , #errors=2
25%-100%	topology ANN=2/10+60/5 lr=0.001 , mseThreshold=2 n=86, #errors=2	n=44, #errors=7	n = 21 , #errors=0

n=number of samples; #=cardinality; lr=learning rate; mseThreshold: mse threshold used to stop training (if it does not stop after 500.000 cycles)

Classification After Multimodal Search with Points of Interest

The three previous approaches led to quite similar results, pointing toward the existence of a (hidden) rule that might determine such performance. Nevertheless, no other relationship was found among the variables than a trend to be extracted from spectral areas where the variance of the signal (absorbance) is higher along the different calibration samples, which, in turn, could be the key for a satisfactory classification.

The idea has a good chemical background, although, disappointingly, many selected variables concentrated on one of the extremes of the spectrum (Figure 9). These areas mainly are related to spectral noise (noise has a — proportionally — large variance but uncorrelated to the classification problem). To avoid this problem, the GA was forced to skip those variables whose indexes were lower than 25 or higher than 140, and the multimodal search was performed again.

Once the noisiest variables were avoided, a second modification was included to improve the chemical understanding of what was going on. Not in vain, it had been observed many

times that the selected variables did not have a chemical meaning, so we could not determine which carbohydrates defined precisely the quantity of apple juice at the beverages. In order to solve this major chemical issue, the GA was fed information regarding the spectral areas that can be associated specifically to the most important carbohydrates (Figure 9). In such a way, the GA will favour those individuals whose variables are closer to those of the carbohydrates.

Table 7 resumes the results of several studies. It can be observed that the classification success is similar — or even slightly superior — to those obtained with multimodal search alone. But, now, the main carbohydrate/s (sugar/s) responsible for the classification model is/are known.

Conclusions

Several conclusions can be derived from the results obtained with the use of different proposals for variable selection:

- The EC techniques, particularly GA, demonstrated that they are a valid way to perform variable selection, since their results were, in general, acceptable.
- The classification success for the pruned search approach is greater than for the fixed search one. This can be explained because the likelihood of joining variables that, together, can offer a good result is lower when the chromosome has only two genes, than when, on the contrary, a sequential — pruned — search is applied. In the latter case, a progressive elimination of variables is undergone until (in the case study presented here) the two most important variables remain.
- Best results were obtained using a multimodal GA, likely because of its ability to maintain the diversity of the genetic population. Such diversity not only induces the appearance of optimal solutions, but also avoids the search to stop on a local minimum. This option not only provides a solution, but a group of them with similar fitness. This allows chemists to select a solution with a sound chemical background. A particular, highly appealing approach is to include "points of interest" into the GA searching strategy, namely the HTP-GA.

References

Braakman, L. (2005). What to get ready for. *Food Engineering & Ingredients, 27*(6), 14-19.

Darwin, C. (1859). *On the origin of species by means of natural selection.*

DeJong, K.A. (1975). *An analysis of the behavior of a class of genetic adaptative systems.* Ph.D. dissertation, University of Michigan.

Dorado, J., Rabuñal, J.R., Pazos, A., Santos, A., & Pedreira, N. (2002). Algoritmo genético de dos poblaciones para búsqueda homogénea en el espacio de soluciones. Proceedings of *I Congreso Nacional de Algoritmos Evolutivos y Bioinspirados (AEB'02)*, Mérida, Spain (pp. 241-245).

Forina, M., Armanino, C., Learde, R., & Drava, G. (1991). A class-modelling technique based on potential functions. *J. Chemometrics, 5, 435-453*.

Gestal, M., Gómez-Carracedo, M.P., Andrade, J.M., Dorado, J., Fernández, E., Prada, D., & Pazos, A. (2004). Classification of apple beverages using artificial neural networks with previous variable selection. *Analytica Chimica Acta*, 225-234.

Gestal, M., Gómez-Carracedo, M.P., Andrade, J.M., Dorado, J., Fernández, E., Prada, D., & Pazos, A. (2005). Selection of variables by genetic algorithms to classify apple beverages by artificial neural networks. *Applied Artificial Intelligence*, 181-198.

Gestal, M., Rabuñal, J.R., Dorado, J., & Pazos A. (2004). Exploración sistemática del espacio de búsqueda mediante un algoritmo genético de 2 poblaciones. *Proceedings of III Congreso Nacional de Metaheurísticas, Algoritmos Evolutivos y Bioinspirados (MAEB'04)*, Córdoba, Spain (pp. 286-291).

Gnanadesikan, R. (1997). *Methods for statistical data analysis of multivariate observations*. New York: John Wiley & Sons.

Goldberg, D. (1987). Simple genetic algorithms and the minimal deceptive problem. In L. Davis (Ed.), *Genetic algorithms and simulated annealing* (pp. 74-88). London.

Goldberg, D. (1989). *Genetic algorithms in search, optimization and machine learning*. Addison-Wesley.

Harik, G. (1995). Finding multimodal solutions using restricted tournament selection. *Proceedings of the Sixth International Conference on Genetic Algorithms (ICGA95)* (pp. 24-31).

Jain, A.K., Murty, M.N., & Flynn, P.J. (1999). Data clustering: A review. *ACM Computing Surveys, 31*(3), 264-323.

Jamin, E., González, J., Remaud, G., Naulet, N., & Martin, G. (1997). *Journal of Agricultural Food Chemistry, 45*(10), 3961-3967.

Krzanowski, W.J. (2001). *Principles of multivariate analysis: A user's perspective*. New York: Oxford University Press.

Saavedra, L., García, A., & Barbas, C. (2000). *Journal of Chromatography Analitycal,* 395-401.

Sareni, B., & Krähenbühl. (1998). Fitness sharing and niching methods. *IEEE Transactions on Evolutionary Computation, 2*(3), 97-106.

Stöber, P., Martin, G., & Pepper, T.L. (1998). *Deutsche Labensmittel-Rundschau, 94*(9), 309-316.

Tomás i Morer, X., & Andrade Garda, J.M. (1999). Application of simplified potential curves to classification problems. *Química Analítica 18,* 117-120.

Tomassini, M. (1995). A survey of genetic algorithms. *Annual Reviews of Computational Physics Vol. III*, *World Scientific*, 87-117.

Whitley, D., Beveridge, J.R., Guerra-Salcedo, C., & Graves, C. (1997). Messy genetic algorithms for subset feature selection. *Proceedings of the Seventh International Conference on Genetic Algorithms (ICGA97)* (pp. 568-575).

Wold, S., Johansson, E., Jellum, E., Bjørnson, I., & Nesbakken, R. (1981). Application of SIMCA multivariate data analysis to the classification of gas chromatographic profiles of human brain tissues. *Analytica Chimica Acta 133,* 251-159.

Yuan, J.P., & Chen, F. (1999). *Food chemistry, 64*, 423-427.

Section IV

Civil Engineering

Chapter VIII

Hybrid System with Artificial Neural Networks and Evolutionary Computation in Civil Engineering

Juan R. Rabuñal, University of A Coruña, Spain

Jerónimo Puertas, University of A Coruña, Spain

Abstract

This chapter proposes an application of two techniques of artificial intelligence in a civil engineering area: the artificial neural networks (ANN) and the evolutionary computation (EC). In this chapter, it is shown how these two techniques can work together in order to solve a problem in hydrology. This problem consists on modeling the effect of rain on the runoff flow in a typical urban basin. The ultimate goal is to design a real-time alarm system for floods or subsidence warning in various types of urban basins. A case study is included as an example.

Introduction

The advances in the field of artificial intelligence keep having strong influence over the civil engineering area. New methods and algorithms are emerging that enable civil engineers to use computing in different ways. One of them is in the area of hydrology.

Hydrology is the science that studies the properties of earth's water movement in relation to land. A river basin is an area drained by rivers and tributaries. Runoff is the amount of rainfall that is carried away from this area by streams and rivers. In the study of an urban basin, the streams and rivers are replaced by a sewage system. An urban basin is influenced by the water consumption patterns of the inhabitants of a city.

The modeling of runoff flow in a typical urban basin is that part of hydrology which aims to model sewage networks. It aims to predict the risk of rain conditions for the basin and to sound an alarm to protect against flooding or subsidence. The goal of the engineers is to build a completely autonomous and self-adaptive system which makes real-time predictions of changing water levels, complemented with alarms that can alert the authorities to flooding risks.

This chapter proposes the application of two techniques of artificial intelligence in hydrology: artificial neural networks (ANNs) and evolutionary computation (EC). We show how these two techniques can work together to solve a problem, namely for modeling the effect of rain on the runoff flow in a typical urban basin. The ultimate goal of this chapter is to design a real-time alarm system for floods or subsidence warning in various types of urban basins. Results look promising and appear to offer some improvement for analysing river basin systems over some other methods such as unitary hydrographs.

Background

Description of the Problem

The problem involves measuring and predicting the rain-based flow of a typical urban basin by means of a sensor located in the city sewer. Two signals are used for modeling: The first one comes from a rain gauge which measures the quantity of rain, and the second one measures the flow level in the sewer. Both the signal corresponding to the flow and the one which corresponds to the pluviometer have been sampled at five-minute intervals.

Then, the *research considers the rainfall* runoff transformation a clear case in which one of the variables (rainfall) is transformed into another one (flowing discharge through a sewer). The transference function involves different conditions, like street slopes, irregular pavement surfaces, roof types, and so forth. These conditions make it impossible to define an equation capable of modeling the course of a drop of water from the

moment it falls to the instant in which it enters the drain network, because the course of water through the network is quite complex.

There are several methods for calculating the rainfall runoff process (Viessmann et al., 1989). Some methods are based on the use of transfer functions, usually called "unit hydrographs," and are sanctioned by experience. Other methods are based on hydraulic equations whose parameters are fixed by the morphologic characteristics of the study area (cinematic wave). Commercial packs for calculating sewage networks usually provide both "unit hydrographs" and "cinematic wave" modules.

The use of forecast methods not based on physics equations, such as ANN and genetic programming (GP), is becoming widespread in various civil and hydraulic engineering fields. The process we are dealing with is particularly apt for this kind of calculation.

Artificial Neural Networks

An artificial neural network (ANN) (Lippmann, 1987; Haykin, 1999) is an information-processing system that is based on generalizations of human cognition or neural biology and are electronic or computational models based on the neural structure of the brain. The brain basically learns from experience. There are two types of ANN, the first one with only feedforward connections is called feedforward ANN, and the second one with arbitrary connections without any direction, are often called recurrent ANN (RANN). The most common type of ANN consists of different layers, with some neurons on each of them and connected with feedforward connections and trained with the backpropagation algorithm (Johansson et al., 1992).

An ANN has a remarkable ability to derive meaning from complicated or imprecise data. The ANN can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques. A trained ANN can be considered as an "expert" in the category of information it has received to analyse. The two main advantages of the ANN are:

- **Adaptive learning:** An ability to learn how to do tasks based on the data given for training.
- **Fault tolerance:** Partial destruction of the network leads to the corresponding degradation of performance.

However, some network capabilities may be retained even with major network damage.

The ANNs have shown to be a powerful tool in many different applications, but they have a big problem: their reasoning process cannot be explained, that is, there is no clear relationship between the inputs presented to the network and the outputs it returns.

A wide range of applications have been investigated in the field of water resources management (IHE-STOWA joint research project, www.stowa-nn.ihe.nl). ANNs can be applied for prediction, simulation, identification, classification, and optimisation (Schulze & Bouma, 2000).

If significant variables are known, but not their exact relationships, an ANN is able to perform a kind of function fitting by using multiple parameters on the existing information and predict the possible relationships for the coming future. This sort of problem includes rainfall runoff prediction, water level and discharge relations, drinking water demand, flow, and sediment transport, water quality prediction, and so forth.

Some examples of ANN use are the following:

- The prediction of the water consumption of a community is not an easy task, due to the influence of many factors. Aafjes et al. (1997) investigated a short-term prediction of community water consumption by ANN, traditional expert system, and combination of ANN and expert system. Water consumption data of two years were used for the study.
- Water treatment process control is a highly non-linear process. ANN was used for controlling coagulation-flocculation-sedimentation processes and also for determining the optimal chemical dosage on the basis of water quality parameters of incoming water (Zhang & Stanley, 1999).
- Prediction of both water level and flow of a river was investigated on the specific case of the Nile River in Egypt (Atiya et al., 1999). The estimation of the river flow can have a significant economic impact mainly regarding agricultural water management, protection from water shortages, and possible flood damage.

Examples of studies in rainfall runoff prediction are the following:

- Loke et al. (1997) studied the application of an ANN for prediction of runoff coefficient by the use of simple catchment data. The input data for the ANN consists on conventional catchment characteristics such as catchment size, or percentage of impervious area, which can be easily derived from normal topographic maps. By the measurement of two rain gauges in Copenhagen, the measurement of the third one was restored. The result was compared with the simple substitution method and the satisfactory result obtained illustrated the ANN ability to deal with this type of problem.
- Sinak et al. (1998) used radial basis function (RBF) network and cascade correlation (CC) networks for predicting the sewer flow on the basis of historical rainfall data. Data for sewer flows were continuously measured by ultrasonic level sensors at three cross-section points in the sewer system. Rainfall data was measured from the gauge in the town centre. CC neural network is a special type of error backpropagation ANN that performed better than RBF network for prediction of sewer flow ahead. The advantage of CC neural network is that it optimises the topology by itself.

Genetic Programming

Genetic programming (GP) (Koza, 1992) is an evolutionary method that creates computer programs that represent approximate or exact solutions to a problem. This technique allows the finding of programs with the shape of a tree, and in their most common application those programs will be mathematical expressions that combine mathematical operators, input variables, constants, decision rules, and relational operators.

All these operators must be specified before starting the search, and so with them GP must be able to build trees in order to find the desired expression that might model the relation between the input variables and the desired output. This set of operators are divided into two groups: terminal set, with the ones which cannot accept parameters, like variables or constants; and function set, with the ones which need parameters, like add or subtract operators. Once the sets have been specified, it would be possible to establish types: Each node will have a type, and the construction of child expressions needs to follow the rules of the nodal type (Montana, 1995).

GP makes a process of automatic program generation based on Darwin's Theory of Evolution (Darwin, 1859), in which, after subsequent generations, new trees (individuals) are produced from old ones by means of crossover, copy, and mutation (Fuchs, 1998; Luke & Spector, 1998), as it happens in natural selection: the best trees will have more chances of being chosen to become part of the next generation. Thus, a stochastic process is established in which, after successive generations, obtains a well-adapted tree.

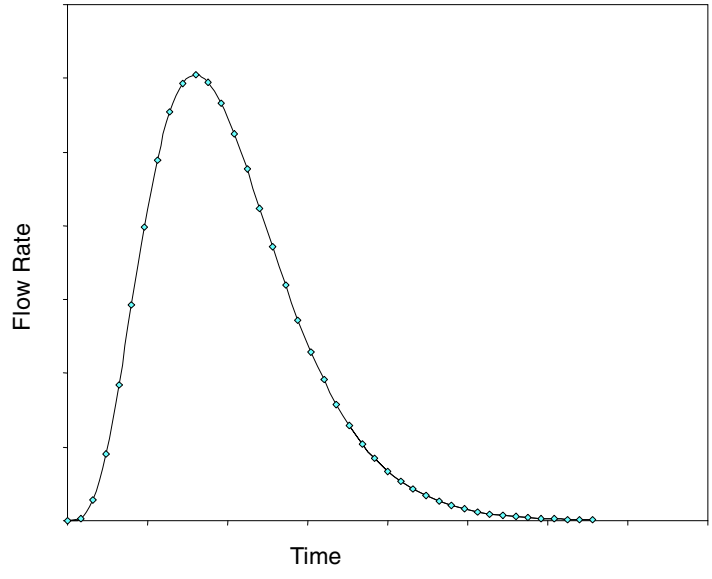
Hydraulic Methods

Focusing on the process of rainwater runoff transformation, there are different calculation philosophies, two of the most commonly applied of which will be discussed here: unit hydrographs and hydraulic methods.

Unit Hydrographs

A unit hydrograph is the runoff hydrograph from one millimetre of rainfall excess falling at a constant rate during a D duration and distributed uniformly over the watershed. The method based on unit hydrographs has been more commonly used in rural catchments than in urban ones. Wisler and Brater (1959) defined unit storm as "the hydrograph of surface runoff resulting from a relatively short, intense rain, called a unit storm." References to this method are commonly found in the bibliography, and the commercial calculation models generally include a calculation package based on this philosophy. The unit hydrograph has a limited number of parameters (between 1 and not more than 3 or 4) which encompasses the hydraulic and hydrological information of the catchment. The purely geometric variables, such as the surface, are generally accepted as unques-

Figure 1. Flow rate from one millimetre of rainfall in a unit hydrograph



tionable, whereas the remaining parameters must be calculated and estimated. Anyone who has worked with these parameters is aware of the degree of uncertainty that arises once they are set.

Perhaps the intention of giving a physical sense to the parameters is not in the nature of unit hydrographs, although, in order to evaluate them, it is indeed necessary to appeal to their relationship to geometric and hydraulic variables. It is not uncommon for a parameter of a unit hydrograph to be associated with a complicated list of physical variables, obtained through a series of experiments, which are merely an adjustment or a resolution of what is known as "back analysis" in optimisation; refer to the calculation models implemented in Hydroworks model (www.eworksllc.com/hydroworks.asp) for the K time constant of the Nash UH, or the methods included in Viessman et al. (1989). The synthesized parameter does not have a concrete physical sense, and the relationship obtained, which does not come from a dimensional analysis, is nothing more than the best fit to a problem that does not appear to have a sole solution applicable to any catchment, due to the number of variables at stake (Vaes, 1999).

Hydraulic Equations

There also are models based on the more or less reliable reproduction of the hydraulic processes that really occur in the catchment. These are the most commonly used models and are based on the Saint-Vennant equations or on some of their simplified versions,

notably the application of cinematic wave equations or similar formulations, such as those used in the RUNOFF module of the SWMM model (Huber, 1992), which is, without a doubt, the most widely used on an international level.

In these types of models, the catchment is idealized in a series of "channels" that represent the roofs and street sections connected to each other, each one having its own hydraulic characteristics (initial depression storage, friction coefficient, etc.), with a known physical sense.

Modeling with the equations of Saint-Vennant for the calculation of flows generated by a rainfall event requires a good knowledge of the drainage network, as well as the characteristics of the flow inlet structures and urban topography. If the catchment is structured with those that have homogeneous hydraulic characteristics, the discretization process would lead strictly to the definition of an infinite number of small elements whose characteristics must be defined individually.

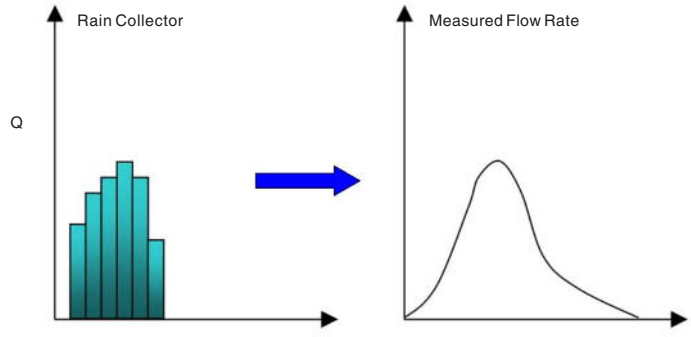
Models Based on Real Data

A reliable method used as far as the limits of means and time permit is the comparison with real data. This is obviously not a valid tool for design development, since the comparison is carried out in the operational stage, but it is useful in the analysis of how a network works, or to study its possible connection to other structures (detention tanks, flood tanks, first flush tanks) downstream.

The comparison is based on equipping the catchment with one or more rain meters and one or several flow meters. If only one is available, it should be placed at the lowest point of the network, where all of the waters intersect. The hydrograph recorded by the flow meter may override the one presented by the numerical model, after the rainfall record has been provided. By comparing the two of them, we will be able to determine whether or not the model reproduces the real situation in a sufficiently reliable manner, and the degree to which some parameters have been over or underestimated. Thus, if the recorded peaks and measurements do not coincide in time, we would have to assume that those parameters that entail delay have not been accurately estimated. If the final and initial volume does not coincide, the problem would lie in the parameters that deal with infiltration and detention as well, and so forth.

With the data of a rainfall event (precipitation and flow), it is quite simple to adjust the parameters so that the measured and calculated hydrographs have a similar appearance. However, this does not guarantee that these corrected parameters will offer an accurate result in another event. If the same exercise is carried out on a succession of rainfalls, it would be logical to assume that the result would be a final set of averaged parameters with forecasting ability, even though the fit for each specific rainfall would not be as good as the fit that would be obtained if it was only applied to this event.

Figure 2. Rain collector sensor and flow rate sensor



Modeling

The problem is predicting the flow of a typical urban basin from a sensor: the rain collected by a sensor located in the city sewer. And the goal is to predict the flow rate resulting from the rain. Usually, these two signals (rainfall and flow due to rain) have a relationship which is shown in another signal that turns the pluviometer signal into that of the flow (Figure 2). This signal has the shape of decreasing exponential functions, acting as a response to the stimulus of a system which relates both signals.

The flow level signal has two distinct causes, so it can be decomposed into two signals, one representing the daily pattern of water use by inhabitants of the city, and the other which is due to rainfall (Figure 3). Therefore, the system can be divided into two parts:

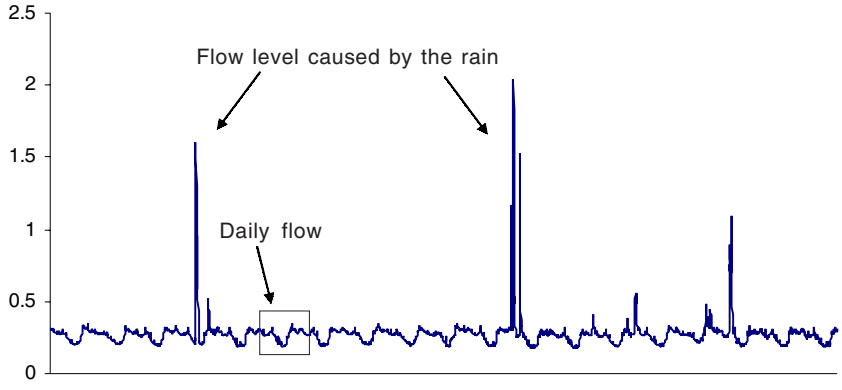
- The modeling of the daily flow rate.
- The prediction of the flow level caused by the rain.

The oscillatory behaviour of the periodic flow level signal (daily flow) depends on the quantity of water consumed by the citizens. For instance, the volume is lower at night and increases during the day with certain peak hours. The runoff caused by sporadic rainfall is added. Normally, both flows are studied separately since their origins differ, and this is the methodology used in the present study.

Modeling of the Daily Flow Rate by ANN

In the field of hydrology, despite the success in results ANNs have obtained (Govindaraju & Rao, 2000), practitioners prefer to use proven technologies rather than new models with

Figure 3. Flow rate signal with the daily flow and the flow level caused by the rain



a short hydrological history. The black box nature of ANNs has also contributed to the reluctance in using these tools despite the existence of techniques that can aid in model interpretation (Rabuñal, 2004).

We can obtain a representative signal of the mean flow in a day by taking only those samples which were not affected by the rain signal, and finding the average value of the samples corresponding to each moment in the day. Given that the samples were taken every 5 minutes, there are 288 samples per day. Therefore, this signal will have that length, as can be seen in Figure 4, where the central plain corresponds to the night hours. Peaks corresponding to the morning shower (about 10:00), lunch (15:00), and dinner (about 22:00) also can be seen. The first point corresponds to 15.00 (lunchtime). The measurement unit of the rain collected by the pluviometer is millimetres (mm) or litres per square metre (l/m^2).

Now, we can extract this signal from the whole runoff signal, so that the resulting signal will only be due to rain. As the signal extracted is an average value of the daily flow signal, there is a small error in the resulting signal. A second type of error is associated with samples affected by rain because flow due to rain and flow due to human activity are not independent. Weather (and, in this case, rain) affects the behaviour of the citizens (for example, people tend to stay at home), and so we cannot extract the real value of the flow due to human activity from the whole flow in the samples affected by rain. We can only extract the daily flow in normal conditions, and therefore there will be a small error. These errors will be corrected with the use of an ANN.

We use a feedforward ANN with a temporal window of length L as input and with a single output: The following sample is of the window, as shown in Figure 5.

Most of the ANN's approaches to the prediction problem use a multilayer network trained with the backpropagation algorithm (Atiya et al., 1999). Consider a time series X_1, X_2, X_n, where it is required to forecast the value of X_{n+1}. The inputs to the ANN are typically chosen as the previous values and the output will be the prediction. Inputs and outputs

Figure 4. Average daily flow

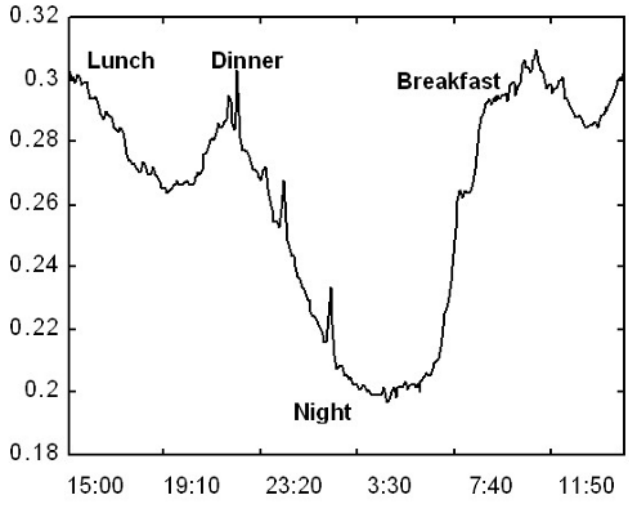
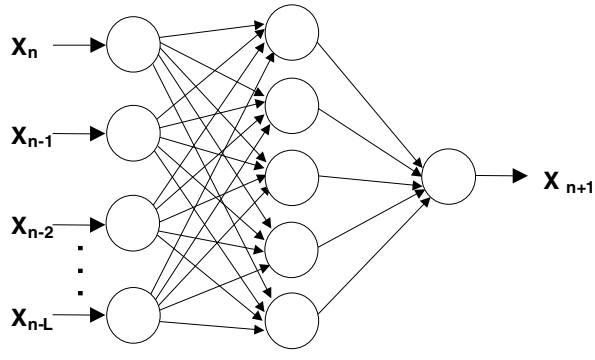


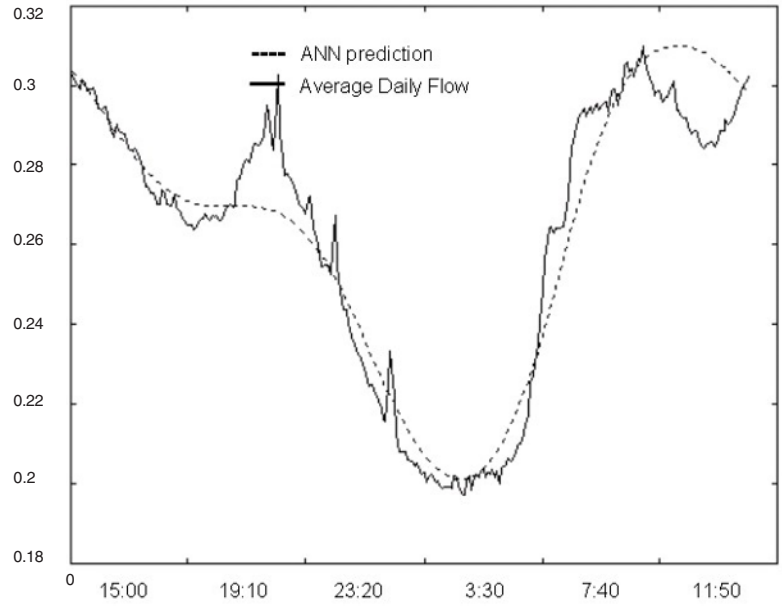
Figure 5. Architecture of the ANN



are pre-processed, this means extracting features from the inputs and transforming the target outputs in a way that might make it easier for the network to extract useful information from the inputs and associate it with the required outputs. The parameters used for the ANN and this creation are described in detail in Dorado et al. (2003).

Without considering the effect of rain, the daily flow can be modeled with an ANN taking input values between 0 and 287 (a whole day), and in comparison to the average daily flow, as observed in Figure 6.

Figure 6. Average daily flow and result of the ANN prediction



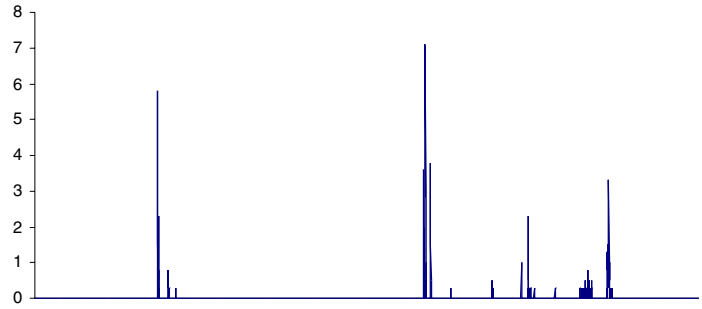
It must be noted that no analytical expressions exist that predict the shape of the daily curve, since it depends on a variety of factors linked to the usage of water — the numerous factors unmeasured which can drive the behaviour of this signal: industrial or commercial activity in the area, permanent or weekend residence, hygiene or alimentary habits, and so forth. Generally speaking, there is a curve formed by a high level (day) and a low one (night). In this way, the approach obtained fulfills the usual calculation standards. It is better to temper the adjustment of the average flow in order to take an average point, due to possible fluctuations in the flow variation depending on external circumstances, that is, to average out the influence of the many unmeasured drivers.

Modeling the Rain Flow

Once the daily flow rate effect has been modeled, a pre-processing of the original signal takes place in order to eliminate this daily component by using the modeled signal. This is done by means of placing the obtained signal upon the real one and calculating the difference between them. Once this is done, the resulting signal will only be due to the rain level which has fallen.

Now the goal is to predict the future flow rate resulting from the rain. Usually, these signals have a relationship which is shown in another signal that turns the pluviometer

Figure 7. Pluviometer signal (input training file – 5426 data points)



signal into that of the flow. This signal has the shape of decreasing exponential functions, acting as a response to the stimulus of a system that relates both signals.

There is a different approach: We are trying to obtain a system which relates the rain level (Figures 7 and 8) to the flow rate (Figures 9 and 10); now, the goal is to find a signal which models the desired system according to the relationship between both signals.

Usually, in the hydrology area, these two signals (rainfall and flow due to rain) have a relationship that can be related with a transfer function, which is called unit hydrograph. This unit hydrograph, which acts as a response to the stimulus provided by the rainfall signal, can be expressed in terms of some mathematical expressions, but the most common ones are the decreasing exponential functions, which can consider recession processes.

Four different methods are going to be used in order to determine which of them has the best performance for this task. The methods are the following ones:

- linear regression function
- recurrent ANN
- genetic programming
- SCS unit hydrograph

Modeling the Rain Flow with a Linear Regression Function

A first basic approach to prediction involves using the lineal regression method. This implies the use of a function as follows:

$$f(x) = m*x + b$$

In such a way, the behaviour of both signals can be analysed with this type of prediction. The most suitable function for flow prediction is shown here:

$$f(x) = 4.3912*x + 0.1595$$

This function produces a medium square error (MSE) of 0.0168 and a medium error (ME) of 0.077 over the test file. In Figure 11, we can see the comparative between the real flow rate and the regression function prediction over the test file.

Modeling the Rain Flow with a RANN

The RANN architecture that produces the best results is the following:

- 3 neurons: 1 input, 1 output, and 1 hidden
- Activation function: lineal (f = K x NET)

Figure 8. Graph a) is the real flow rate signal (daily flow and the flow level caused by the rain); graph b) is the daily flow; graph c) is the flow level caused by the rain (output training file – 5426 data points).

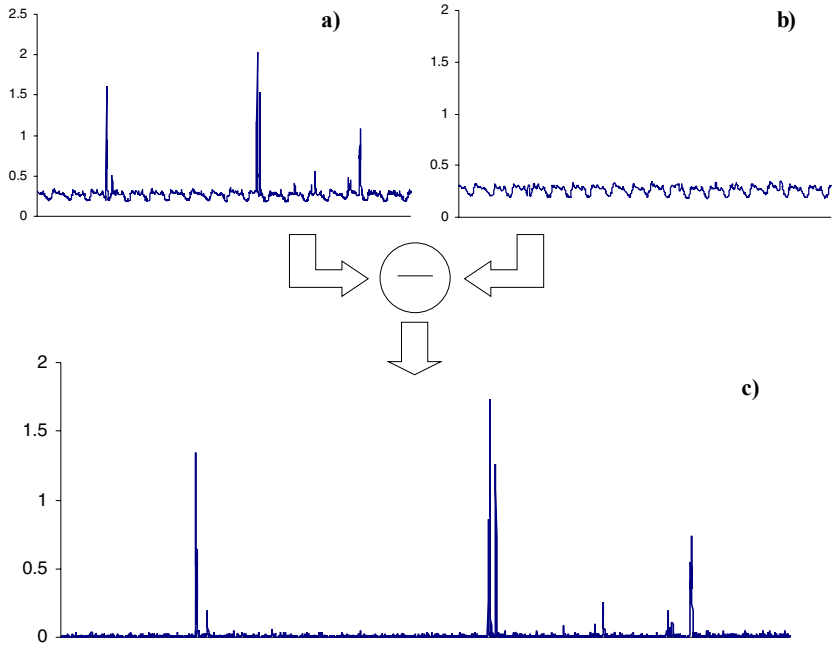


Figure 9. Pluviometer signal (input test file – 250 data points)

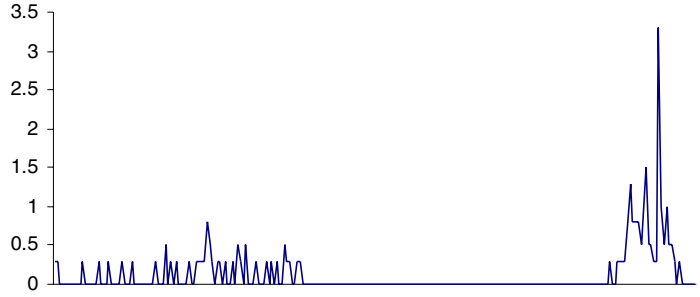
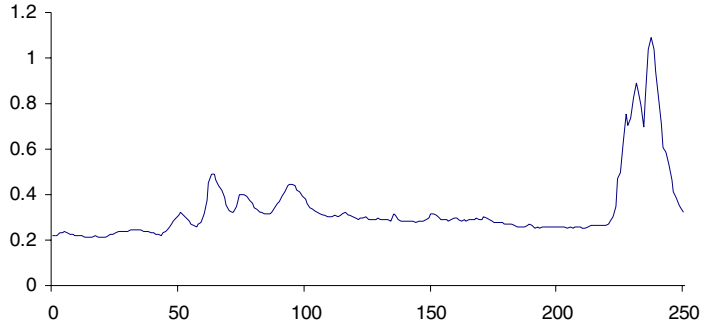


Figure 10. Flow rate signal (output test file – 250 data points)



This RANN produces a MSE of 0.0037 and a ME of 0.0343 over the test file. In Figure 13, we can see the comparative between the real flow rate and the RANN prediction over the test file.

Modeling the Rain Flow with Genetic Programming

The main goal is to achieve a signal that might model the system that has been defined by the relationship between both signals, as shown in Figure 14.

This expression models a filter between signals, which means a transformation of an input signal (the rain level) into an output signal (the flow) (Dorado et al., 2002, 2003). This transformation is characterised by a signal h(n) which also has this form:

$$\sum_i c_i \cdot e^{-k_i \cdot n}$$

Figure 11. Flow rate signal and result of the regression function prediction (test file)

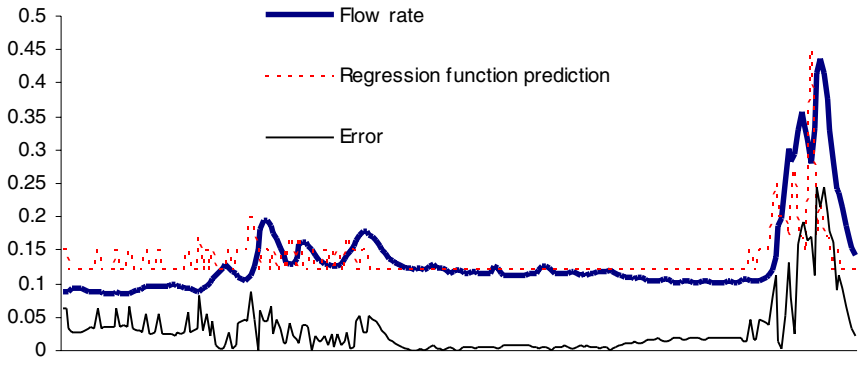
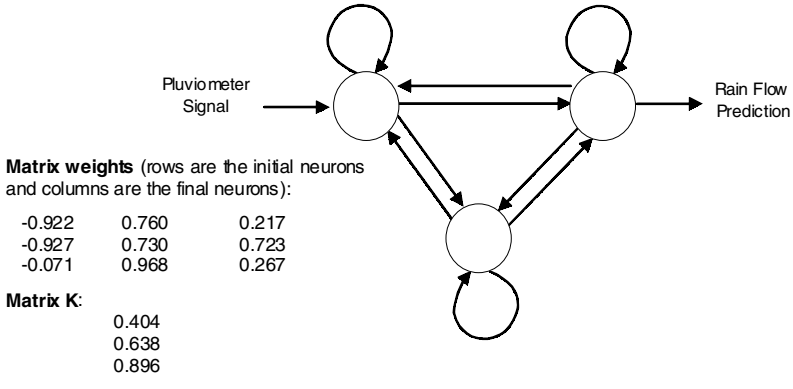


Figure 12. Architecture of the RANN



This will be a discrete system; therefore, the signal relationship is shown at the equation with differences as follows:

$$\sum_{i=0}^{M} b_i \cdot y(n-i) = \sum_{i=0}^{N} a_i \cdot x(n-i)$$

Where x(n) is the input signal, y(n) the output signal, and finally, b_i and a_i are the constants that relate to the samples that have been displaced from the input and the output. M represents the number of samples from the past of the generated input that

Figure 13. Flow rate signal and result of the RANN prediction (test file)

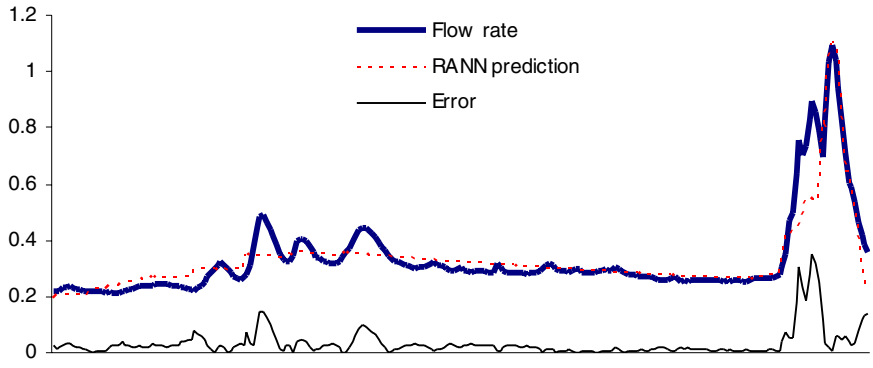
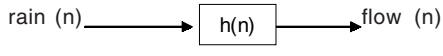


Figure 14. Relationship between both signals that are going to be modeled



are used, and N is the number of samples from the past of the input that are used for output generation.

The following configuration is used in order to generate the set of poles and zeros:

- Set of non-terminal elements: { Filter, odd, Pole, Zero, +, *, -, % }
- Set of terminal elements: { [-1,1] }

The tree that is generated by GP and that produces the best results is shown in Figure 15.

If this tree is converted into a mathematical expression, the following expression would be obtained

$$y(n) = 0.0347 \cdot x(n-1) + 0.0386 \cdot x(n-2) + 0.0018 \cdot x(n-3) + 0.0005 \cdot x(n-4)$$
$$+1.3507 \cdot y(n-1) - 0.8578 \cdot y(n-2) + 0.4293 \cdot y(n-3) - 0.1109 \cdot y(n-4)$$

Figure 15. Result of the GP method

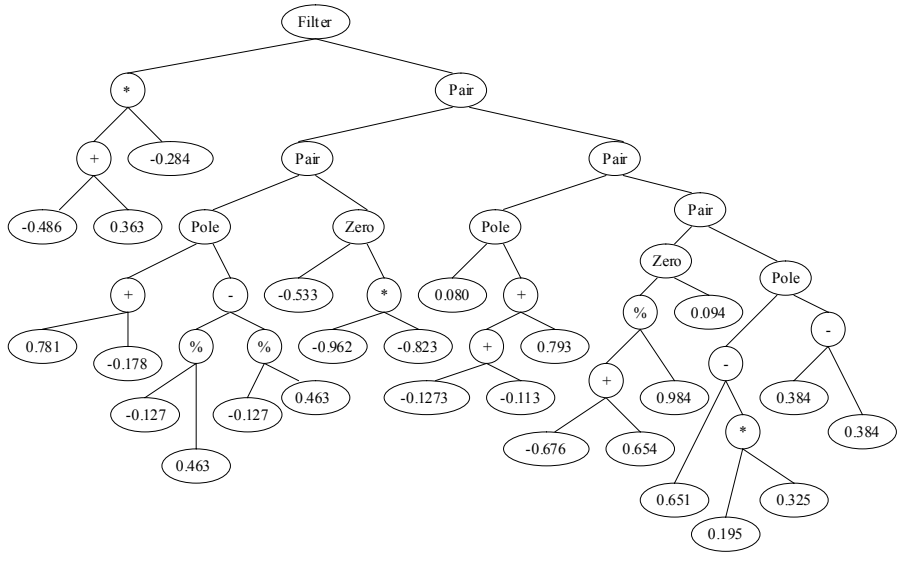
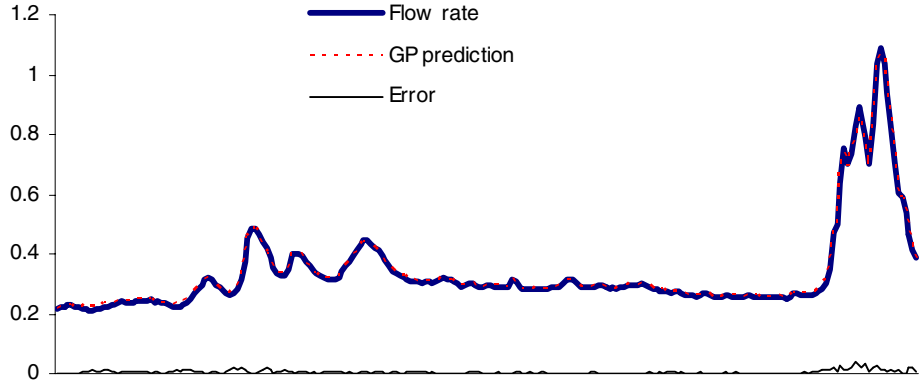


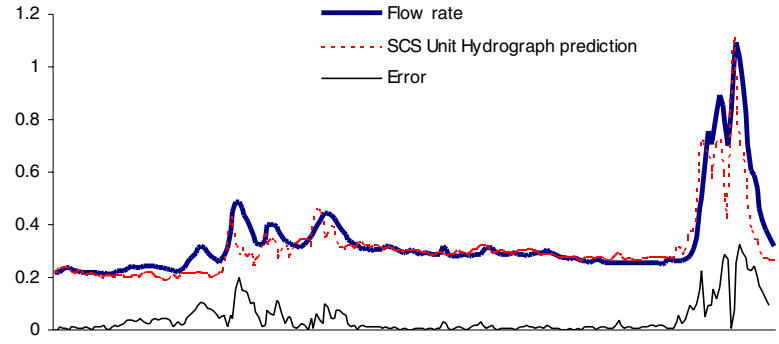
Figure 16. Flow rate signal and result of the GP expression prediction (test file)



This method and its results can be consulted about in the paper "Prediction and Modeling of the Rainfall-Runoff Transformation of a Typical Urban Basin Using ANN and GP" (Rabuñal et al., 2003).

This expression produces a MSE of 0.0001 and a ME of 0.0052 over the test file. In Figure 16, we can see the comparative between the real flow rate and the GP expression prediction over the test file.

Figure 17. Flow rate signal and result of the SCS unit hydrograph prediction (test file)



Modeling the Rain Flow with SCS Unit Hydrograph

The experts on hydrology would apply methods such as SCS unit hydrograph for this type of urban basin. In this way, the analysis of the answer would be done by means of the unitary impulse data, which means the response provided by the system to 1 mm of water drop. Other models based on hydraulic equations might be used as well, but they are more complex and their cost regarding the time of work is quite different to that proposed here.

The SCS unit hydrograph produces a MSE of 0.0056 and a ME of 0.0428 over the test file. Figure 17 shows the comparative between the real flow rate and the SCS unit hydrograph prediction over the test file.

Responses of Unit Hydrograph by the Methods

The response to the unitary impulse that is produced by the different methods can be observed in Figure 18.

Results

Table 1 shows a comparison among the resulting signals, appearing to show that the evolutionary method (GP) yields better results than the RANN method and the traditional method (SCS).

Figure 18. Unit hydrograph of the methods: (a) regression function, (b) RANN, (c) GP

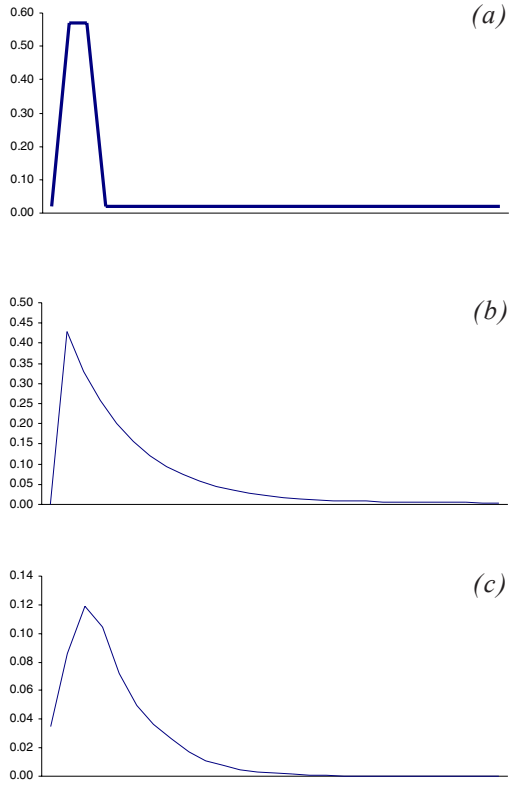
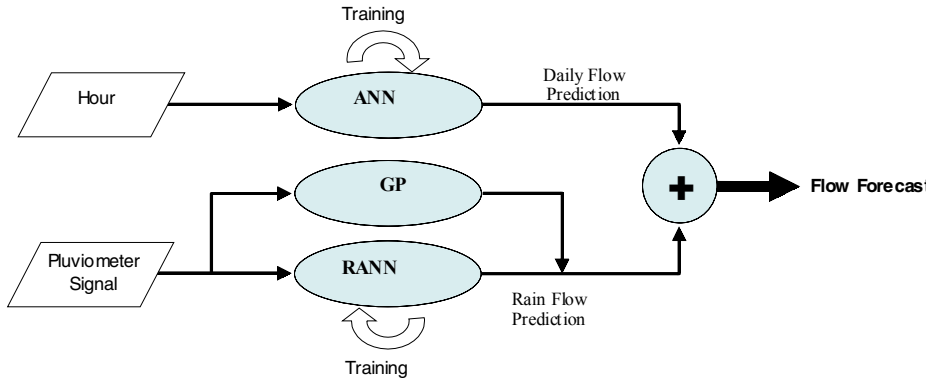


Table 1. Comparison between the basic method (regression), the artificial intelligence methods (RANN and GP), and the unitary hydrograph method (SCS)

	Regression	RANN	GP	SCS
Medium Error	0.0776	0.0343	0.0052	0.0428
Medium Square Error	0.0169	0.0037	0.0001	0.0056

Figure 19. Diagram of the system's global functioning



System Description

The system will admit two inputs: the measured pluviometer level and the hour at which the two different points of this signal have been measured.

The pluviometer input is contrasted with the recurrent ANN or GP expression, thus obtaining a prediction of the flow resulting from rain. To obtain the flow due only to human activity in the past, we extracted this signal from samples of the real flow. Now, with this human activity signal, we can make a prediction of the next human activity sample by the feedforward ANN. Just adding these two resulting samples (the human activity prediction with ANN and that predicted from rain with RANN or GP) we get the prediction of the whole runoff signal, as shown in Figure 19.

Future Trends

Once the modeling of a typical basin is obtained, the present technique will be applied to the modeling of other less typical urban basins, which are affected by various physical phenomena that disturb the surface permeability, the distribution of water through the subsoil, pollution, and so forth.

The technique also will be applied to rural basins, where there is no typical daily behaviour, given that there is no residual human activity. Because rural basins possess reduced residual human activity, the modeling method may be more effective at predicting runoff volumes. However, rural areas also possess surface characteristics which can be more sensitive to isolated natural phenomenon (storms, droughts, etc.), thus decreasing the effectiveness of the modeling method to accurately predict runoff volumes.

The goal is to implement an autonomous system which might perform real-time predictions, being self-adaptive and correcting its parameters dynamically as the rain measurements take place. Thus, alarm systems which are able to predict the risk of rain conditions for the basin can be obtained.

Conclusions

The results prove that the signal obtained is quite similar to the real one, so the used method not only automates the process, but also appears to be better than that commonly used in hydrology.

An expression that relates the time of day with the corresponding flow has been obtained for daily flow modeling, by means of a feedback between input and output values.

After many tests, it has been shown that the method with best results was the one in which the GP typing power was used for searching both the poles and zeros of the system.

The importance of GP is manifest, not only when trying to find relations among parameters, but also in order to obtain expressions with restrictions for the application of other techniques, such as the search for parameters in filter design.

As a final conclusion, it can be stated that the best results are obtained with the use of a hybrid system that combines feedforward ANN for daily flow prediction with GP for flow prediction according to the rainfall.

References

Aafjes, J., Hendriks, L.J.W., Verberne, A.J.P., & Vingerhoeds, R.A. (1997). Voorspellen van het Drinkwaterverbruik met een Neuraal Netwerk. *H2O, 30*(15), 484-487.

Atiya, A., El-Shoura, S., Shaheen, S., & El-Sherif, M. (1999). A comparison between neural-network forecasting techniques. Case study: River flow forecasting. *IEEE Transactions on Neural Networks*, *10,* 402-409.

Darwin C. (1864). *On the origin of species by means of natural selection or the preservation of favoured races in the struggle for life* (6th ed.) (originally published in 1859). Cambridge, UK: Cambridge University Press.

Dorado, J., Rabuñal, J., Pazos, A., Rivero, D., Santos, A., & Puertas, J. (2003). Prediction and modelling of the rainfall-runoff transformation of a typical urban basin using ANN and GP. *Journal of Applied Artificial Intelligence, 17,* 329-343.

Dorado, J., Rabuñal, J., Puertas, J., Santos, A., & Rivero, D. (2002, April 3-5). Prediction and modelling of the flow of a typical urban basin through genetic programming, *Proceedings of the 5th European Congress on Genetic Programming,* Ireland.

Fuchs, M. (1998). Crossover versus mutation: An empirical and theoretical case study. *3rd Annual Conference on Genetic Programming.* Morgan-Kauffman.

Govindaraju, R.S., & Rao, A.R. (2000). Artificial neural networks in hydrology. *Water Science and Technology Library* (vol. 36). Kluwer Academic Publishers.

Huber, W.C., & Dickinson, R.E. (1992). *Storm water management model. User's manual, Version 4*. Athens, GA: Environmental.Protection Agency.

Johansson, E.M., Dowla, F.U., & Goodman, D.M. (1992). Backpropagation learning for multi-layer feed-forward neural networks using the conjugate gradient method. *International Journal of Neural Systems, 2*(4), 291-301.

Koza, J.R. (1992). *Genetic programming. On the programming of computers by means of natural selection*. Cambridge, MA: MIT Press.

Lippmann, R.P. (1987). An introduction to computing with neural nets. *IEEE ASSP Magazine*.

Loke, E., Warnaar, E.A., Jacobsen, P., Nelen, F., & do Ceu Almeida, M. (1997). Artificial neural networks as a tool in urban storm drainage. *Water Science and Technology, 36*(8-9), 101-109.

Luke, S., & Spector, L. (1998). *A revised comparison of crossover and mutation in genetic programming*. The 3rd Annual Conference on Genetic Programming. Morgan-Kauffman.

Montana, D.J. (1995). Strongly typed genetic programming. *Evolutionary Computation, 3*(2), 199-200.

Rabuñal, J.R., Dorado, J., Pazos, A., Pereira, J., & Rivero, D. (2004). A new approach to the extraction of ANN rules and to their generalization capacity through GP. *Neural Computation,* 7(16), 1483-1523.

Schulze F., & Bouma N. (2000). Use of artificial neural networks in integrated water management. *Proceedings MTM-III - Artificial Neural Networks in Water Management* (pp. 333-342).

Sinak, P., Bundzel, M., Soka, M., Sztruhar, D., & Marsalek, J. (1998). Urban runoff prediction by neural networks. In V. Babovic & L. C. Larsen (Eds.), *Proceedings of Hydroinformatics '98 Conference*, Balkema, Rotterdam (pp. 791-796).

Vaes, G. (1999). *The influence of rainfall and model simplification on combined sewer system design*. Ph.D. thesis. Belgium: K.U. Leuven.

Viessmann, W., Lewis, G.L., & Knapp, J.W. (1989). *Introduction to hydrology*. Harper Collins.

Wisler, C.O., & Brater, E.F. (1959). *Hydrology*. New York: John Wiley & Sons.

Zhang, Q., & Stanley, S. (1997). Forecasting raw-water quality parameters for the North Saskatchewan River by neural network modeling. *Water Research, 31*(9), 2340-2350.

Chapter IX

Prediction of the Consistency of Concrete by Means of the Use of Artificial Neural Networks

Belén González, University of A Coruña, Spain

Mª Isabel Martínez, University of A Coruña, Spain

Diego Carro, University of A Coruña, Spain

Abstract

This chapter displays an example of application of the ANN in civil engineering. Concretely, it is applied to the prediction of the consistency of the fresh concrete through the results that slump test provides, a simple approach to the rheological behaviour of the mixtures. From the previously done tests, an artificial neural network trained by means of genetic algorithms adjusts to the situation, and has the variable value of the cone as an output, and as an input, diverse variables related to the composition of each type of concrete. The final discussion is based on the quality of the results and its possible application.

Introduction

Concrete is one of the construction materials that is currently most used, since it provides a high compressive strength (similar to a rock) and can to be adapted into different shapes during its production. It is made of a mixture of water, cement, fine aggregate (sand, 0 to 4 mm), and coarse aggregates (6 to 12 mm and 12 to 25 mm), and, in some cases, other mineral admixtures (additions) or chemical admixtures (additives) with different functions.

Concrete Rheology and Workability

The main properties of the concrete are determined in two different stages, the first one being manufactured, in which the material is in a fresh state, and the second stage corresponding to its hard state. In the first stage, the concrete appears like a fluid material and can be adapted to any shape. This property is associated to its workability. Its measurement is necessary because the capacity of the concrete to be adapted into another shape will depend on it. A suitable workability also allows for the possibility to compact the concrete that is required to evacuate the air content that is incorporated in the processes of production and placement.

The workability (also the facility of placing, docility, or consistency) depends on the rheological properties of the fluid, which are usually associated to a Bingham model (Tattersall, 1991) and must be determined to ensure that the concrete will be able to be placed. According to this theory, the fresh concrete can be described by two fundamental parameters: its plastic viscosity (the excess of the shear stress over the yield stress divided by the shear rate) and its yield stress (a critical shear stress to a point that once the yield stress is exceeded a viscoplastic material flows like a liquid). Most of the widely used tests are unsatisfactory in that they measure only one parameter, which does not fully characterize the concrete rheology. Figure 1 shows how two concretes could have one identical parameter and a very different second parameter. These concretes may be very different in their flow behaviours. Therefore, it is important to use a test that will describe the concrete behaviour, by measuring at least both factors.

Figure 1. Bingham model for two concrete mixtures of different behaviour

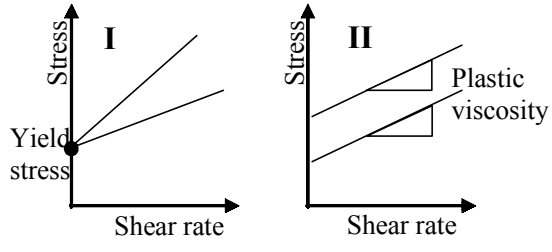


Figure 2. Testing process using the Abrams cone

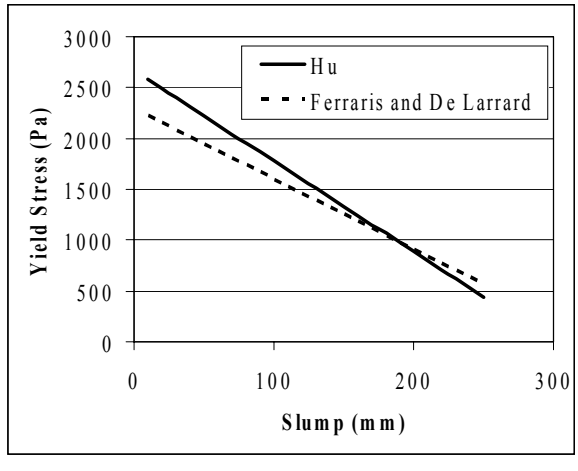
Measurement of the Workability of the Concrete

The methods to determinate the workability are usually indirect methods. The common methods are the rate of flow, the compacting factor, and the slump (being the oldest) (Abrams, 1922). This oldest method is the most used because of its simplicity. Other viscometers and rheometers have been developed to adjust to the parameters of the Bingham model, although their use in construction is very limited.

The test is very simple, inexpensive, and is used in construction to determine if a concrete can be accepted before its positioning. The test method is widely standardized throughout the world, including in ASTM C143 in the United States and EN 12350-2 in Europe (Koehler & Fowler, 2003).

The apparatus consists of a metallic mould in the form of a truncated cone whose larger base has a diameter of 20 cm, its smaller base of 10 cm, and a total height of 30 cm. The mould is stuffed with concrete in three layers of equal volume. Each layer is compacted

Figure 3. Relations studied between slump and yield stress



with 25 strokes of a tamping rod. When finalizing the filling, the cone is extracted and the final height of the mass of concrete is measured. Figure 2 shows the test process.

In this test, the stress is composed of the weight of the concrete per unit area. The slump is influenced by both yield stress and plastic viscosity; however, in most cases, the effect of plastic viscosity on the slump is negligible. The concrete will slump or move only if the yield stress is exceeded and will stop when the stress (or weight of the concrete/area) is below the yield stress.

Equations have been developed to calculate yield stress from slump, using rheometers or numerical methods to calculate it. In spite of a strong dependency on the experimental method used, the equation contributed by Hu et al. (1996) is emphasized and modified by Ferraris and De Larrard (1998). Figure 3 shows the relation which both equations provide between yield stress in Pa and slump in mm.

The Spanish code EHE (1999) classifies fresh concretes according to their workability, (the used term is consistency), and it admits a small tolerance in the results of the slump test, concluding in the following values:

- **Dry:** slump between 0 and 2 cm (test between 0 and 2 cm)
- **Plastic:** slump between 3 and 5 cm (test between 2 and 6 cm)
- **Soft:** slump between 6 and 9 cm (test between 5 and 10 cm)
- **Fluid:** slump greater than 10 cm (test greater than 8 cm)

The value of the consistency of a concrete has a great importance apart from the previously mentioned technical questions. It becomes a fundamental parameter when the concrete is received on a construction site, because of the implications that have been explained. If the concrete is within the tolerated limits (slump values), the material is accepted, and it is rejected if it is not.

Factors that Affect the Workability

One of the most important factors that affect the workability is the amount of water, in particular the water/cement ratio. This ratio must be low to ensure a high compressive strength and durability, and this low ratio is incompatible with the workability of the mixture. That is the reason why water-reducing admixtures (superplasticizers known for both their dispersing properties and because they can neutralize the electrical charges present at the surface of cement particles which reduce their tendency to flocculate) are usually included in the mixes. Many other factors, however, directly affect workability. They are:

- Type and amount of cement
- Amount, shape, and moisture of the aggregates

- Sequence of introduction of the materials in the mix machine
- Type of mix machine and time of mixture
- Time passed from the production of the mixture to the measurement of the workability
- Environmental conditions (temperature and atmospheric moisture).

It can be seen that in the fundamental variables that need to be understood the workability of the concrete are known. It also has been observed that the theoretical models are close to the real behaviour but require difficult systems that, presently, are not used in construction. Finally, there are numerous tests that identify almost every parameter that is involved in workability. So, the conditions to determine the workability of concrete seems to be suitable for use in techniques based on artificial intelligence, in particular, the artificial neural networks (ANN).

Construction of the ANN

Initial Data

The following data has been obtained from tests made in the Centre for Technological Innovation in Building and Civil Engineering (CITEEC) of the University of A Coruña (González Fonteboa, 2002). In these tests, concrete mixes are designed to maintain certain parameters constant:

Table 1. Studied variables and considered extremes

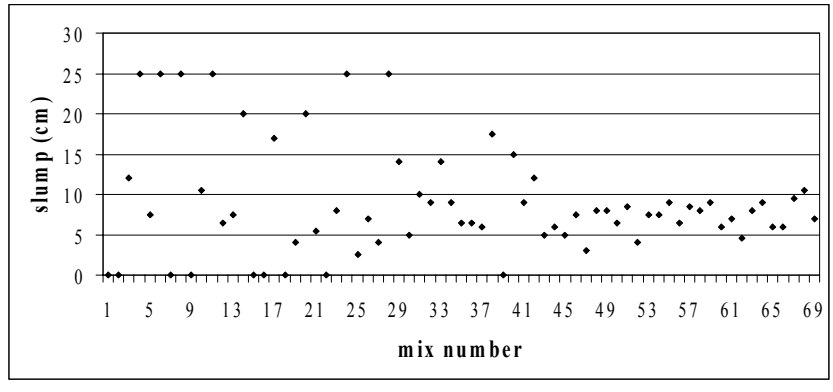
Parameter	Lower extreme		Upper extreme		Accepted domain
	Value	Mix number	Value	Mix number	
FA-0-6 (kg) (sds*)	65.24	6	105.49	57	60 to 110
CA 6 –12 (kg) (sds)	27.29	2, 3	52.44	19	20 to 60
CA 12 –25 (kg) (sds)	36.92	6	59.44	63	30 to 60
Cement (kg)	23.15	4	35.31	57	20 to 40
Cement fineness (Blaine)	3890	1 to 37	4270	38 to 69	0.2 or 0.8
HRWR (kg)	0.14	1	0.666	23	0.1 to 0.7
Water in aggregates (kg)	0.04	67	3.5	3	0 to 4
Free water (kg)	9.33	1	19.31	57	9 to 20
Superplasticizer (kg)	0.14	1	0.666	23	0.1 to 0.7
Mixer status	First mix	1, 8, 15, 17, 22, 25, 30, 33, 37, 39, 41, 49 and 51	Second mix or following	Rest	0.2 or 0.8
Mix weight (kg)	173.323	3	269.794	57	170 to 270
* sds = saturated dry surface					

- Temperature (20 ºC) and atmospheric humidity (60%)
- Water to cement ratio, fixed in all the tests at 0.55
- Fine aggregate to coarse aggregate ratio (in weight), constantly set between 0.48 y 0.51
- Aggregate to cement ratio, fixed to 6.08
- Type of materials used (with exception of cement fineness, although the type of cement remained the same in all the tests)
- Time of measurement of the workability, in all cases right after the mixture process
- Sequence and time of mixture
- Type of mix machine

The studied variables were the following:

- Fine aggregate (FA) 0 to 6 mm: of limestone
- Coarse aggregate (CA) 6 to 12 mm: of quartz, as a consequence of crushing
- Coarse aggregate (CA) 12 to 25 mm: of quartz, as a consequence of crushing
- Water content of the aggregates (because of its moisture)
- Cement: Portland CEM I – 42,5R cement
- Weight of the free water: added water during the mixture process
- Superplasticizer: a high range water reducer (HRWR) naftalenesulfonate was used
- Mix weight: total sum of the masses of the materials used in the mix
- Cement fineness: measured by the Blaine surface parameter; only two degrees of fineness were used.

Figure 4. Slump values for all mixes



- Mixer status: The mixer was used several consecutively times. After each mix, the mixer was emptied, however, residues and moisture remained for the next mix. This did not happen in the first mix. Therefore, there were only two possible statuses.

According to Table 1, 69 mixtures were completed. In this table can be observed the extreme values that were attributed to the studied variables and finally the considered domain to construct the ANN (Rodríguez, 2002) has been taken into consideration.

The results of all 69 mixes appear in Figure 4. As can be observed, the slump adopts a value between 0 cm (mix 1, 2, 7, 9, 15, 16, 18, 22, and 39) and 25 cm (mix 4, 6, 8, 11, 24, and 28).

Sixty mixes were destined for the training phase and the other 9 for the evaluation phase (mix 1, 10, 20, 30, 35, 40, 50, 60, and 69).

Prior Questions About the Architecture of the ANN

First, an evaluation of the different parameters that can vary within the neuronal network was made. This analysis is conditioned by the kind of problem we have been faced with.

- **Number of Neurons in Internal Layers:** Each neuron that presents the topology to train will be associated with a series of interconnections with the corresponding function of activation to weigh. In connection with the information previously available limits the possibilities of over-dimensioned configurations for the previous information available. On the other hand, a very elevated number of neurons must be avoided to limit the time of calculation.
- **Number of Layers:** The disposition in layers is conditioned by the same motives as the number of neurons. A network of three layers is seemingly suitable due to the correspondence between the three layers and the input-processing-output functions.
- **Activation and Output Function:** Of all the possible functions — linear, threshold, hyperbolic, tangent, and exponential — the research is limited to the continuous and positive functions (excluding the threshold function). The negative outputs are also not admitted for the same reason.
- **Recursive Architecture:** This was not employed as it did not have a physical justification in the process. The implementation of networks which recursive architecture are appropriate for problems in which one result can have an influence on the following one, which is typical in temporary series and not in this case.

Training Using Genetic Algorithms

The creation and training of the neuronal network has been developed in the software ERNAAG (Rabuñal, 1999). It uses the technology of the genetic algorithms in the training

of different types of ANN (calculating weights) as is also used in the design of architecture (number of layers, number of neurons per layer, and types of activation functions). Tests were carried out with algorithms in training based on the backpropagation from the error, but the available tests did not present a continuous distribution inside the intervals. The convergence was not always reached, and in this case, the results presented large errors.

Generic parameters (limit of the value of the weights, limit of the slope, type of activation functions, and the networks own parameters) also are codified within the ANN.

Once the individuals are designed, the following phase is to generate a random population of the individuals to apply the genetic operators to them. An individual is chosen at random as well as a value of weight or slope is chosen. A new value is generated at random, and it is incorporated into the individual. This individual is revalued and is inserted within the population depending on its new level of adaptation to the problem.

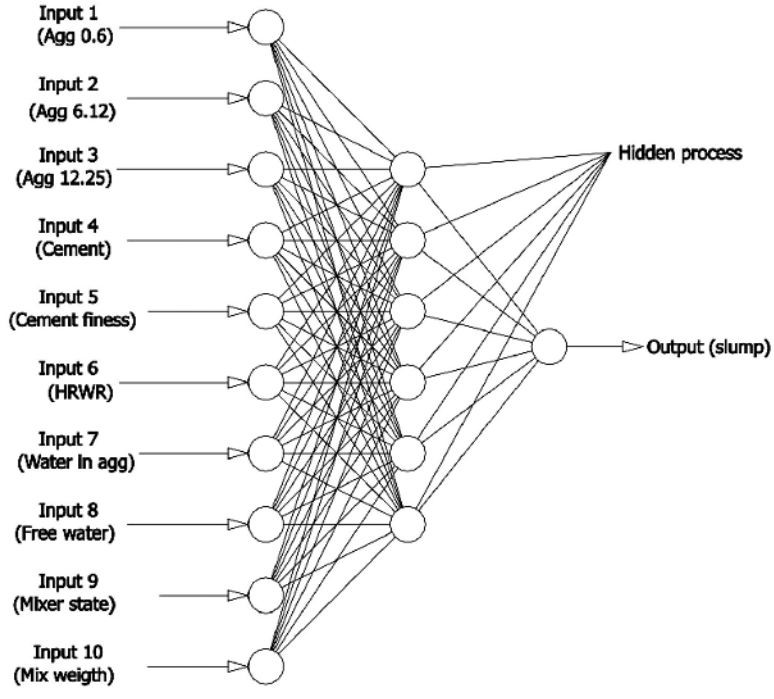
The mixing operator is applied in three phases. The first consists of choosing the individuals that are going to be mixed. In order to avoid the population becoming too homogenised, due to the continuous mixing of the best individuals, an individual of the population and another individual from the better half of the population are mixed at random. This strategy is the one that has produced the best results in the experiments of training because it not abuse using the individuals of better adaptation and takes advantage of the genetic diversity of any individual of the population. After that, individuals selected at random are mixed and the new individuals are generated. In the last phase, both new individuals are valued and only the best descendant survives in the population and the other individual with the worst adaptation in the initial population is eliminated.

Configuration of the Final ANN

Numerous tests were made to determine the ANN with the best results. The parameters explained in the previous subparagraph were varied. The final characteristics of the neural network used are summarized here:

- Input parameters divided by the weight of the mixture and normalized between 0 and 1. The non-continuous inputs 5 and 9 vary between two discrete values, 0.2 and 0.8. In diverse made tests, it was stated that these values gave better results than a simple normalization 0/1.
- Structure of three layers (17 neurons arranged in the following way: 10,6,1) in accordance with the attached scheme (Figure 5)
- Interpolation hyperbolic-tangent function
- Number of iterations: 6603269
- Quadratic error: 0.0146
- Medium error: 0.6842

Figure 5. Neural network topology



The result for this network appears in Table 2. As it is possible to be observed, the results expressed as slump (cm) adjust reasonably except in the more fluid mixes. Nevertheless, the allocation of consistency given by EHE code adjusts, in all the cases to the output of the ANN.

Analysis of the Results

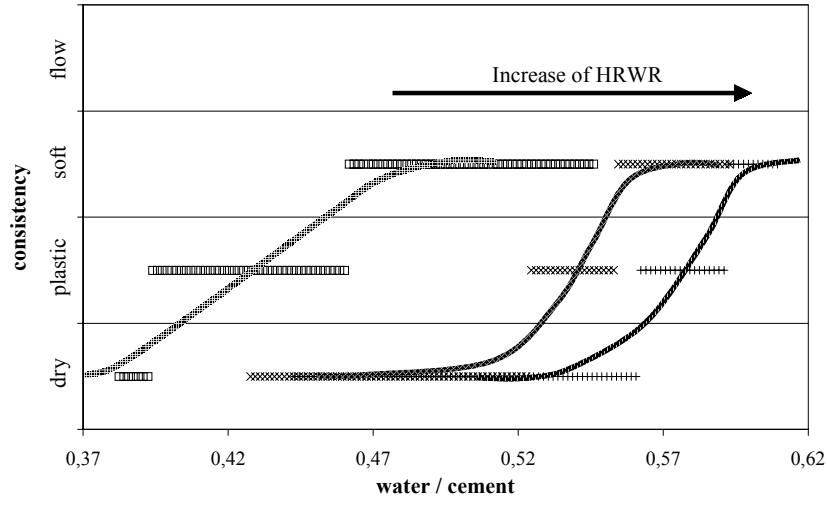
In the mixtures 20 and 40, the results have a remarkable precision. The study has been centred on a small group of mixtures and with parameters within limited ranks. It is evident that the results cannot be extrapolated to any mixture, even within the rank, since the tests come from very concrete conditions that include certain types of aggregate, type of mixer, and so forth.

In spite of this, this situation takes place frequently in batching plants, where the number of products is not elevated and they are always produced in very similar conditions. The results indicate a clear capacity to predict the consistency in a trained network with few

Table 2. Result of ANN (normalized values); slump (cm) and consistency

Mix number	Real slump			ANN output			Difference
	Norm.	0 – 25 cm	Consistency EHE	Norm.	0 – 25 cm	Consistency EHE	cm
1	0,0000	0	Dry	0,1001	2,0	Dry- Plastic	2,0
10	0,4200	10,5	Fluid	0,4237	10,6	Flow	0,1
20	0,8000	20	Fluid	0,3826	9,6	Soft-Flow	-10,4
30	0,2000	5	Plastic-Soft	0,2641	6,6	Soft	1,6
35	0,2600	6,5	Soft	0,3656	9,1	Soft-Flow	2,6
40	0,6000	15	Fluid	0,3369	8,4	Soft-Flow	-6,6
50	0,2600	6,5	Soft	0,3366	8,4	Soft-Flow	1,9
60	0,2400	6	Plastic-Soft	0,2972	7,4	Soft	1,4
69	0,2800	7	Soft	0,3148	7,9	Soft	0,9

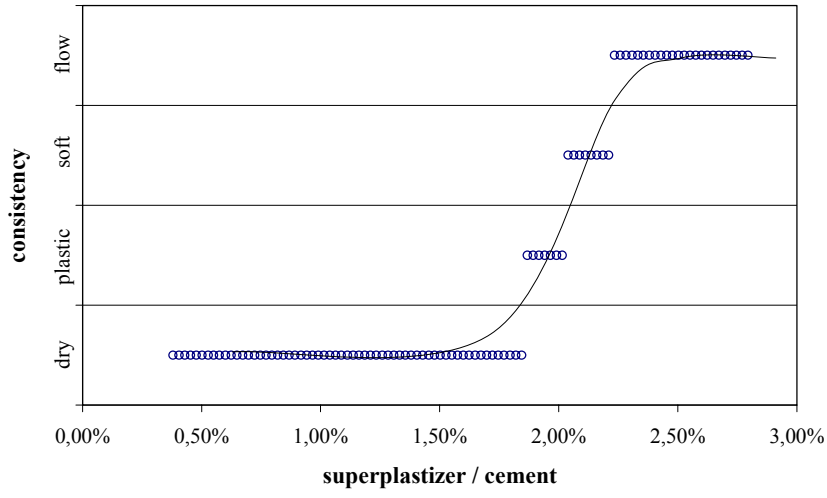
Figure 6. Relation between consistency and water/cement ratio; the "S" curves are adjusted to the results



mixes. It should be observed that the consistency is a critical parameter as it can enforce the return of the product when this product does not comply with the necessary specifications.

Thus, the trained network can contribute to interesting analysis of the dependency between certain parameters and the consistency. The most interesting includes the water/cement and the superplasticizer/cement ratios:

Figure 7. Relation between consistency and HRWR/cement ratio; the "S" curve is adjusted to the results



- In order to obtain the relation between the consistency and the water/cement ratio, the parameters "water" (input 8) and "cement" (input 4) were modified in real mixes, always within the limits for which the ANN has been trained.
- In order to obtain the relation between the consistency and superplasticizer/ cement, the parameters "HRWR" (input 6) and "cement" (input 4) were modified in real mixes, always within the limits for which the ANN has been trained.

Figure 6 shows the first analysis. The consistency variation is observed in accordance to the increase of the water in the mixture. Each one of the three curves corresponds to a dose of superplasticizer. Figure 7 gives the relation between consistency and superplasticizer /cement ratio for a standard mixture with a water/cement ratio of 0.55. It has been illustrated that additive induces the fluidification of the mixture, as well as the loss of effectiveness from elevated doses, which is demonstrated in the laboratory experiments.

Conclusions

Development of ANN models for consistency, measured by slump, in the case of conventional concrete has been described. The ANN models constructed and trained provide an efficient, quantitative, and rapid means of obtaining adequate solutions to workability in concrete mixtures.

When the training data does not completely cover the range studied, genetic algorithms are better than backpropagation as a process of training.

The generated ANN correctly predicts, in a high-quality way, the slump expected when the fundamental parameters of the mixture are varied. It can be, therefore, an adequate method for the design of mixtures within the studied rank.

The method used, with a greater number of tests and with a more extensive rank, can help to improve the analytical formulations employed until now.

Acknowledgments

The work that appears here has been possible thanks to the support of the investigation projects MAT2001-0765 (Spanish Ministry of Science and Technology) and PGIDIT02PXIC11801PN (Xunta de Galicia): "Experimental, analytic and numerical study about anchorage designs for prestressed concrete. Develop application using different concretes. (ZANCLA)." Cement was provided by CEMENTOS COSMOS. The authors also especially wish to thank the work developed by D. Ibán Rodríguez (Civil Engineer) and the collaboration of D. Juan Rabuñal (Computer Engineer, PhD, CITEEC of UDC), and of Fernando Martínez-Abella (Civil Engineer, Ph.D., Professor at UDC).

References

Abrams, D.A. (1922). Proportion concrete mixtures. *Proceedings of the American Concrete Institute* (pp. 174-181).

EHE. (1999). *Instrucción de Hormigón Estructural.* Publicaciones del Ministerio de Fomento. Secretaría General Técnica.

Ferraris, C.F., & de Larrard, F. (1998). Modified slump test to measure rheological parameters of fresh concrete. *Cement, Concrete, and Aggregates, 20*(2), 241-247.

González Fonteboa, B. (2002). *Recycled concretes with recycled concrete aggregates: Dosages, mechanical properties and shear structural behaviour.* Doctoral thesis, A Coruña, Spain.

Hu, C., de Larrard, F., Sedran, T., Bonlag, C., Bose, F., & Deflorenne, F. (1996). Validation of BTRHEOM, the new rheometer for soft-to-fluid concrete. *Materials and Structures, 29*(194), 620-631.

Koehler, E.P., & Fowler, D.W. (2003). *Summary of concrete workability test methods. Report No. ICAR 105-1.* International Center for Aggregates Research.

Rabuñal J.R. (1999). *Entrenamiento de Redes de Neuronas Artificiales mediante Algoritmos Genéticos*. Graduate thesis, Facultad de Informática, University of A Coruña, Spain.

Rodríguez, I. (2002). *Study of concrete dosages using artificial neuronal network trained with genetical operators.* A Coruña, Spain.

Tattersall, G.H. (1991). *Workability and quality-control of concrete.* London: E & FN Spon.

Section V

Financial Analysis

Chapter X

Soft Computing Approach for Bond Rating Prediction

J. Sethuraman, Indian Institute of Management, Calcutta, India

Abstract

Soft computing is popularly referred to as a collection of methodologies that work synergistically and provide flexible information processing capabilities for handling real-life situations. Its aim is to exploit the tolerance for imprecision, uncertainty and approximate reasoning in order to achieve tractability and robustness. Currently, fuzzy logic, artificial neural networks, and genetic algorithms are three main components of soft computing. In this chapter, we show the application of soft computing techniques to solve high dimensional problems. We have taken a multi-class classification problem of bond rating prediction with 45 input variables and have used soft computing techniques to solve it. Two techniques, namely dimensionality reduction technique and variable reduction techniques, have been tried and their performances are compared. Hybrid networks are found to give better results compared to normal fuzzy and ANN methods. We also have compared all the results with normal regression techniques.

Introduction

Soft computing is popularly referred to as a collection of methodologies that work synergistically and provide flexible information processing capabilities for handling real-life situations. Its aim is to exploit the tolerance for imprecision, uncertainty, and approximate reasoning in order to achieve tractability and robustness. Currently, fuzzy logic (Wang, 1997), artificial neural networks (Mehrotra, 1997; Hassoun, 1995), and genetic algorithms (Deb, 2001) are three main components of soft computing. ANN is suitable for building architectures for adaptive learning, and GA can be used for search and optimization. Fuzzy logic provides methods for dealing with imprecision and uncertainty. The analytic value of each one of these tools depends on the application.

Neural networks learn from experience, especially used in pattern recognition (Mehrotra, 1997; Hassoun, 1995). This distinguishes neural networks from traditional computing programs, which simply follow instructions in a fixed sequential order. Fuzzy inference systems (Wang, 1997) are useful for situations where human expertise (that cannot be translated into a set of equations) needs to be incorporated into a decision-making, automated process (e.g., power plant control). Evolutionary programming, evolutionary strategies, and genetic algorithms (Wang, 1997) are useful for optimization problems where their particular difference is in how they avoid local extrema (i.e., error minimization for parameter estimation).

We term high dimensional problems as those problems which have a relatively greater number of input variables.

Proposition 1: A problem with n input variables is said to be an n-dimensional problem.

In this chapter, we have used soft computing techniques to handle high dimensional problems. We have taken an example of bond rating prediction problem (Sehgal, Ramasubramanian, & Rajesh, 2001; Chaveesuk, Srivaree-ratana, & Smith, 1999). Bond rating prediction is a multi-class classification problem which makes it a really tough problem to solve.

According to Standard & Poor's (S&P), "The bond or credit rating is an opinion of the general creditworthiness of an obligor with respect to a particular debt security or other financial obligation, based on relevant risk factors."

When investors lend money to companies or governments it is often in the form of a bond, a freely tradable loan issued by the borrowing company. The buyers of the bond have to make a similar assessment on creditworthiness of the issuing company, based on its financial statement (balance sheet and income account) and on expectations of future economic development. Most buyers of bonds do not have the resources to perform this type of difficult and time-consuming research.

Fortunately, so-called rating agencies exist that specialize in assessing the creditworthiness of a company. The resulting credit or bond rating is a measure for the risk of the company not being able to pay an interest payment or redemption of its issued bond. Furthermore, the amount of interest paid on the bond is dependent on this expected

chance of default. A higher rating implies less riskful environment to invest your money and less interest is received on the bond. A lower rating implies a more riskful company and the company has to pay more interest for the same amount of money it wants to borrow. Unfortunately, not all companies have been rated yet. Rating agencies also are often slow to adjust their ratings to new important information on a company or its environment. And sometimes different rating agencies assign different ratings to the same company. In all of these situations, we would like to be able to make our own assessment of creditworthiness of the company, using the same criteria as the rating agencies. The resulting measure of creditworthiness should be comparable to the rating issued by the rating agencies.

This is difficult because the rating process is not available to the outside public. Rating agencies closely guard their rating process; they merely state that financial and qualitative factors are taken into account when assigning ratings to companies.

We would like to try to open this black box by describing the relationship between the financial statement of a company and its assigned credit rating. This might enable us to say how much of a company's rating is affected by the qualitative analysis performed by the rating agency. And the found knowledge could of course be used to classify companies that have not yet been rated or recently have substantially changed.

Credit rating may be given to cities, towns, countries, and so forth, apart from debt instruments. We consider here debt instruments only. Typically, in a market there exists a number of bonds, which are not rated. An investor who wants to invest in these bonds may wish to have some idea about the rating of the bond. This project deals with assigning a rating class to an unrated instrument.

Literature Survey

A rating agency assesses the relevant factors relating to the creditworthiness of the issuer. These include the quantitative factors like the profitability of the company and the amount of outstanding debt, but also the qualitative factors like skill of management and economic expectations for the company. The whole analysis is then condensed into a letter rating. Standard & Poor's (S&P) and Moody's both have been rating bonds for almost a century and are the leading rating agencies right now. Other reputable rating institutions are Fitch and Duff & Phelps (DCR). In India, we have CRISIL and ICRA doing the same.

Different rating agencies use different methods for rating the instruments and their ratings also differ from each other. Generally, the ratings used follow the same standards, but there may be some variations which lead to the changing ratings between the rating agencies. The various ratings that some of the rating agencies give to the bonds are shown in Table 1.

Table 1. Rating scales

Moody's	S&P	Fitch	DCR	Definitions
Aaa	AAA	AAA	AAA	Prime. Maximum Safety
Aa1	AA+	AA+	AA+	High Grade. High Quality
Aa2	AA	AA	AA	
Aa3	AA-	AA-	AA-	
A1	A+	A+	A+	Upper Medium Grade
A2	A	A	A	
A3	A-	A-	A-	
Baa1	BBB+	BBB+	BBB+	Lower Medium Grade
Baa2	BBB	BBB	BBB	
Baa3	BBB-	BBB-	BBB-	
Ba1	BB+	BB+	BB+	Non Investment Grade
Ba2	BB	BB	BB	Speculative
Ba3	BB-	BB-	BB-	
B1	B+	B+	B+	Highly Speculative
B2	B	B	B	
B3	B-	B-	B-	
Caa	CCC+	CCC	CCC	Substantial Risk
	CCC			In Poor Standing
	CCC-			
Ca				Extremely Speculative
C				May be in Default
		DDD		Default
		DD	DD	
	D	D		
			DP	

Statistical Approaches to Bond Rating

The process of bond rating as performed by an expert committee of a specific financial agency is rather covert because of confidentiality issues. Accordingly, many researchers have tried to formulate alternative approaches to predict bond ratings, such as statistical approaches and human knowledge processing approaches, with concentration on the statistical approaches. Previous studies focused on linear regression analysis and linear discriminant analysis (Everitt & Dunn, 2001; Maddala, 1997). Many researchers (Refenes, 1994) have criticized the use of conventional statistical analyses techniques, such as multiple regression models, as having limited success in predicting bond ratings because the application does not adhere to common functional forms. However, recently neural networks, fuzzy systems, and other expert systems are being applied to solve such multi-class classification problems (Chaveesuk, Srivaree-ratana, & Smith, 1999). One of the human knowledge processing approaches is an expert system, which relies on knowledge engineering techniques. It performs the classification task using a ruleset formulated by

experts. Rules are derived according to the opinions of experts on the effects of each related financial variable. However, it is very difficult to develop a rule-based expert system for rating bonds since few experts are available and most knowledge about the process of rating is confidential.

Problem Formulation

The bond rating prediction is a multi-classification type of problem since it deals with a set of input variables mapping to a set of discrete and mutually exclusive classes. Here, the different bond issues form the set of input data instances and the various bond ratings form the set of possible classes to which the input bonds can belong. Bonds must be assigned to a class and to one class only.

We have taken 15 types of bond ratings (as mentioned in the appendix) and 46 variables for predicting the bond rating. So as per *proposition 1*, it is a problem of 46 dimensions. The reason for choosing so many variables is to incorporate as much information as possible about the companies. The system should be capable of learning the dependencies in the data. Since such a problem is very tough, two techniques, namely *dimensionality reduction technique (using PCA)* and *variable reduction technique,* have been tried and their performances are compared using three different types of unsupervised learning techniques. They are:

1. Kohonen network or self-organizing map, a neural network- (Hassoun, 1995) based approach.
2. Fuzzy-means (FCM), a fuzzy-based clustering technique (Bezdek, 1981; Bezdek & Pal, 1992)
3. Fuzzy Kohonen, a neuro-fuzzy system (Tsao, Bezdek, & Pal, 1994).

Kohonen Networks

The Kohonen network (Kohonen, 1989), or "self-organizing map" (SOM), has been developed by Teuvo Kohonen. The basic idea behind the Kohonen network is to set up a structure of interconnected processing units ("neurons") which compete for the signal. The SOM defines a mapping from the input data space spanned by $x_1 \ldots x_n$ onto a one- or two-dimensional array of nodes. The mapping is performed in a way that the topological relationships in the n-dimensional input space are maintained when mapped to the SOM. In addition, the local density of data is also reflected by the map: Areas of the input data space which are represented by more data are mapped to a larger area of the SOM.

Each node of the map is defined by a vector w_{ij} whose elements are adjusted during the training. The basic training algorithm is quite simple:

1. Initialize the neuron weights to random values
2. Select an object from the training set
3. Find the node — winning unit m_c which is closest to the selected data (i.e., the distance between w_{ij} and the training data is a minimum)

$$\|x - m_c\| = \min_i \|x - m_i\|$$

4. Adjust the weight vectors of the closest node and the nodes around it in a way that the m_i move toward the training data.

$$m_i(t+1) = m_i(t) + h_{ci}[x(t) - m_i(t)]$$

where $h_{ci} = h_o \exp\left[-\frac{\|r_i - r_c\|^2}{\sigma^2}\right]$

5. $h_0 = h_0(t)$ and $\sigma = \sigma(t)$ as suitable deceasing functions of time and r_c and r_i denoting the architectural coordinates of the winning neuron and the neuron i, respectively.
6. Repeat starting with Step 1 for a fixed number of repetitions

The amount of adjustment in Step 3, as well as the range of the neighborhood, decreases during the training. This ensures that there are coarse adjustments in the first phase of the training, while fine-tuning occurs during the end of the training.

The Kohonen map reflects the inner structure of the training data. However, one cannot say which neurons are activated by which input vectors. In addition, the neurons corresponding to some input vectors after a particular training will correspond to another set of vectors after another training run. So the SOM has to be calibrated. This can be achieved by presenting well-known examples to the net and by recording which neuron is activated with a given example vector. As Kohonen maps tend to form some kind of elastic surface on the range of input vectors of the training data, neurons which are not activated in the calibration process may be interpreted by interpolation.

FCM

The fuzzy c-means algorithm (FCM) is given next, together with its advantages and disadvantages. This iterative algorithm is based on the classical isodata method of Ball and Hall (1967) .The number of clusters c to be found needs to be given beforehand, where c is greater than or equal to two and less than or equal to the number of objects K. In addition, the so-called exponent m (m > 1.0) needs to be given.

The exponent m determines the degree of fuzziness of the resulting clustering process. As m -> 1 the fuzziness of the clustering result tends to the results derived with the classical isodata method. As m -> ∝ the membership values of all the objects to each cluster tend to the reciprocal of the number of classes 1/c. The next section provides a more detailed explanation of the clustering process.

Clustering (Training)

The clustering (or training) algorithm of the fuzzy c-means algorithm reads as follows:

1. Initialize the membership values μ_{ik} of the x_k k objects to each of the i clusters for $i = 1, ..., c$ and $k = 1, ..., K$ (for example randomly) such that:

$$\sum_{i=1}^{c} \mu_{ik} = 1 \quad \forall k = 1,....K \text{ and } \mu_{ik} \in [0,1] \quad \begin{matrix} \forall i = 1,....c \\ \forall k = 1,....K \end{matrix}$$

2. Calculate the cluster centers using these membership values:

$$v_i = \frac{\sum_{k=1}^{K} \left(\mu_{ik}\right)^m x_k}{\sum_{k=1}^{K} \left(\mu_{ik}\right)^m} \quad \forall i = 1,....c$$

3. Calculate the new membership values μ_{ik}^{new} using these cluster centers:

$$\mu_{ik}^{new} = \left[\sum_{j=1}^{c} \left(\frac{\left\| v - x_i \right\|_A}{\left\| v_k - x_j \right\|_A} \right)^{\frac{2}{m-1}} \right]^{-1} \quad \begin{matrix} \forall i = 1,....c \\ \forall k = 1,....k \end{matrix}$$

4. $f \left\| \mu^{new} - \mu \right\| > \in$, $let \mu = \mu^{new}$ and go to step 2.

To calculate the vector distances in Step 3, a suitable measure of distance needs to be chosen. The process ends when the distance between two successive membership matrices falls below a stipulated convergence threshold.

Labeling

In order to be able to apply the found cluster center as classifiers, they need to be given reasonable names. The process of assigning class names to cluster centers is called labeling. A labeling method for the fuzzy c-means algorithm is to inspect the cluster centers and their respective membership values of the various features, and to assign a label manually.

The principal idea of this procedure is to present a sample of objects to the fuzzy c-means classifier whose class memberships are known (in the form of 0/1 values) and also have been calculated by the algorithm. By means of the given cluster membership information of each individual object and the membership values for each cluster center, the cluster centers can be associated with their respective classes.

The linking of the class names to the cluster centers is achieved with a scoring matrix. Given c clusters and a set of examples x_k each belonging to a cluster class (k) = {1… c} a scoring matrix P is given of dimensions cXc and whose elements are initialized as 0.0.

The cluster centers can then be labeled automatically with the following algorithm:

1. Calculate the membership function μ_{ik} of the object x_k of class (k) to all the cluster centers vi

$$\mu_{ik} = \left[\sum_{j=1}^{c} \left(\frac{\left\| v - x_i \right\|_A}{\left\| v_k - x_j \right\|_A} \right)^{\frac{2}{m-1}} \right]^{-1} \quad \forall i = 1, \ldots . c$$

2. Let $P_{Class(k),\, i} = P_{Class(k),\, i} + \mu_{ik}$, $\forall i = 1, \ldots, c$
3. Go to Step 1 until all examples x_k have been processed by Steps 1 and 2.
4. Determine $l_i = \max_{k=1,\ldots c} \{p_{k,i}\} for \; i = 1, \ldots, c$
5. Assign the label l_i to the cluster center v_i.

The procedure fails if a label is assigned to two or more classes, meaning that a cluster center was not found for all class labels. In rare cases, it also can happen that the maximum of the scoring matrix of Step 4 is not established beyond doubt. In either of these two cases, the algorithm is interrupted with an error report.

Fuzzy Kohonen Clustering Network (FKCN)

A fuzzy Kohonen clustering network (FKCN) (Tsao et al., 1994) is a neuro-fuzzy model in which fuzzy sets and artificial neural networks, rather than being stand-alone models, exist side-by-side in one complete model. Specifically, a self-organizing Kohonen map (Kohonen, 1989) is combined with the fuzzy c-means algorithm (Bezdek, 1981).

Training Algorithm

When training a fuzzy Kohonen network, the learning rate of the Kohonen network is essentially determined dynamically by means of a fuzzy c-means step. An adaptation of the neuron neighborhood does not take place. It is essentially different to the training algorithm of a Kohonen network in that a learning step always considers all of the training examples together. The fuzzy Kohonen network is also trained using a cumulative strategy, while the Kohonen network uses single step learning.

The training process is controlled by two parameters: the exponent m_o and the exponent step Δm. The exponent m_o corresponds to the exponent of the fuzzy c-means algorithm (see Section Fuzzy C-Means Algorithm), the exponent Δm step controls the rate of decrease of the adjustment of the neurons' weights. To continue the analogue with fuzzy c-means training a convergence threshold $\varepsilon > 0$ is given.

The training algorithm of the fuzzy Kohonen network is as follows:

1. The elements w_{ij} of the weights vector w_i are initialized using random numbers.
2. Calculate for each input vector x_k the membership $u_{ik}(t)$ to the individual neurons:

$$u_{ik}(t) = \sum_{j=1}^{c} \left(\frac{\left\| x_k - w_i(t) \right\|}{\left\| x_k - w_j(t) \right\|} \right)^{\frac{-2}{m(t)-1}} \quad \forall i = 1,....c \quad , \quad \forall k = 1,....K$$

Calculate the learning rate $\alpha_{ik}(t)$ using these membership values:

$$\alpha_{ik}(t) = (u_{ik}(t))^{m(t)}$$

3. Adjust the weight vectors w_i such that:

$$\forall i = 1,....c$$

4. Let, $m(t+1) = m(t) - \Delta m$

If $m(t+1) > 1.0$ and $\|w(t+1) - w(t)\| > \in$ then go to Step 2.

Where K is the number of training examples and c the number of neurons in the network. This method of training retains the self-organizing properties of the Kohonen feature map, but rather than using an explicit neighborhood for the adjustment of the weights, an individual learning rate for each neuron is calculated using the current exponent. In this way, the training focus is on the winning neuron. The decrease in the exponent rate (subtracting the exponent step after each training) results in $m(t) \rightarrow 1$, that is, the learning rate is more concentrated on the winning neuron. In the extreme case of $m(t) = 1$ only the winning neuron would be adapted.

Since, however, the neighborhoods of the neurons on the map are not considered explicitly, the visual properties of the Kohonen network are lost when using this procedure. As the adjustment of the neuron weights is achieved with regard to all of the feature vectors, the process is independent of the presentation order of the training examples (as opposed to Kohonen networks).

In this way, by integrating the fuzzy c-means algorithm into the network, the weaknesses of the Kohonen network mentioned are avoided. Moreover, it turns out that the convergence properties of the network are improved upon, far less training cycles being needed to complete the task (Bezdek & Pal, 1992). This, in itself, is an advantage over the Kohonen network, saving much time in the model building stage.

In the case of the exponent step being chosen and the number of neurons on the Kohonen map being the same as the number of classes, then the algorithm is equivalent to the fuzzy c-means algorithm. If, however, there are more neurons than expected classes, this has the advantage over the fuzzy c-means algorithm in that more than one neuron can represent a class, so more complex class boundaries can be built. The hyper-spherical class boundaries of the individual neurons produce the resultant class boundary. In this case, one no longer speaks of class centers but rather class representations.

Solution Methodology

Input Selection

The inputs were chosen from a wide range of variables including a firm's profitability ratios, turnover ratios, age, and so forth. These metrics describe the organization's size, profitability, leverage, liquidity, and obligations. The rationale behind choosing so many variables is to include as much information about the firm as possible, so that the system would learn the dependencies in the data. However, it has been found out that qualitative factors, such as management performance and investor confidence, and non-financial factors, like market risk, management risk, and so forth, are greatly influential in the rating of Bond/Company. However, since non-financial data are not easily available and qualitative factors become more subjective, we use only financial data.

The input consisted of 45 financial ratios (Manuj, 1993). Apart from financial ratios, other known data like age of the company also was used. The list of input parameters considered are shown in the Table 2.

Table 2. Input variables

Pbdit(nnrt) / sales	avg. debtors / avg. current assets
Profit before tax(nnrt) / sales	avg. working capital / total assets
Pbdit(nnrt) / avg capital employed	avg. working capital / sales
Pbit(nnrt) / interest	long-term borrowings / net worth
Profit after tax(nnrt) / total asset	total borrowings / net worth
Sales / total asset	total borrowings / total assets
Sales / current assets	(operating profit – interest) / total assets
Sales / capital employed	total income / net worth
net worth / total liabilities	operating profit / total assets
Reserves / total assets	operating profit / capital employed
Avg. days creditors	operating profit / sales
Avg. days debtors	cash profits / sales
Avg. days debtors / avg. days creditors	cash profits / total borrowings
Avg. inventory / avg. current assets	cash profits / capital employed
Avg. inventory / sales	avg. cash bank / avg current assets
Avg. inventory / avg. working capital	avg. cash bank / quick assets
Avg. inventory / avg. total assets	current assets / sales
Avg. inventory / avg. daily cost of production	current assets / total assets
Total assets	current assets / current liabilities
Sales	quick assets / total assets
Age	quick assets / current liabilities
(Advertising+Distribution+Marketing)/Sales	(cash profits/(current portion of long term liabilities +interest)
(Total raw material expense + energy + indirect taxes + wages + other operating expenses)/Sales	

Note: pbdit (nnrt) denotes profit before dividend; interest and tax, after extraordinary income and expenditure, have been removed from the calculations

Pre-Processing

The input variables chosen were not on the same scale. Therefore, a normalization procedure was desirable. Here, each input variable was normalized to standard normal form.

Dimensionality Reduction

Principal component analysis (Everitt & Dunn, 2001; Maddala, 1997) could be used for the reducing the dimensionality. It describes the variation of a set of multivariate data in terms of a set of uncorrelated variables, which are linear combinations of the original variables.

Performance Metrics

For a multi-class classification problem like bond rating prediction, several methods for the measurement of accuracy and performance can be applied. We have chosen to calculate the number of records (or data which have been classified correctly) and the number of samples misclassified in either direction (Sehgal et al., 2001). A higher rating bond misclassified to be that of a lower rating bond is of lesser significance than the lower rating bond being classified as a higher rating.

Experimental Set Up

Data Sources

The data taken for the prediction of rating is a CRISIL 1999 data (Rating Scan, 1999). CRISIL is popular bond rating agency in India. Typically, the instruments rated by rating agencies include:

- Long-term debt instruments (e.g., bonds, debentures)
- Medium term debt instruments (e.g., fixed deposits)
- Short-term debt instruments (e.g., commercial paper)
- Structured obligations
- Preference shares

Credit ratings of long-term debt instruments of manufacturing companies have been used for experiments. Data about the companies were collected from prowess database, maintained by the Centre for Monitoring Indian Economy (CMIE).

Datasets Used

- **Training Dataset:** The training dataset consisted of 170 records. (All classes were included).
- **Testing Dataset:** The testing dataset consisted of 33 records. (All classes were included).

The rating scales also have also been reduced because of the complexity of problems since there were originally 15 classifications. The full information on rating scales can be found in the appendix. They were finally reduced to six dimensions by clubbing the following ratings into one group:

AAA and AA+ – Highest Safety Bonds

AA and AA- – High Safety Bonds

A+ , A and A- – Adequate Safety Bonds

BBB+, BBB and BBB- – Moderate Safety Bonds

BB+ and BB – Inadequate Safety Bonds

B, C and D – High Risk Bonds (Junk Bonds)

Methods

Dimensionality Reduction Technique

For the purpose of reducing the dimensionality of the problem, we have used principal component analysis (PCA) (Everitt & Dunn, 2001; Maddala, 1997). PCA involves a mathematical procedure that transforms a number of (possibly) correlated variables into a (smaller) number of uncorrelated variables called *principal components*. The first principal component accounts for as much of the variability in the data as possible, and each succeeding component accounts for as much of the remaining variability as possible. This was applied on all 45 variables. The components were chosen based upon their Eigen values (components with Eigen values greater than the average were chosen). These components almost covered more than 50% of the variance in the samples.

Reduced Variable Technique

In this technique, a t-test over a multiple linear regression reduced the input variables from 45 to 20. These 20 variables were those which had coefficients more than 0.3. The list of these significant variables is shown in the Table 3.

Comparative Solution Methodology

We used the method of multiple linear regression (Maddala, 1997) on the original data as well as the components from the principal component analysis to predict rating and compared it with the original results obtained from fuzzy model and neural-fuzzy network.

Strict and conservative lower band (CLB) accuracy are given in percentages.

Strict accuracy (SA) – Percentage of cases where the predicted rating exactly matched the given rating.

Conservative lower band (CLB) – Percentage of cases where the output of the system falls, at most one level below the actual rating. This is significant because the loss on prediction of a low rating bond as high is much higher than the loss on predicting high rating bond as low.

Table 3. Reduced set of variables

Avg. debtors / avg. current assets
Avg. working capital / total assets
Pbdit(nnrt) / avg. capital employed
Sales / total asset
Sales / capital employed
Reserves / total assets
Operating profit / sales
Cash profits / total borrowings
Avg. inventory / avg. current assets
Avg. cash bank / avg. current assets
Avg. cash bank / quick assets
Current assets / sales
Current assets / total assets
Total assets
Sales
Age
Total borrowings / total assets
Total income / net worth
Current assets / current liabilities
Quick assets / total assets

Table 4. Dimensionality reduction technique

Architecture	SA	CLB
SOFM	33	79
Fuzzy C Means	47	79
Fuzzy Kohonen	51	80
Linear Regression	45	76

Table 5. Variable reduction technique

Architecture	SA	CLB
SOFM	52	76
Fuzzy C Means	55	81
Fuzzy Kohonen	62	80
Linear Regression	48	76

Results and Analysis

Dimensionality Reduction Technique

Table 4 gives the summary of the best results obtained from trying out the various combinations of the various architectures. We have considered three types of unsupervised networks, namely Kohonen, fuzzy c-means, and fuzzy-Kohonen. We found the best results with neuro-fuzzy technique.

Variable Reduction Technique

Table 5 gives the summary of the best results obtained from trying out the various combinations of the various architectures. We have considered three types of unsupervised networks, namely Kohonen, fuzzy c-means, and fuzzy-Kohonen. We found the best result with the neuro-fuzzy technique.

The strict accuracy varied from 33% for SOFM networks to 51% for Fuzzy Kohonen.

The strict accuracy varied from 48% for linear regression networks to 62% for fuzzy Kohonen. Neural networks and fuzzy techniques outperform the normal statistical methods. However, the conservative band accuracy is almost the same for all networks.

Comparison Between Dimensionality Reduction Technique and Variable Reduction Technique

We see from Table 5 that the variable reduction technique gives better strict accuracy across various architectures. Variable reduction techniques completely outperform the dimensionality reduction technique. We also see that the neuro-fuzzy system performs very well in both the techniques. The quintessence of designing intelligent systems of this kind is that neural networks recognize patterns and adapt themselves to cope with changing environments, and fuzzy inference systems incorporate human knowledge and perform inferencing and decision-making. The important thing about the constituents of hybrid systems is that they are complementary, not competitive, offering their own advantages and techniques to partnerships to allow solutions to otherwise unsolvable problems.

Conclusions

In this chapter, we have successfully established soft computing techniques for predicting bond ratings. We found that neuro-fuzzy systems outperform other types of networks. However, one of the major limitations of such techniques is their inability to explain their behavior. It is difficult, for example, to explain the relative importance of the various inputs. Often weight sensitivity analysis is carried out to develop an understanding of the model's behavior. Though we have not done the weight sensitivity analysis, this forms the basis for the formulation of rules, which can be applied in the form of supervised networks. The future scope of work is to develop some sort of rule base using the domain expertise of analysts to make accurate predictions. This can be achieved by incorporating the fuzzy inferencing systems (Takagi & Sugeno, 1985) inside the neural network learning.

References

Ball, G.H., & Hall, D.J. (1967). A clustering technique for summarizing multivariate data. *Behavioral Science, 12,* 153-155.

Deb, K. (2001). *Multi-objective optimization using evolutionary algorithms*. New York: John Wiley & Sons.

Bezdek, J.C. (1981). *Pattern recognition with fuzzy objective function algorithms*. New York; London: Plenum Press.

Bezdek, J.C., & Pal, S.K. (1992). *Fuzzy models for pattern recognition.* New York: IEEE Press.

Chaveesuk, R., Srivaree-ratana. C., & Smith, A.E. (1999). Alternative neural network approaches to corporate bond rating. *Journal of Engineering Valuation and Cost Analysis, 2*(2),117-131.

Everitt, B.S., & Dunn, G. (2001). *Applied multivariate data analysis* (2nd ed.). Arnold.

Hassoun, M.H. (1995). *Fundamentals of artificial neural networks.* Cambridge, MA: MIT Press.

Kohonen, T. (1989). *Self-organization and associative memory* (3rd ed.). Berlin, Heidelberg; New York: Springer Verlag.

Maddala, G.S. (1997). *Econometrics*. McGraw Hill.

Manuj, H.K. (1993). *An evaluation of the statistical properties of financial ratios*. Ph.D. thesis. Indian Institute of Management Calcutta.

Mehrotra, K., Mohan, C.K., Ranka, S., & Mohan, C.K. (1997). *Elements of artificial neural networks*. Penram International Publishing.

Rating Scan. (1999). *CRISIL,* August.

Refenes, Apostolos-Paul. (1994). *Neural networks in the capital markets.* New York: John Wiley & Sons.

Sehgal, K.S., Ramasubramanian, G.S., & Rajesh, K.M. (2001). *Application of fuzzy table look-up method to credit rating.* Euro Summer Institute XIX on Decision Analysis and Artificial Intelligence, Toulouse, France.

Takagi, T., & Sugeno, M. (1985). Fuzzy identification of systems and its application to modeling and control. *IEEE Transactions on System, Man, Cybernetics, 15*(1), 116-132.

Tsao, E.C.-K, Bezdek, J.C., & Pal, N.R. (1994). Fuzzy Kohonen clustering networks. *Pattern Recognition, 27*(5), 757-764.

Wang, L. (1997). *A course in fuzzy systems and control*. Prentice-Hall.

Appendix

AAA (Highest safety)	Judged to offer highest safety of timely payment of interest and princi Principal. Though the circumstances providing this degree of safety are likely to change, such changes as can be envisaged are most unlikely to affect Adversely the fundamentally strong position of such debentures.
AA (High safety)	Judged to offer high safety of timely payment of interest and principal. Differ in safety from 'AAA' debentures only marginally.
Investment grades	
A (Adequate safety)	Judged to offer adequate safety of timely payment of interest and principal. Pricipal.However, changes in circumstances can adversely affect such debentures More than those in the higher rated categories.
BBB (Moderate safety)	Judged to offer moderate safety of timely payment of interest and principal Principal for the present; however, changing circumstances are more likely to lead to a weakened capacity to pay interest and repay principal than for debentures Rated in higher categories.
Speculative grades	
BB (Inadequate safety)	Judged to carry inadequate safety of timely payment of interest and prin Principal, while less susceptible to default than other speculative grade deben in the immediate future; the uncertainties that the issuer faces could Lead to inadequate capacity to make timely payments.
B (High risk	Judged to have greater susceptibility to default; while currently interest and principal are met, adverse business or economic conditions would lead To lack or ability or willingness to pay interest or principal.
C (Substantial risk)	Judged to have factors present that make them vulnerable to default; timely Payment is possible only if favorable circumstances continue.
D (Default	Already in default and in arrears of interest or principal payments, or ex Expected to default on maturity. Extremely speculative - returns may be Realized only on reorganization or liquidation.

'+' (plus) or '-' (minus) signs may be applied for ratings from 'AA' to 'C' to reflect comparative standing within that category.

Chapter XI

Predicting Credit Ratings with a GA-MLP Hybrid

Robert Perkins, University College Dublin, Ireland

Anthony Brabazon, University College Dublin, Ireland

Abstract

The practical application of MLPs can be time-consuming due to the requirement for substantial modeler intervention in order to select appropriate inputs and parameters for the MLP. This chapter provides an example of how elements of the task of constructing a MLP can be automated by means of an evolutionary algorithm. A MLP whose inputs and structure are automatically selected using a genetic algorithm (GA) is developed for the purpose of predicting corporate bond-issuer ratings. The results suggest that the developed model can accurately predict the credit ratings assigned to bond issuers.

Introduction

This chapter demonstrates the capability of a backpropagation-trained, multi-layer perceptron (MLP) to accurately model the corporate bond-issuer credit rating process. Although MLPs have been widely applied to real-world problems, the development of

a quality MLP model for a specific task can be time-consuming, as the modeler must decide which inputs to use and what internal architecture to employ in the MLP. In the application in this chapter, an evolutionary algorithm (the genetic algorithm) is used to automate components of these tasks.

Most large firms raise both share and debt capital to provide long-term finance for their operations. Debt capital may be provided by a bank, or may be obtained by selling bonds directly to investors. As an example of the scale of the U.S. bond markets, the value of bonds issued in the first quarter of 2003 totalled $1.70 trillion (Bond Market Statistics, 2003). When a publicly traded company wants to issue traded debt or bonds (a bond is defined as a debt security which constitutes a promise by the issuing firm to pay a stated rate of interest based on the face value of the bond, and to redeem the bond at this face value at maturity), it must obtain a credit rating for the issue from at least one recognised rating agency such as Standard and Poor's (S&P), Moody's, or Fitches'. The credit rating represents the rating agency's opinion at a specific date, of the creditworthiness of a borrower in general (known as an issuer credit rating), or in respect of a specific debt issue (a bond credit rating). The rating serves as a surrogate measure of the risk of non-payment of the interest or non-repayment of the capital of a bond.

Several categories of individuals would be interested in a model which could produce accurate estimates of bond ratings. Such a model would be of interest to firms which are considering issuing debt as it would enable them to estimate the likely return investors would require, thereby providing information for the pricing of their bonds. The model also could be used to assess the creditworthiness of firms which have not issued debt, and hence do not already have a published bond rating. This could be useful to bankers or other companies which are considering whether they should extend credit to that firm. Much rated debt is publicly traded on stock markets, and bond ratings are typically changed infrequently. An accurate bond rating prediction model could indicate whether the current rating of a bond is still justified. To the extent that an individual investor could predict a bond rerating before other investors foresee it, this may provide a trading edge.

Motivation for Study

There are a number of reasons to suppose *a priori* that the combination of an evolutionary algorithm with a MLP can prove fruitful in the prediction of issuer bond ratings. The application domain is characterised by the lack of a strong theoretical framework and has a multitude of plausible, potentially interacting, explanatory variables. The first problem facing the modeler is the selection of a good subset of these variables, and the second problem is the selection of an appropriate model form. In applications of MLPs this is not a trivial task as many choices are open to the modeler, including the nature of the connection structure, the form of activation function at each node, and the choice of learning algorithm and its associated parameters. The selection of quality explanatory variables and model form represents a high-dimensional combinatorial problem, giving rise to potential for an evolutionary methodology which could automate this process (Mitchell, 1996). Such automated methodologies have clear potential for extension to a variety of data-mining applications. To date, only a relatively limited number of studies

have applied evolutionary methodologies — including genetic algorithms (GA), genetic programming (GP), and grammatical evolution (GE) — to the domain of bond rating prediction or the related domain of corporate failure prediction (Varretto, 1998; Kumar, Krovi, & Rajagopalan, 1997; Back, Laitinen, Sere, & van Wezel, 1996; Brabazon & O'Neill, 2003). This chapter addresses this gap.

Structure of the Chapter

The rest of this chapter is organised as follows. The next section provides a concise review of the bond rating process, and of prior literature concerning bond rating prediction. This is followed by a section which introduces the GA, and which describes how MLP and GA methodologies can be combined. Next, the dataset and the methodology used is described. The remaining sections provide the results of the experiments followed by a number of conclusions and suggestions for future work.

Bond Rating

Although the precise notation used by individual rating agencies to denote the creditworthiness of a bond or an issuer varies, in each case the rating is primarily denoted by a letter. Taking the rating structure of S&P as an example, ratings are broken down into 10 broad classes (see Table 1). The strongest rating is denoted AAA, and the ratings then decrease in the following order, AA, A, BBB, BB, B, CCC, CC, C, D. Therefore, credit

Table 1. Rating letter grades for Moody's and S&P

Rating		Definition
Moodys	S&P	
Aaa	AAA	Best quality with the smallest degree of investment risk. Referred to as "gilt edged bonds."
Aa	AA	High-quality, rated lower than AAA as margins of protection for the lender are not as great as for AAA.
A	A	Upper Medium Grade Obligations - more vulnerable to changing economic conditions.
Baa	BBB	Medium Grade Obligations - neither highly protected nor poorly secured.
	BB	Speculative debt with moderate security. Protection of interest and principal payments is not well safeguarded.
B	B	Has notable risk of future default. Limited degree of assurance with respect to interest and principal payments.
Caa	CCC	Poor quality bonds with considerable risk characteristics.
Ca	CC	Bonds with a very high chance of default.
C	C	Bonds issued by companies which are in bankruptcy.
	D	Bonds which are in default.

ratings are conveyed to investors by means of a discrete, mutually-exclusive, letter grade. Ratings between AAA and BBB (inclusive) are considered by investors to represent *investment grade* ratings, with lower quality ratings considered to represent debt issues with significant speculative or risky characteristics. Sub-investment grade bonds are known as *junk bonds*. A C grade rating represents a case where a bankruptcy petition has been filed, and a D rating represents a case where the borrower is currently in default on their financial obligations. Ratings from AAA to CCC can be modified by the addition of a + or a - to indicate at which end of the category the rating falls.

The Bond Rating Process

Rating agencies earn fees from bond issuers for evaluating the credit status of new bonds, and for maintaining credit rating coverage of these bonds. A company obtains a credit rating for a debt issue by contacting a rating agency and requesting that an *issue rating* be assigned to the new debt, or that an overall *issuer rating* be assigned to the company. As part of the process of obtaining a rating, the firm submits documentation including recent financial statements, a prospectus for the debt issue, and other non-financial information. Discussions take place between the rating agency and management of the firm, and a rating report is then prepared by the credit analysts examining the firm. This rating report is then considered by a committee in the rating agency which decides the credit rating to be assigned to the debt issue or to the issuer. Rating agencies emphasise that the credit rating process involves the consideration of financial as well as non-financial information about the firm, and indicate that the rating process also considers industry and market-level factors. The precise factors and related weighting of these factors are not publicly disclosed by the rating agencies.

Subsequent to their initial rating, a bond or issuer may be re-rated upwards (upgrade) or downwards (downgrade) if company or environmental circumstances change. A re-rating of a bond from investment grade to junk bond status, such bonds are colourfully termed a *fallen angel*, may trigger a significant sell-off. Many institutional investors are only allowed by external or self-imposed regulation, to hold bonds of investment grade.

The practical effect of a bond (or an issuer) being assigned a lower rather than a higher rating is that its perceived riskiness in the eyes of potential investors increases, consequently, the required interest yield of the bond rises. As an estimate of this differential, the rate spread between investment grade and junk bonds is estimated at about 5 points (*Business Week,* 2003). In addition to influencing the interest yield of a bond, the credit rating assigned also impacts on the marketability of a bond.

Bond Rating Prediction

The objective of the literature on bond rating prediction is to construct a model of rating agency behaviour, using publicly available information. A large literature exists on bond rating prediction. Earliest attempts utilised statistical methodologies such as linear regression (OLS) (Horrigan, 1966; Pogue & Soldofsky, 1969), multiple discriminant analysis (Pinches & Mingo, 1973), the multi-nominal logit model (Ederington, 1985), and

ordered-probit analysis (Gentry, Whitford, & Newbold, 1988). The results from these studies varied, and typically results of about 60 to 70% prediction accuracy (out-of-sample) were obtained when discriminating between investment/non-investment grade ratings, using financial data as inputs. With the advent of artificial intelligence and machine learning, the range of techniques applied to predict bond ratings has expanded to include neural networks (Maher & Sen, 1997), case-based approaches (Shin & Han, 2001), and support vector machines (Huang, Chen, Hsu, Chen, & Wu, 2004). In the case of prior neural network research, the predictive accuracy of the developed models has varied. Several studies employed a binary predictive target and reported good classification accuracies. For example, Dutta and Shekhar (1988) used a neural network to predict AA or non-AA bond ratings, and obtained an accuracy of approximately 83.3%. This study used a small sample size (47 companies). A more difficult problem is that of predicting the exact rating for a bond, from the multiple possible ratings which a rating agency can assign. Moody and Utans (1995) produced predictive accuracies of 36.2% in the 16 rating case, improving to 63.8% when the number of ratings was reduced to five cases. Unfortunately, the study did not disclose either the sample size or the model input variables, making it difficult to assess its generalisability.

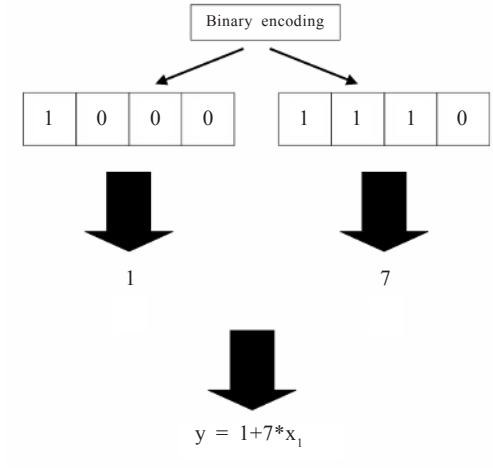
The task of prediction of bond-issuer credit ratings bears notable similarity to the well-known corporate failure prediction problem. In each domain, a key objective is to anticipate financial distress. Prior research on corporate failure has strongly indicated that models employing financial data as inputs are capable of predicting corporate failure several years before it occurs (Altman, 1993; Brabazon & Keenan, 2004). This suggests that it is plausible to assume that data extracted from financial statements of bond issuing firms may provide a useful input into a credit rating prediction model for those companies.

The Genetic Algorithm

Although the development of the GA dates from the 1960s, Holland (1975) first brought them to the attention of a wide audience. GAs have been applied in a variety of business settings including the domain of finance (Bauer, 1994; Deboeck, 1994; Varetto, 1998), and several branches of management science such as inventory management (Sarker & Newton, 2002) and vehicle routing (Baker & Ayechew, 2003).

The GA can be considered as a mathematical optimisation algorithm whose workings are metaphorically inspired by neo-Darwinian evolution. The GA adopts a populational unit of analysis, and each member of the population corresponds to an encoding of a potential solution to the problem of interest. Traditionally, the encodings (or *genotypes*) were binary strings (0,1,0,1, ...), but the GA also can operate on integer or real-valued encodings. The genotypes can be interpreted (or decoded) in a wide variety of ways depending on the nature of the problem. For example, a genotype could be interpreted to produce a problem solution (the *phenotype*) such as the coefficients for a regression model (see Figure 1), a classification rule, a production schedule, or a MLP structure. The quality of each phenotype is determined by reference to a problem-specific fitness function. Evolution in the population of genotypes is simulated by means of a pseudo-

Figure 1. An example of how a binary string can encode two coefficients for a linear model. The string is initially decoded into two integers, which in turn are inserted in the linear model as coefficients.



natural selection process using differential-fitness selection, and pseudo-genetic operators to induce variation in the population between successive generations. It is important to note that the evolutionary process of a GA operates on the genotypes, rather than directly on the solutions themselves.

In general, evolutionary algorithms including the canonical GA, can be characterised as:

$$x[t+1] = v(s(x[t]))$$

where $x[t]$ is the population of genotypes at iteration t, $v(.)$ is a random variation operator (crossover and mutation operators), and $s(.)$ is a selection operator. Therefore, the canonical GA can be described as a stochastic algorithm that turns one population of genotypes into another, using selection, crossover, and mutation. Selection exploits information in the current population, concentrating interest on "high-fitness" genotypes. Crossover and mutation perturb these in an attempt to uncover better genotypes, and these operators can be considered as general heuristics for exploration. The GA can be formulated as a finite-dimension Markov chain, wherein each state corresponds to a configuration of the population of genotypes. Depending on the form of genetic operators implemented in the algorithm, the transition probabilities between states will vary. In the canonical GA, the inclusion of a mutation operator implies that there are no absorbing states in the Markov process, and that all states can potentially be visited.

Application of a GA

Although the GA can utilise a variety of encodings, we concentrate on the binary encoding in our description of the workings of the GA in this subsection. Once the initial population of genotypes has been formed (perhaps randomly) and the fitness of their corresponding phenotypes have been evaluated, a reproductive process is applied in which the genotypes corresponding to better quality phenotypes have a higher chance of being selected for propagation of their genes (bits) into the next generation of genotypes. Over a series of generations, the better genotypes, in terms of the given (problem-specific) fitness function, tend to flourish in the population, and the poorer solutions tend to disappear.

The reproductive stage provides the engine for this search process, as it biases the search process toward high quality existing genotypes and uses information from these to navigate the search space. The reproductive process is generally governed by two operators, crossover and mutation (Mitchell, 1996). The crossover operator takes two genotypes from the population and swaps component parts of each in order to create potential members of the next generation of the population. As an example, suppose two parent genotypes are selected, and that they have the following binary strings: (0001 1101) and (1100 1100). If a single point crossover is applied after the fourth bit, the two resulting child genotypes are (0001 1100) and (1100 1101). The mutation operator causes small random changes in one or more of the genes of a child solution. In a binary representation, a 0 may mutate to a 1, or a 1 to a 0. Successive generations of genotypes are evaluated, selected for reproduction based on their differential fitness, and then subjected to a reproductive process until predefined stopping criteria are satisfied. An outline the operation of a canonical GA (Goldberg, 1989) is provided as:

1. Initialise a population of genotypes.
2. Evaluate each genotype in the population by applying the objective function to its corresponding solution (phenotype).
3. Select higher fitness genotypes to be parents in the creation of new genotypes.
4. Create new genotypes by applying crossover and mutation.
5. Delete members of the current population to make room for newly created genotypes.
6. Evaluate each genotype in the population by applying the objective function to its corresponding solution (phenotype).
7. If the stopping criterion has been satisfied, stop and return the genotype with best fitness, otherwise go back to Step 3.

Combining GA and MLP Methodologies

Despite the apparent dissimilarities between GA and MLP methodologies, they can usefully complement each other by combining the non-linear mapping capabilities of a

MLP, with the optimising capabilities of a GA. For example, a GA could be used to select any or all of the following:

- a choice of model inputs
- the number of nodes in each hidden layer
- the nature of activation functions at each node
- the form of learning algorithm and its associated parameters
- the number of hidden-layers
- the connection weights between each node

When evolving MLPs using a GA, the first step is to define how the genotype is decoded into a specific MLP. The genotype could encode the MLP using a binary, an integer, or a real-valued representation (or some mix of these), and the form of encoding will have implications for the appropriate design of crossover and mutation operators. There are a multitude of ways that a MLP structure could be encoded, and a multitude of GA-variants which could be applied to evolve these encodings. Taking a simple example, consider a case where a GA is being used to evolve the inputs and structure of a MLP, but not its connection weights. A canonical GA-MLP algorithm could be defined as follows:

1. Decode the genotype into a MLP structure.
2. Randomly initialise the weights for the arcs in this structure.
3. Train the MLP using the backpropagation algorithm.
4. Determine the classification accuracy (fitness) of the resulting MLP.
5. Perform fitness-based selection to create a new population of genotypes.
6. Create diversity in the genotypes of the new population, using crossover and mutation.
7. Replace the old population by the new one.
8. Repeat the above steps until stopping criteria are met.

This represents an example of a *generational* GA, as the entire existing population is replaced by a newly created population in each iteration of the algorithm. Other replacement strategies that could be used include *steady state*, where only a small number of the least-fit individuals in the population are replaced by better newly-created solutions in each generation. The selection step in the GA also can be implemented in a variety of ways, including fitness-proportionate selection, ranking selection, or tournament selection (Mitchell, 1996).

The GA/MLP combination described produces *dual-level* learning. The choice of inputs and the architecture of the MLP is encoded in a genetic structure (for example a binary string) which is altered over time as the GA searches the space of binary strings in order

to uncover strings which encode better MLPs. This represents *phylogenetic* (or genetic) *learning*. For each distinct MLP structure, the backpropagation algorithm is used in order to discover good weights, representing *epigenetic* (or lifetime) *learning* (Sipper & Sanchez, 1997). For a more detailed discussion of issues arising in combining GA and MLP methodologies, refer to Yao (1999) and Dorado, Santos, and Rabuñal (2001).

Methodology

Two experiments are conducted. The utility of a MLP/GA model to distinguish between investment grade and non-investment grade bond ratings is initially investigated. Secondly, the ability of a MLP/GA model to predict precise rating classes for a series of bonds is investigated. In this chapter, we limit attention to four bond classes A, BBB, BB, and B.

Data Collection

In collecting data, attention was limited to non-financial companies in order to ensure that the accounting ratios obtained were comparable across the companies. The following criteria were applied to obtain the final sample:

1. Accounting information and issuer ratings must be available from the S&P Compustat database.
2. Companies must have a financial year-end of December 31, 2003.
3. Companies must have an S&P long-term issuer rating at December 31, 2003.
4. Accounting data must be available for three years prior to and including 2003.

Table 2. Issuer-rating dataset

	Bond Rating	Number of Bonds
Investment Grade	AAA	6
	AA	13
	A	137
	BBB	206
Non-Investment Grade	BB	172
	B	117
	CCC	12
	CC	2
	D	2
Total		667

Satisfaction of the criteria left a sample of 667 companies with long-term issuer ratings ranging from AAA to D (see Table 2). Companies with a rating of AAA, AA, CCC, CC, and D were eliminated due to the small number of companies with these ratings, leaving a final dataset of 632 companies. The next stage of sample selection required the random division of the remaining companies into training and testing datasets.

In the binary prediction case (prediction of investment grade versus non-investment grade), 460 firms were randomly selected from the total sample of 632 firms, to produce two groups of 230 investment grade and 230 sub-investment grade ratings. The 460 firms were randomly allocated to the training set (320) or the hold-out sample (140), ensuring that each set was equally balanced between investment grade and non-investment grade ratings. In the multi-rating case, the training and hold-out datasets were equally balanced between A, BBB, BB, and B rated firms, again with a training set of 320, and a hold-out sample of 140. The dataset of 632 companies was recut seven times, in order to create a series of training and out-of-sample datasets.

Selection of Input Variables

The selection of input variables was guided by prior literature on bankruptcy prediction, literature on bond rating prediction, and by statistical analysis. Several previous bond rating prediction studies (Horrigan 1966; West, 1970; Pogue & Soldofsky, 1969; Pinches & Mingo, 1973; Dutta & Shekhar, 1988; Surkan & Singleton, 1990; Brennan & Brabazon, 2004; Yesilyaprak, 2004), and seven corporate failure studies were reviewed, yielding 54 potential input variables. From these a set of 20 distinct potential input variables was identified.

The objective in selecting a set of proto-explanatory variables is to choose financial variables whose values vary between companies in different bond rating classes, and where information overlaps between the variables are minimised. Following a statistical analysis of the financial ratios, the set of potential inputs was reduced to 14 (see Table 3). All selected ratios demonstrated a statistically different mean at the 1% level between the investment grade/junk bond categories. As expected, the financial ratios for the investment grade issuer ratings are stronger than those for the junk bond ratings. The only exception is the current ratio which is stronger for the junk bond rated companies, possibly indicating a preference for these companies to hoard short-term liquidity, as their access to long-term capital markets is limited.

GA/MLP Model Construction

Initially, a population of 50 binary genotypes, each corresponding to a distinct MLP structure and set of explanatory variables, was randomly generated. The genotype was designed to encode a range of structures in which:

Table 3. Mean of selected financial ratios for the investment grade and non-investment grade issuers

	Investment Grade	Non-Investment Grade
Size		
Log Total Assets	3.748	3.216
Profitability		
Net income/total assets	0.048	0.0005
Net income/sales	0.070	-0.115
Retained earnings/total assets	0.221	-0.191
EBIT/Interest	9.283	2.357
Liquidity		
Current ratio	1.492	2.099
Working capital/sales	0.089	0.404
Leverage		
Long-term debt/total assets	0.262	0.391
Activity		
Sales/total assets	0.757	0.873
Inventory/working capital	0.378	0.573
Stability		
[NM 2003/NM 2001] - 1	0.362	-0.563
[TA 2003/TA 2001] - 1	0.017	0.015
[CR 2003/CR 2001] - 1	0.123	0.292
[NI/TA 2003/NI/TA 2001] - 1	0.248	-0.472

- the number of inputs could vary up to a maximum of eight,
- the form of activation function at each processing node could vary between linear or logistic, and
- the number of hidden layer nodes could vary up to a maximum of four.

In developing the models, a limit of four hidden layer nodes and eight input variables was initially imposed. The object in placing these constraints on the network structures being searched by the evolutionary process was to conserve degrees of freedom and reduce the danger of model over-fit. A series of experiments were subsequently undertaken to determine whether these limitations adversely impacted on the accuracy of the developed classifiers. In the initial experiments, each genotype was characterised as a 23 bit binary string, where each of the first 14 bits indicate whether a specific input variable is used, and the next 8 bits (split into four groups of two bits) correspond to a choice of whether a particular hidden layer node is used, and the form of the activation function at that hidden node. The final bit encoded the choice of activation function at the output node.

Each resulting MLP was trained three times using the backpropagation algorithm with different initial randomisations of the weight vectors, reducing the *noisy fitness* evaluation problem which can emerge when a network architecture is evolved (Yao, 1999). The average of the three fitness values was used to drive the evolutionary process. The model-building dataset was subdivided to provide both a training and (in-sample) test dataset. The networks were constructed using the training data and network fitness was assessed during the evolutionary process based on their performance on the (in-sample) test dataset, where networks with higher classification accuracy were considered more fit. The out-of-sample data was not used in the model development process.

The evolutionary process during each generation was as follows. A roulette selection process, explicitly favouring more fit members of the current population, was applied to select members for the mating pool. Parents were randomly selected from this pool, and single-point crossover was applied, with probability of 0.5 to produce child genotypes. A mutation operator was next applied whereby each bit of a child genotype can be mutated with a probability of 0.04, in order to produce approximately one expected mutation per child genotype. Finally, the current population of genotypes was replaced by the child genotypes. The choice of parameter values for the GA is judgmental, and was guided by initial experiments, and by Mitchell (1996) who suggests that population sizes of about 50 crossover rates of about 0.6, and bit mutation rates of about 0.01, are commonly used in practical applications of GAs. The future work section of the chapter outlines alternative selection, replacement, and diversity-generating strategies which could be considered in future extensions of this study.

Results and Discussion

Tables 4 and 5 provide the in-sample and out-of-sample confusion matrices for the predicted versus the actual bond rating for the binary classification case, averaged across seven recuts of the dataset. A classification accuracy in-sample (out-of-sample) of 83.08 (81.43)% is obtained, and it is noted that the accuracy is reasonably symmetric between the two rating classes. Table 6 provides the out-of-sample confusion matrix for the predicted versus the actual bond rating, for the four rating classification problem. As would be expected, the four class case is a more demanding test of a classification model, and the overall in-sample (out-of-sample) classification accuracy obtained drops to 48.75 (49.29)%. To examine the classification accuracy more closely, a "one-step away" classification accuracy also was calculated, which assesses how many of the predicted ratings are within one grade of the actual rating. In this case, a classification accuracy of 92.5 (92.1)% is obtained, indicating that relatively few large rating misclassifications are occurring.

In order to assess the out-of-sample classification accuracies, Press's Q statistic (Hair, Anderson, Tatham, & Black, 1998) was calculated for both the binary classification model and the four-rating classification model. In both cases, the null hypothesis, that the out-of-sample classification accuracies are not significantly better than those that could occur by chance alone, was rejected at the 1% level.

Table 4. In-sample binary classification accuracies of the best evolved MLP, averaged over seven recuts

	Predicted		
Actual	*Investment grade*	*Non-investment*	*Total*
Investment grade	126.6 (79.11%)	33.4 (20.89%)	160 (100%)
Non-investment	20.7 (12.95%)	139.3 (87.05%)	160 (100%)
Total	147.3	172.7	320 (100%)

Table 5. Out-of-sample binary classification accuracies of the best evolved MLP, averaged over seven recuts

	Predicted		
Actual	*Investment grade*	*Non-investment*	*Total*
Investment grade	60 (85.71%)	10 (14.29%)	70 (100%)
Non-investment	16 (22.86%)	54 (77.14%)	70 (100%)
Total	76	64	140 (100%)

Table 6. Out-of-sample accuracy for four classification case (best evolved MLP), averaged over seven recuts

	Predicted				
Actual	*A*	*BBB*	*BB*	*B*	*Total*
A	12 (34.29%)	17 (48.57%)	4 (11.43%)	2 (5.71%)	35 (100%)
BBB	5 (14.29%)	24 (68.57%)	6 (17.14%)	0 (0.0%)	35 (100%)
BB	1 (2.86%)	10 (28.57%)	14 (40.0%)	10 (28.57%)	35 (100%)
B	1 (2.86%)	3 (8.57%)	12 (34.29%)	19 (54.29%)	35 (100%)
Total	19	54	36	31	140

Table 7 provides details of the ratios utilised by the best evolved MLP for the binary classification case. The key variables are the size of the firm, its profitability, its profit history since incorporation, its interest coverage, and the change in its total assets in the past three years (a proxy for its recent profit history). Each of these ratios are plausible variables in assessing the future viability of a firm. It was noted that most of the MLPs in the population utilised similar inputs by the end of the 80 evolutionary iterations. Examination of the internal structure of the best MLP showed that it utilised four hidden layer nodes with logistic activation functions at each hidden layer node. Table 8 also provides details of the ratios utilised by the best MLP model evolved for the four rating classification case. The ratios selected are similar to those for the binary classification case, the size of the firm, its profitability, a measure of its short-run liquidity, and the change in its total assets over the past three years. Examination of the internal structure of the best evolved MLP showed that it used three hidden layer nodes (two logistic activation functions and a linear activation function).

Table 7. Inputs used in the best MLP model for the binary classification and the four rating cases

Binary Classification	***Four Rating Classification***
Log (Total assets)	Log (Total assets)
Net Income / Sales	Current Ratio
EBIT / Interest	% change in Total Assets
% change in Total Assets	Net Income / Total Assets
Net Income / Total Assets	
Retained Earnings / Total Assets	
Inventory / Working Capital	

Table 8. Average populational fitness (in-sample) of the 10 best MLPs, on first randomisation of dataset

Generation	**Mean (%)**	**Std. Deviation**
10	84.05	0.68
20	84.75	0.50
30	85.04	0.44
40	85.29	0.35
50	85.31	0.34
60	85.39	0.36
70	85.46	0.33
80	85.52	0.33

In selecting the number of generations (80), we were guided by the results of preliminary experiments which suggested that average fitness (classification accuracy on the training data) in the population typically converged to a plateau within about 50 generations. To demonstrate the convergence of classification accuracy, a tabulation of average populational fitness of the 10 best MLPs in the population, every 10 generations for the first randomisation of the dataset is provided in Table 10. The convergence of average in-sample fitness in the population is noticeable by generation 50.

In evolving the above MLP structures, the number of hidden nodes and ratio inputs which could be included was limited to four and eight, respectively. This was done in order to conserve degrees of freedom in the final models so as to reduce the chance of overfit. A natural question is whether these limits had an adverse effect on the classification accuracies obtained. In order to examine this issue, a series of additional experiments were conducted using the binary classification dataset (investment grade versus non-investment grade). In the first experiment, the number of permissible inputs was increased to 12. In the second experiment, the number of inputs was restricted to eight, but the internal structure of the MLP was allowed to contain up to two hidden layers, and each hidden layer could contain up to eight nodes. It was found that the classification accuracy in-sample and out-of-sample did not change markedly in either

of these experiments from that already reported. Typically, the evolved MLPs continued to use a maximum of seven inputs and one hidden layer.

Hence, there was no evidence that the restrictions on the number of inputs or the number of hidden layer nodes adversely impacted on the reported results.

Conclusions

This chapter examined the utility of a MLP/GA hybrid, which combines the global search potential of an evolutionary algorithm with the non-linear modeling capabilities of a MLP in modeling the corporate bond issuer rating process. A series of classification models were constructed. Despite using input financial data drawn from companies in a variety of industrial sectors, the models demonstrated a useful capability to discriminate between bond rating classifications and obtained results which were significantly better than random chance. The chapter also demonstrated the potential for the application of evolutionary algorithms to automate the time-consuming process of creating MLPs.

Future Work

This chapter provides an example of how elements of the task of constructing a MLP can be automated by means of an evolutionary algorithm. A variety of avenues are left for further experimentation. In respect to the GA, there are a variety of alternative selection, replacement, and diversity-generating strategies which could be employed and which could improve the results further. For example, the use of roulette selection can lead to a loss of diversity in the population of genotypes, leading to a convergence of the population of resulting MLP structures. An interesting extension of the study would be to examine whether the use of a selection method with lower selection pressure would enhance the quality of the MLPs evolved by maintaining a more diverse population of genotypes. Methods of maintaining diversity include the implementation of a *crowding* operator (Mitchell, 1996), whereby a newly formed genotype replaces the genotype most similar to itself in the existing population, or the implementation of *fitness-sharing* (Goldberg & Richardson, 1987) whereby the fitness of genotypes is reduced if there are other similar genotypes in the population. This latter approach can lead to speciation in the population of genotypes, leading to the climbing of distinct peaks on the fitness landscape, if they exist. If such peaks have similar fitness, this would suggest that there are a range of MLP structures which will produce equivalent results. Alternative replacement strategies which could be applied to the problem include steady-state, where only a few individuals, usually the least-fit, are replaced by new genotypes in each generation. Alternative forms and rates of crossover, such as two-point crossover or uniform crossover, also could be explored to determine whether the classification accuracies could be further improved. More generally, it is noted that there are a large

range of applications of MLPs in the domain of finance including market prediction, credit scoring, sales/earnings forecasting, fraud detection, and portfolio optimisation (Wong, Lai, & Lam, 2000). These problems offer potential for the application of a combined GA/MLP methodology.

References

Altman, E. (1993). *Corporate financial distress and bankruptcy*. New York: John Wiley and Sons.

Back, B., Laitinen, T., Sere, K., & van Wezel, M. (1996). *Choosing bankruptcy predictors using discriminant analysis, logit analysis and genetic algorithms.* Technical Report no. 40. Turku Centre for Computer Science, Turku School of Economics and Business Administration.

Baker, B.M., & Ayechew, M.A. (2003). A genetic algorithm for the vehicle routing problem. *Computers & Operations Research, 30*(5), 787-800.

Bauer, R. (1994). *Genetic algorithms and investment strategies*. New York: John Wiley & Sons.

Bond Market Statistics (2003). New York: The Bond Market Association.

Brabazon, A. (2002). Neural network design using an evolutionary algorithm. *Irish Accounting Review, 9*(1), 1-18.

Brabazon, A., & Keenan, P. (2004). A hybrid genetic model for the prediction of corporate failure. *Computational Management Science, 1*(3-4), 293-310.

Brabazon, A., & O'Neill, M. (2003). Anticipating bankruptcy reorganisation from raw financial data using grammatical evolution. In G. Raidl, J.A. Meyer, M. Middendorf, S. Cagnoni, J.J.R. Cardalda, D.W. Corne, et al. (Eds.), *Proceedings of EvoIASP 2003, Lecture Notes in Computer Science (2611): Applications of Evolutionary Computing* (pp. 368-378). Berlin: Springer-Verlag.

Brennan, D., & Brabazon, A. (2004, June 2004). Corporate bond rating using neural networks. In H. Arabnia et al. (Eds.), *Proceedings of the International Conference on Artificial Intelligence 2004 (ICAI '04)*, Las Vegas, NV (vol. 1, pp. 161-167). CSEA Press.

Business Week (2003, August 4). A frenzied race to refinance. *Business Week*, p. 38.

Deboeck, G. (1994). *Trading on the edge: Neural, genetic, and fuzzy systems for chaotic financial markets*. New York: John Wiley & Sons.

Dorado, J., Santos, A., & Rabuñal, J. (2001, June 13-15). Connectionist models of neurons, learning processes and artificial intelligence. In *Proceedings of the 6th International Work-Conference on Artificial and Natural Neural Networks, IWANN 2001*, *Lecture Notes in Computer Science (2084),* Granada, Spain (vol. 1, pp. 717-724). Berlin: Springer-Verlag.

Dutta, S., & Shekhar, S. (1988). Bond rating: A non-conservative application of neural networks. *Proceedings of IEEE International Conference on Neural Networks* (vol. 2, pp. 443-450).

Ederington, H. (1985). Classification models and bond ratings. *Financial Review, 20*(4), 237-262.

Gentry, J., Whitford, D., & Newbold, P. (1988). Predicting industrial bond ratings with a probit model and funds flow components. *Financial Review, 23*(3), 269-286.

Goldberg, D. (1989*). Genetic algorithms in search, optimization and machine learning.* Boston: Addison Wesley Longman.

Goldberg, D., & Richardson, J. (1987). Genetic algorithms with sharing for multi-modal function optimization. In J. Grefenstette (Ed.), *Genetic algorithms and their applications. Proceedings of the Second International Conference on Genetic Algorithms*. Erlbaum.

Hair, J., Anderson, R., Tatham, R., & Black, W. (1998). *Multivariate data analysis*. Upper Saddle River, NJ: Prentice Hall.

Holland, J. (1975). *Adaptation in natural and artificial systems.* Ann Arbor: University of Michigan Press.

Horrigan, J. (1966). The determination of long term credit standing with financial ratios, *Journal of Accounting Research*, (supplement), 44-62.

Huang, Z., Chen, H., Hsu, C., Chen, W., & Wu, S. (2004). Credit rating analysis with support vector machines and neural networks: A market comparative study. *Decision Support Systems, 37*(4), 543-558.

Kumar, N., Krovi, R., & Rajagopalan, B. (1997). Financial decision support with hybrid genetic and neural based modelling tools. *European Journal of Operational Research*, *103*(2), 339-349.

Maher, J., & Sen, T. (1997). Predicting bond ratings using neural networks: A comparison with logistic regression. *Intelligent Systems in Accounting, Finance and Management*, *6,* 23-40.

Mitchell, M. (1996). *An introduction to genetic algorithms*. Cambridge, MA: MIT Press.

Moody, J., & Utans, J. (1995). Architecture selection strategies for neural networks application to corporate bond rating. In A. Refenes (Ed.), *Neural networks in the capital markets* (pp. 277-300). Chichester, UK: Wiley.

Pinches, G., & Mingo, K. (1973). A multivariate analysis of industrial bond ratings. *Journal of Finance, 28*(1), 1-18.

Pogue, T., & Soldofsky, R. (1969). What's in a bond rating? *The Journal of Financial and Quantative Analysis, 4*(2), 201-228.

Sarker, R., & Newton, C. (2002). A genetic algorithm for solving economic lot size scheduling problem. *Computers & Industrial Engineering, 42*(2-4), 189-198.

Shin, K., & Han, I. (2001). A case-based approach using inductive indexing for corporate bond rating. *Decision Support Systems, 32,* 41-52.

Sipper, M., & Sanchez, E. (1997). A phylogenetic, ontogenetic, and epigenetic view of bio-inspired hardware systems. *IEEE Transactions on Evolutionary Computation, 1*(1), 83-97.

Surkan, A., & Singleton, J. (1990). Neural networks for bond ratings improved by multiple hidden layers. In *Proceedings of the IEEE International Conference on Neural Networks,* San Diego, CA (vol. 2, pp. 163-168).

Varetto, F. (1998). Genetic algorithms in the analysis of insolvency risk. *Journal of Banking and Finance, 22*(10), 1421-1439.

West, R. (1970). An alternative approach to predicting corporate bond ratings. *Journal of Accounting Research, 7,* 118-127.

Wong, B., Lai, V., & Lam, J. (2000). A bibliography of neural network business applications research: 1994-1998. *Computers and Operations Research, 27*(11-12), 1045-1076.

Yao, X. (1999). Evolving artificial neural networks. *Proceedings of the IEEE, 87*(9), 1423-1447.

Yesilyaprak, A. (2004). Bond ratings with artificial neural networks and econometric models. *American Business Review, 22*(1), 113-123.

Section VI

Other Applications

Chapter XII

Music and Neural Networks

Giuseppe Buzzanca, State Conservatory of Music, Bari, Italy

Abstract

This chapter pertains to the research in the field of music and artificial neural networks. The first attempts to codify human musical cognition through artificial neural networks are taken into account as well as recent and more complex techniques that allow computers to learn and recognize musical styles, genres, or even to compose music. Special topics covered are related to the representation of musical language and to the different systems used for solving them, from classic backpropagation networks to self-organizing maps and modular networks. The author hopes that this chapter will disclose some significant information about this emerging but nonetheless important subfield of AI and at the same time increase some interest and allow for a better understanding of this complex field.

Introduction

Some 20 years ago, James McClelland and David Rumelhart (1986) commented on an apparent incongruity: In spite of the fact that more and more computationally powerful machines were available to researchers, human ability in specific tasks was still to be

regarded as incomparable. Besides that, in connection with certain aspects, for example the recognition of forms, what machines could do (albeit improved with reference to miniaturization and computational power) was simply insignificant.

What makes people smarter than machines? They certainly are not quicker or more precise. Yet people are far better at perceiving objects in natural scenes and noting their relations, at understanding language and retrieving contextually appropriate information from memory, at making plans and carrying out contextually appropriate actions, and at a wide range of other natural cognitive tasks. People also are far better at learning to do these things more accurately and fluently through processing experience. What is the basis for these differences? ...In our view, people are smarter than today's computers because the brain employs a basic computational architecture that is more suitable to deal with a central aspect of the natural information processing tasks that people are so good at...we will show through examples that these tasks generally require the simultaneous consideration of many pieces of information or constrains. (Rumelhart & McClelland, 1986, p. 3)

McClelland and Rumelhart recognized the fundamental computational architecture of the brain (billions of units connected among them, the neurons) as well as in his ability to instantaneously contemplate abundant constraints and "splinters" of knowledge, one of the reasons for the absolute perfection of the human capabilities in managing information. Their lesson would have been able to induce, over the years, an extraordinary number of progresses in the development of intelligent systems; also, a significant part of the scientific community would have reread cognition from innovative perspectives, at such a degree that the saying "connectionist revolution" would have been adopted by some scholars (Medler, 1998).

In this chapter, we will take into consideration some fundamental aspects on the theme of music and ANNs, by introducing both the epistemological and methodological topics of this rather new area of knowledge (*music and artificial intelligence* and consequently music and artificial neural networks); subsequently, we will present a broad literature review, divided into the literature of early years and current literature. The necessity to distinguish between old and new literature comes from the rather different perspective depicted: more "low-level" oriented in the first years (as expected since representational issues — how to encode pitches, durations, etc. — were to be solved to begin with), more focused on the creation of broad models of musical cognition in the recent years (in the remarkable attempt to model high-level topics like the recognition of genres, musical style, etc.).

A discussion on ANNs and music also poses two fundamental questions to be addressed. These questions cross over the boundaries represented by the computational aspects of the paradigm. In fact, they appear to be closely linked on one hand to the possibility of using connectionism to articulate realistic models of musical cognition in general (thus achieving a better understanding of those aspects of cognition that seem so natural to human beings); on the other to the quest for the solution of problems not imitable (or at least hardly imitable) by traditional artificial intelligence (AI) techniques: Very much is needed, in fact, to embody complex, multidimensional, meaningful stimuli

and develop effective models (e.g., for face recognition or musical style recognition). This last point has to be considered particularly important in connection with the themes of this chapter, since neural networks (besides the better degree of "biological realism" they offer for the modeling of musical cognition, see Buzzanca, 2002a) have allowed for the solution of problems to be deemed simply insurmountable with traditional AI approaches, like in the case of the studies in automatic music performance (see Bresin, 2000).

Music and AI

A chapter devoted to artificial neural networks and music research has to take into account that the reader may come from very different fields of expertise: music, computer science, cognitive science, and so forth. Consequently, it seems desirable to recall a few general notions mostly concerning this rather new field where music and computer science gather; in the following, the reader is supposed to be familiar with the basic ideas in neural networks and music theory.

Music and AI (also called *cognitive musicology* or *computational musicology*) is an emerging subfield of AI, which takes advantage of a variety of disciplines and aims to a better understanding of "how" music works (see Reck Miranda, 2000, for example). This complex and very rich field results from a synthesis of several different areas of knowledge: music, computer science, cognitive science; but also mathematics, physics, linguistics, and so forth. This obviously means that even though many different *weltanschauungen* coexist within the same field (the music-oriented, the engineering-oriented, the cognitive-oriented, etc.) and new knowledge incessantly deepens motives, goals, and methods, two fundamental paradigms differentiate all existing research: the traditional AI approach and the connectionist (also referred to as *parallel distributed processing*, *artificial neural networks*, etc.).[1]

It is well known that the traditional AI approach (Newel & Simon, 1972) aims to simulate intelligent behavior representing information through symbols, like in those computer systems (also called *expert systems*) that encode knowledge in the form of rules that are used afterward to solve specific problems. This paradigm is particularly suited for situations where it is possible to describe a problem in "procedural" terms (in other words, for situations that let us establish the better action to undertake when facing a specific condition: *if* <situation> *then* <action>). The fact that in these systems every rule represents a "splinter" of knowledge, clearly separated from the others, makes it relatively easy to manage the whole system's complexity. Nevertheless, the fact that this approach entirely depends on explicit knowledge sometimes constitutes a serious inadequacy, since a great deal of processes do not seem to be constrainable within the rigid boundaries resulting by a division into unconnected and/or antagonistic fragments (i.e., the rules). This appears particularly true when dealing with specific aspects of music cognition: In fact, while it is both feasible and reasonable to classify a chord using rules (e.g., *if* the three notes of a chord reveal a major third interval between the first and the second and a perfect fifth between the second and the third, *then* we have a major chord),

it would be almost impossible to do so for something proteiform such as musical style. Even though it is true that we could attempt to capture musical style through rules (for example, rules that describe some grammar as in Baroni, Dalmonte, & Jacoboni, 1999), it is also true that we would most certainly end up constructing a grammar so unwieldy that — aside from its interior intricateness — it would cease to serve the generalizing functions that are the definition and purpose of grammars in the first place (Buzzanca, 2002a). Besides that, it would be extremely difficult, if not impossible, to construct a grammar able to include all true instances of a style and exclude all false ones. As a result, only a small fraction of the examples produced by that grammar would be actual instances of the style which it is supposed to model. More importantly, can we positively affirm, in a cognitive perspective, that we consciously make use of rules when discerning among musical styles? Just to keep up with this example, everyone knows that musical style recognition is somehow intrinsic to human nature, and therefore even the average layperson can tell the difference between a motet by Josquin and a symphony by Beethoven, even though he or she has never received any musical education. This clearly suggests that music can expose the researcher to plenty of information in such a form that makes it difficult (or even impossible) to elaborate symbolic descriptions, especially in a perspective of cognitive verisimilitude. Therefore, the need for mimicking specific aspects of musical cognition (aspects that rules cannot efficiently unravel) is to be considered one of the reasons that has determined, over the recent years, a growing interest in the connectionist approach, together with the fact that several cognitive processes have been proven (resembling under this point of view musical style recognition or the ability to play music) to fall outside of our conscious awareness (Velmans 1991). It is well known that artificial neural networks (McClelland & Rumelhart, 1986) — albeit in a very broad sense — emulate the structure and functioning of the brain, introducing a whole new way to tackle problems by means of a massive number of elementary interconnected processors (i.e., the networks' components, the neurons) that operate in a parallel distributed way. This new approach allows for learning and generalization capabilities, that together with noise tolerance, constitute interesting aspects for the modeling of musical cognition. In fact, a large amount of musical activities in which we engage subtend the necessity to dig out implicit knowledge from examples: Epistemological problems that arise in the process of attaining knowledge (learning, constraint satisfaction, feature abstraction, intelligent generalization) are somehow embodied in the connectionist approach. The same applies to those musical tasks that require an effective low-level processing of noisy data (perception of pitch, timbre, etc.) that would be impossible to deal with by means of traditional AI methods.

The fact that ANNs do not require information to be structured in any preexisting *a priori* has already allowed researchers (as we will soon explain) to tackle a number of questions pertaining to musical cognition that proved impossible to solve in a symbolic fashion (Buzzanca, 2002). Nevertheless, this should not deceive the reader into thinking that modeling music through neural networks is a trivial task: Several questions regarding data encoding both at a micro level (melody, rhythm, melodic contour, etc.) and at a macro level ("directionality" of music, levels of hierarchy, etc.), together with several other determining parameters, make the effort of discussing about it well worth while.

Music and ANNs

The issues discussed in this chapter do not pretend to be a thorough discussion of all relevant questions involved in the field of music and neural networks; this is also a field that is at present a highly interdisciplinary one. Mathematics as well as cognitive science have all contributed to its improvement; this ultimately means that important lessons can be learned in other domains: Many valuable sources have undoubtedly been missed. Nevertheless, we do believe this chapter can be a good starting point for searching the field. Emphasizing recent publications, it includes reports of experiments, theoretical discussions, case histories, new techniques, and so forth.

As said before, the field of music and ANNs gathers the contributions of a number of different disciplines, in the attempt to accomplish a better understanding of the processes involved in music production and appreciation.

As the reader will notice throughout the next paragraphs, two different trends characterize the literature discussed here: one claiming that neural networks allow for an accurate modeling of human musical cognition and behavior (for instance Large, Palmer, & Pollack, 1999; Krumhansl et al., 2000; Bosi, Buzzanca, Buzzanca, Gargiulo, & Lamanna, 2003); the other, which could be deemed closer to the area of *machine learning*, more interested in ANNs as techniques that allow computers to learn and within reasonable terms, refrain from the need for human intervention in the analysis of the data (this is specifically the case of those studies on the automatic recognition of musical style, like Rauber & Frühwirth, 2001, for instance).

Buzzanca (2001) has shown how complex the analysis of musical style can be by means of a symbolic approach; in his work, the recognition of musical style is attempted through an expert system that implements style specific grammatical rules as its knowledge base (Baroni et al., 1999) and parsing techniques. The necessity for *ad hoc* rules (Camilleri, 1992) and furthermore the complexity of traditional parsing techniques suggest a different approach: Artificial neural networks seem particularly suitable for the solution of such problems. Artificial neural networks do not structure knowledge in accordance with any methodological "a priori." As mentioned before, they are in fact formed by a copious number of very simple and heavily interconnected processing units; each unit integrates the incoming signals (from other units) in order to output its own signal (directed to other units). This approach (massively parallel and distributed) has allowed for the solution of several inconveniences intrinsic in the symbolic approach, spacing from a higher consistency with real cognitive processes to such qualities as graceful degradation, adaptiveness, noise tolerance, and so forth (Rumelhart & McClelland, 1986); in fact, it is very difficult to treat less than perfect signals in a symbolic fashion.

Nevertheless, the most fascinating characteristic of parallel distributed systems is their ability to learn: Exposed to examples of what is necessary to learn and using special algorithms that modify the strength of the single connections among the units, such systems can learn a wide variety of very complex phenomena (recognition of images, attribution of meaning to the words, etc.). The ability to learn from experience through the modification of the connections' weights evidently brings connectionist systems near to biological ones and has to be deemed extremely important from the theoretical

point of view since it offers a convincing alternative to learning based on the construction of explicit rules. Artificial neural networks do not implement any preexisting rules or constraints. The modeling of a fact emerges as an epiphenomenon, that is, in a description of a more abstract level, while at a lower level, the whole information is stored in the connections of the units.

This is not that much of a paradox, as we will see; these systems are able to generalize (i.e., to manage) cases on which they have not been trained before. It is well known that a neural network trained to recognize handwritten characters will be able to recognize (albeit within reasonable limits) handwritten characters it has never seen before. In other words, what makes these systems particularly suitable for their employment in the field of music research is their ability to accomplish those very tasks that require the simultaneous consideration of a massive number of variables.

Early Literature on Music and ANNs

The first applications of neural networks in music research — in the role of systems able to process information introduced in sub-symbolic form — can be situated in the late 1980s, following the general enthusiasm which accompanied the neural net revival by Rumelhart and others.

The advantage of some of those papers was the attention to the cognitive perspective, which contributed in giving a better explanation of why connectionist approaches had to be deemed valuable in the field of music research.

Nevertheless, the number, relevance, and richness of the application of such systems to music can be seen on the rise starting from the second half of the 1990s. The earlier research (particularly the one concerned with the low-level matter, like for example how to encode music in a network, Linster 1989), whilst sometime in a vague and teleologically unclear fashion, would constitute the foundation for the development of further and more complex theories and techniques that characterize breath and depth of later research (from the mid to late 1990s to the present day), including the attempt to verify the validity of such theories and techniques at a cognitive level (like in Krumhansl et al., 2000).

Thus, from the initial spark of inspiration, a very rich and articulated approach has flourished through the years, which encompasses the analysis and modeling of very complex musical phenomena (for example, the research on the cognitive aspects linked to the recognition of musical style), for which important confirmations come from the fields of neurophysiology and neuroimaging (see, for example, Janata & Grafton, 2003).

It is useful to face this discussion to understand aspects of feasibility, limitations, and underlying motivations of the choice of specific approaches.

General indications on connectionist models in music are found in Bharucha and Todd (1988), Leman (1988), Lischka (1991), and Loy (1991). Bharucha (1988) describes various models of musical cognition, including a constraint satisfaction network with which he attempts to model tonal harmony. Barucha uses error backpropagation of error to learn sequential musical schemas and specific musical sequences. A good literature review of the trends of these first years can be found in Leman (1988) who also shows how much

musical information can be implemented and processed by a neural network together with various precepts concerning the applications of neural networks in the field of music research (Leman, 1989).

More recent trends are summarized by Lischka (1991), together with the statement that PDP systems allow an alternative and more biologically oriented approach. Todd and Loy (1991) have put together an anthology that, on one hand, discusses several computer music issues and, on the other, shows the influence of connectionist theories on them.

Research on the perception of harmony or tonality has been tackled during those years by several researchers. Leman (1990, 1991) delineates a model for the study of the ontogenesis of tonal functions, representing in a distributed fashion the perception of chords according to Terhardt's (1982) theory of tonal perception.

In this research, chords are considered as static, avulsed from time; later on, Leman (1991) adopts a dynamic approach instead: A series of musical data are fed to the network respecting the precise temporal sequence. This approach has been covered again by the same author in recent years (Leman & Carreras, 1996) to analyze the first book of the *Well-Tempered Clavier* by J.S. Bach.

Connectionist models for the recognition of timbre have been developed by De Poli, Prandoni, and Tonella (1993) and Cosi, De Poli, and Lauzzana (1994). De Poli et al. (1993) have used a three-dimensional version of a Kohonen network to recreate Grey's timbre space. Cosi et al. (1994) have realized an auditory model (by means of a Kohonen net) to map 12 acoustic musical instruments under conditions of clear or noisy signal. The network is able to recognize noisy forms of the sounds it hears.

Desain and Honing (1989) have realized a connectionist quantizer which is able to infer the correct meter even if fed with musical inputs containing variations of the meter. Their model perceives the existing temporal values among the various note onsets so that every couple of values is compared with a standard value. Not dissimilar is the approach of Large and Kolen (1994) who have elaborated a neural network that analyzes rhythmic patterns.

On the side of music production (algorithmic composition to be more precise), Todd (1989) has used a so-called sequential network by Jordan: A multilayered neural network (with error backpropagation) supplies one note at a time. The temporal context is given by integrating the activation values of the input units. Nevertheless, the compositions produced by this network lack general structure (in the author's opinion). For this reason, in a following paper, Todd (1991) suggests the use of two interconnected networks. In the same direction, the research by Mozer (1991, 1994) uses neural networks for automatic composition. A critical perspective is found in Page (1994) concerning the preceding connectionist approaches in the field of music research: These models would not be believable models of human perception. In his research, he adopts a new approach, training a network on melodies from lullabies. Kaipaninen (1994) uses PDP models to explain the acquisition and use of musical knowledge.

Recent Literature on Music and ANNs

More recent literature benefits from the width of previous experimentation: Neural networks had already proven suitable for problems being too difficult to describe within frameworks assembling rules or algorithms. The capabilities demonstrated by the early connectionist approaches is thus extended to the modeling of musical cognition, spacing from the investigation of the cognitive processes involved in learning music to the origin of musical emotions.

Nevertheless, the modeling of musical cognition through neural networks is not the only perspective in recent literature: Several interesting clues come from studies where neural networks are used as a promising area in the domain of machine learning techniques (Buzzanca, 2003; Ponce de León, Pérez-Sancho, & Iñesta Quereda, 2004). Although apparently different, these studies share a fundamental aim: to understand and explain how music works through an in-depth exploration of its structure.

ANNs and Music Composition

Composition by means of artificial neural networks is tackled by Hörnel and Ragg (1996), Hörnel and Degenhardt (1997), and Hörnel (1998), the creator of HARMONET, a system able to compose four parts chorals. The term chorale is used to indicate music relating to Lutheran church (or, more generally, to Protestant churches). The necessity believed by Luther to confer a more modern appearance (in comparison to the Italian Renaissance Polyphony) and to make it easier for people to sing and follow music when attending the mass or a religious service, is at the base of a new formal organization, characterized by simple contrapuntal structure and harmony. Despite the incredible artistry accomplished by Bach (as well as a few other baroque composers), the chorale remains extremely regular from a perceptive point of view, making it suitable for experimenting. According to Hörnel, the design of a network able to reproduce musical style has to act in accordance with some important facts: (1) Networks contemplating a fixed number of features have problems with learning musical style; (2) Nets of small dimensions are simply incapable to learn; (3) Nets of large dimensions tend to learn 'by heart' (i.e., not giving certainty that a real generalization process has taken place).

A net only considering a fixed number of characteristics is unable to comprehend which features have to be taken into account for the replication of that style. The *corpus* studied by Hörnel is the baroque chorale (Bach, Pachelbel). The solution proposed is equally concerned with effective learning and cognitive questions. Since the task of harmonizing a melody is very complex, it is decomposed in sub-tasks; express rules take care of specific aspects deemed not dependent from the style, like the *ambitus* or the presence of parallel fifths. The creative part is submitted to three different neural networks, n.1 to learn the harmonic functions, n.2 to correctly detect the bass note, and n.3 to learn the ornamentations that usually enrich baroque music (specifically melody).

The entire corpus is analyzed within windows of given length; more specifically, to learn harmony, data concerning the harmonic sequence for the three preceding quarter notes

at time *t* (i.e., harmony *H* at time *t-1, t-2* and *t-3*) are fed to the network together with the melodic context (i.e., melody M at time *t-1, t* and *t+1*) while, as already mentioned, two other networks manage the bass note and the melodic ornamentations.

The question relative to the dimensions of the network is tackled, implementing an evolutionary algorithm for neural network optimization (ENZO based on the principle of the survival of the fittest): Networks configuration giving the best results are maintained, while nets giving bad results are suppressed. Professional musicians have judged the chorals composed by HARMONET stylistically convincing. Hörnel is one of the most fecund researchers in this field (Hörnel, 1998).

A connectionist framework is also utilized by Bosi et al. (2003) to investigate prediction and completion of musical sequences excerpted from J.S. Bach's chorals and compared to human experts' performance (composers). The authors account for: (1) consistency with Bach's musical style; (2) variance with respect to the Neural Network's output. The corpus considered includes a significant number of chorales in 4/4 time. In almost all of them, the first quarter note coincides with the strong beat; sequences of more than 14 eight notes are never present (chorale BWV 248). In that *corpus* peculiar melodic ornamentations are present (like the *volta,* for example) which can be ascribed to the occurrence of a dissonance. Harmony is plain: Some chordal paths that may seem complex at first sight are in fact a distinct succession of cadences, as Steinitz (1991) has already pointed out. In their experiment, the authors have provided each human expert with a copy of an excerpt (bar 13 through bar 16) of the chorale *Es woll uns Gott genadig sein* (BWV 311). Experts also have been requested to complete that fragment, while unaware both of the composer and kind of piece of music. Results are quite interesting: All experts have accomplished the task in a way which is coherent with Bach's style. However, the authors point out some specific traits that seem sometime stylistically inappropriate: wrong voicing, wrong harmonization, wrong final cadence, and a whole phrase which sounds in the "ancient" chorale style (that is homophonic and homorhythmic). In this last case though, the original melody was actually taken from the *Geistliche gesang Buchlein*, the anthology of chorales (1524) selected by Luther and Walter and therefore closer to the popular style of the late musical Renaissance (distinct delimitation among phrases, etc.). The authors consider the task of melodic completion in terms of prediction of future melodic and harmonic events as a result of the analysis of available pertinent data (i.e., the fragment by Bach). This is a two-folded task, since it includes: (1) looking at the past, seeing which variables are significant indicators of the behavior of other variables; (2) getting a better understanding of which variables are important ones to watch as signals for further developments. Obviously, if we knew the true underlying model generating the data we observe, we would know how to obtain the best forecasts; however, the true underlying model may be too complex, even in Bach's chorales, so that we are not sure which model, among many competing ones, is the true one. Even in the case of Bosi et al. (2003), a connectionist approach has been considered suitable for this task. The authors use in their experiments a time lagged recurrent network (TLRN), which closely resembles a multi-layer perceptron extended with memory structures that have local recurrent connections. In fact, TLRNs have been shown to be appropriate models for processing time-varying information (see Principe, Euliano, & Lefebvre, 1999, p. 544). Cadences suggest the length of musical segments which are subsequently fed to the network. The path from one cadence to the other corresponds to the length of a verse,

with a few distinctions among the considered cadences: (1) *Authentic cadence* most decisively ends the last phrase of a period and is represented by the harmonic progression V-I. (2) *Half cadence* occurs in the middle of a musical period and typically ends an internal phrase. This cadence is represented by several harmonic progressions such as *I-V, IV-V, ii-V,* and *vi-V*. Its nature is arrival at the dominant. (3) *Musical period*, is a complete musical sentence. It consists of two phrases, the first leading to a half cadence, the second leading to an authentic cadence. (4) *Phrase* is a musical sentence leading to a cadence. The network is fed with the first half of a period and has to predict the second half. Prediction information concerns various parameters: (1) theme class; (2) melodic profile; (3) harmonic field. Training set consisted of phrases excerpted from the 348 four voiced chorales by J. S. Bach. After training the network, the authors tested the network performance on a 20% of data that the network was not trained with. Straight backpropagation could not be used for training, since recurrent networks must be trained using a dynamic learning algorithm. The backpropagation through time (BPTT) algorithm was used (Principe et al., 1999), which is quite complex and has the disadvantage of requiring a lot of memory. To account for the special nature of musical input, the authors provide a recursive memory of the input signals past. Therefore, each memory tap within the data vector is computed by simply copying the value from the previous tap of the delayed data vector. This means that a given tap within data vector is computed by taking a fraction of the value from the previous tap of the delayed data vector and added with a fraction of the same tap. The first unit of each channel is simply the channel's input and is not modified. Albeit the authors consider their effort just a preliminary study, results seem to show that for humans, it takes much less effort to become skilled in some specific musical style (Bach's style in this case) in connection with forecasting/completion tasks. An important question that the authors pose refers to the fact that with what would be considered an incredibly scarce training set for any network (only two chorales), experts were able to accomplish the task in a stylistically coherent fashion, literally dwarfing networks that still need large training sets and long computational times.

Jordan-Elman networks are used by Gang, Lehman, and Wagner (1998) to solve a problem which presents some analogies with the work of Hörnel (1998) and Bosi et al. (2003). They implement context units with local feedback to provide memory and store the recent past. Since the feedback is usually fixed, they can be placed anywhere in multi-layer perceptrons without changing the feedforward nature of the learning. The authors connect context units to the input layer of a regular MLP to provide a memory of the recent input data. The memory depth in each context unit is adjusted through its feedback gain, or time constant. In this specific case, Jordan's nets are used for harmonization tasks: The Authors pose a very interesting question in relation to harmonization tasks (i.e., first sight harmonization). Thus, this is not trivial question, since there is a number of tasks — in which musicians are concerned — where it is necessary to improvise assigning chords to some bass melodic line. One example is the *basso continuo* praxis. The *continuo* (also called *figured bass*) is in fact a system of partially improvised accompaniment played on a bass line, usually on a keyboard instrument; it was customary during the 17th and 18th centuries and gave considerable flexibility to the keyboard player, usually an organist or harpsichordist, in assigning the chords.

Sight harmonization (that is extemporaneously setting chords while sight-reading a melody) is a complex task; a prerequisite to sight harmonization is a knowledge of the

fundamentals of musical structure. In the case of a jam session for example, the musicians share a certain amount of information relative to the harmonic paths to follow. This means that the construction of a human model for real-time harmonization of unfamiliar melodies has to deal with both sequential and temporal information. A prerequisite will be awareness of the location in the metric hierarchy and structure, harmonic and melodic expectations, and reduction. The authors implement an algorithm of backpropagation and a sub-net supplies information corresponding to the meter. The net also learns the sequence of chords in relation to the melody notes and their metric position. The input layer receives data concerning the melodic line, while the output layer outputs the chord type (it embodies 14 chords: 7 major triads — triads in tonal music are defined as three-note chords that have a root pitch, a third above the root, and a fifth above the root — and 7 seventh chords — seventh chords are merely triads with the note an interval of a seventh above the root added). One of the hidden layers represents the chromatic pitches and is connected to the output layer so to create a sort of constraint between the note and the chord that is being assigned. The authors trained the network with 18 melodies and report successful learning; they also believe that modeling musical cognition through networks can shade light on how musical knowledge is exactly used by a musician when sight harmonizing a melody.

ANNs and Music Analysis

ANNs for the Modeling of Musical Cognition

One more possible employment of ANNs (as briefly mentioned in the introductory paragraph) is investigated by those scholars interested in using them as a platform on which to study and build models of human cognition, albeit they do not necessarily correlate directly with how humans might produce intelligence. An interesting example of cognitive studies that employ ANNs to model human cognition in a way that is concrete and real is accomplished by Large, Palmer, and Pollack (1999) and tackles the problem of variation. Not to be considered restricted to the field of music, the problem of variation is deemed particularly important to cognitive science on the whole, in view of the fact that it seeks to grasp complex relationships between the species' abilities and culture in the broad sense of the term. The hypothesis drawn from the authors is that listeners produce internal representations of musical sequences; in doing so, they implicitly evaluate the structural importance of events, ascribing more significance to certain events and less to others; this is functional to the relationships that listeners hear, for example, between a musical melody and its variations. This implies that in some way a listener is able to extract the relevant features of music by mining through different levels of musical information. Schenker's theory (1956), for example, is a reductionist theory of music since it suggests that we can keep stripping away "layers" of the music in order to find diminutions that span larger and larger sections of a piece; Schenker's model distinguishes three different layers (or levels) of a musical structure. Reductionist theories are useful to explain musical variation since they postulate the existence of some kind of similarity of the underlying structures in related melodies. This very procedure

should allow listeners to grasp statistical regularities of the particular musical culture or style in question (Knopoff & Hutchinson, 1983; Palmer & Krumhansl, 1990).

A few important considerations are needed at this point: the fact that reduced descriptions require knowledge of the relative culture and style means that there is to account for a significant learning component; listeners catch regularities listening to musical patterns and the subsequent cognitive representations determine the fact that listeners abstract and store prototypes of musical styles, that lead to musical expectations. For this reason, the authors propose a model capable of producing reduced memory representations for music, based on sequences that afterwards model reduced descriptions. This is attained by recursive distributed representations, a connectionist formalism that allows the representation of symbolic data in the form of patterns of activation of a network. An important element, which emphasizes the intention of the authors to use connectionism to model cognition, is the "empirical" component of their research, in which variations realized by human experts are analyzed and compared with predictions based on reductionist principles. A neural network (trained to produce recursive distributed representations for that same group of melodies) exhibits a form of learning and is believed to model the mechanism that, on one hand, allows listeners to acquire knowledge through the exposure to music and, on the other, allows them to assemble reduced memory representations for musical sequences.

ANNs for Recognition of Music Genre and Musical Style

The subject of stylistic recognition by means of neural networks deserves special mention because it constitutes a valuable effort of theoretical interpretation of a phenomenon that musicology is still unable perfectly to define.

The process of stylistic recognition can be considered as a process of supervised learning, that is, a listener can be given several pieces of music in a specific style (the examples), together with other pieces that cannot be ascribed to the same composer or age; this last one can be thought of as the non-examples.

It is well known that neural networks do extremely well with supervised learning: A network can be fed with passages by specific composers (or even *genres* as we will see) and trained to correctly classify examples and non-examples (Buzzanca, 2002a).

If the learning process has been successfully completed, the network is able, in this phase, to discern the music of a composer from those of other composers (or pieces of music of one *genre* from those of other *genres*, Ponce de León et al., 2004). Another advantage which is worth mentioning is that ANNs allow for detection of profound structural constituents and consequent classification; they also outperform traditional AI approaches as we will discuss in-depth in the following paragraphs (see Buzzanca, 2002). Research seems focused along two principal routes: (1) recognition of different genres (i.e., jazz from classical); (2) recognition of different styles.

Birmingham et al. (2001) have attempted the recognition of musical style: The authors criticize that previous approaches have almost always considered the encoding of low level elements (e.g., pitches and durations). High-level elements have in a good number

of cases been disregarded, since they may seem both highly unstructured and rather chaotic (e.g., if MIDI encoding is used; MIDI is the acronym for *Musical Instrument Digital Interface*, see Selfridge-Field, 1997). The authors experiment with the recognition of four different performing styles (e.g., *lyrical, frantic, syncopated,* and *pointillistic*). Their objective is to have the machine correctly recognizing several chunks of music, each chunk performed in a different style. Music samples are played by human performers that read on a screen the style they have to conform to. In other words, the performer has to play in the performing style which is displayed on a monitor; if *lyrical* is displayed on the monitor, he or she will play the music in a lyrical fashion; if the monitor suddenly displays a different style, *frantic* for example, he or she will turn his or her performing style into frantic and so on. Afterwards, the various samples are submitted to the performer so that he or she can confirm that the piece of music being heard right then is indeed performed in the style it was intended to be played. Since the changes in performing style are displayed on the screen without giving any notice to the performer, the first four seconds are discarded while the following ones are retained to ensure that the sample reliably represents the corresponding performing style.

The authors employ a training set consisting of 1,200 seconds which is afterwards fed to the machine, in MIDI format. The machine considers 13 low-level parameters (pitches, durations, volume, etc.). The results of different classification methods are compared (Bayes, linear classifier, neural network). Results from the network outperform the others, showing that neural nets can be used to construct reliable style classifiers. The conclusions drawn by the authors suggest that machine learning techniques and, specifically, neural networks succeed well where other approaches have failed because they are able to take into consideration a vast number of multi-dimensionally different parameters (the authors consider 132 different characteristics).

Special Issues About Music and ANNs

The Encoding of Music

An important topic is how to encode music: A network can, in fact, be fed either with some sound wave (in this case, it would be necessary to identify which of the various components of physical sound are critical to recognition) or with some high-level, structured, and exact representations of the music (see Buzzanca, 2002b). In any case, a nontrivial matter is segmentation of musical phrases, because that too has been shown a crucial element (see Buzzanca, 2002b). Carpinteiro (1995) proposes three different ways to segment music: one that locates the caesura in correspondence with a longer duration within a group of notes; another that locates it where pauses are found; and the last one places the caesura within a group of notes whenever the two subgroups in which the group can be segmented present with equal durations in each subgroup and at the same time different durations between the first and the second subgroup (e.g., group of notes consists of six notes, three semi-quavers, plus three quavers; in this case, the caesura goes right after the third note).

Once a segmentation setting has been decided, another important element is the encoding of the remaining parameters (pitches, durations, etc.); Carpinteiro differentiates among sounded note (note is actually played), tied note (note has a pitch but is not actually played since it is the prolongation of a preceding note), and pause (no sound at all), encoding them as *11, 01, 00,* respectively; in his experiment, each part of the *Two part inventions* by Bach is implemented as a single melodic line (the network works "monophonically," i.e., one melodic line at a time rather than polyphonically). Carpinteiro's research confirms that a neural network which implements supervised training can effectively learn how to segment musical phrases.

Ponce de León and Iñesta Quereda (2003a) and Ponce de León et al. (2004) have investigated the recognition of musical genre. Since the two chosen genres can be very different one another (jazz and classical), a slight perplexity on the genuine degree of intricacy of the classification task deserves in our opinion to be further investigated in this case; nonetheless, several elegant solutions make their work valuable. The authors implement the scores in MIDI standard format. The corpus includes 110 multitrack MIDI files (i.e., files that contain multiple voices), from which the melody is extracted; this way, the analysis is functional to the single voice that has been previously avulsed. The authors use "descriptors" that are inferred from "windows" of appropriate width. Albeit in their first paper, Ponce de León and Iñesta Quereda (2003a) do not use neural networks, there are two elements that are worth highlighting: (1) the importance of the segmentation problem; (2) the choice of appropriate encoding for the musical data. The choice of MIDI deserves a short digression. There are multiple ways for the encoding of music on the computer (Selfridge-Field, 1997); nonetheless, the encoding method has to be carefully chosen in connection with such aspects as efficient storage and high parsability. In our opinion, a first distinction is *de rigueur*: Production and analysis of music behave differently in connection with the encoding technique which is chosen. MIDI, for example, does not actuate a bijective correspondence between numbers and notes: Number 60, for instance, identifies middle C (number 61 identifies C sharp and D flat, 62 D, 63 D sharp, and so on); but at the same time, since there are notes that are written differently (e.g., B flat and A sharp) but are enharmonically the same note in the tempered scale, MIDI does not allow to uniquely identify a note. As a result, if the already mentioned MIDI number 60 is passed to the computer for playing a note it will play middle C. On the contrary, if the computer reads number 60 because we want him to identify the corresponding pitch, it becomes impossible to uniquely classify it because no one would be able to positively affirm if that number 60 stands for middle C or B sharp or D double flat. It is obvious that while, on one hand, there are no consequences if playing MIDI music, analysis can be negatively affected by the consequent misrepresentation of intervallic width. What is the width of the MIDI interval 60 to 64? Does it stand for the major third C-E? For the diminished forth B sharp E? For the more than diminished fifth B sharp F flat? It is not known if this can alter the results of analysis; we have partially demonstrated that it can (Buzzanca, 2002).

Turning back now to the issue of segmentation, we have to express perplexity for the method of windows of predetermined length (Ponce de León & Iñesta Quereda, 2003). While it could be acceptable for machine learning applications, it seems to us that it is improper for the construction of connectionist systems that aspire to model musical cognition. In fact, segmenting music into fixed width chunks constitutes a simplification

in comparison to real life, since the "amputation" of musical stimuli into sequences of predetermined length is not realistic neither at a perceptive nor at a theoretical/ compositional level (see Sloboda, 1985; Smoliar, 1992).

A rather complete discussion of these issues can be found in Buzzanca (2002), who has designed a neural stylistic recognizer implemented using a backpropagation neural network, with a number of innovative techniques to account for the special nature of musical input. Buzzanca's technique makes no assumptions about the period or style of music being studied, but his experiments have focused on Palestrina and polyphonic music from the Italian Renaissance. The network model is designed with three main purposes: to account for musical input of varying length in an even manner; to model the hierarchical nature of musical recognition; and to capture the importance of directional flow in music. At the heart of Buzzanca's method is the technique of *shared weights* developed by LeCun (1989). The idea is quite simple: In an ordinary neural network, there is a separate learnable weight associated with every edge between two nodes. Using shared weights, many of these weights are forced to be the same, in a symmetric way (see schema in Figure 1).

Thus, all edges belong to a certain type, and weights are associated with types of edge rather than individual edges. An example of a type of edge in the network is the edge from the node, representing the interval from the second to the third beat in a measure, to a node summarizing the entire measure. This edge will have the same weight for all measures. However, the edge from the node represent the interval between the first and second beats to the node for the whole measure will be different. Shared weights have several advantages. As seen in the previous example, they capture the fact that different edges represent the same concept, so the learning is shared between all those edges. Furthermore, it is well known that the sample complexity of a neural network, therefore, the number of training examples needed to learn accurately is closely related to the concept of *VC-dimension* (see Vapnik, Levin, & LeCun, 1994), which depends on the number of different weights. Shared weights greatly reduce the total number of different weights, hence, the amount of training data needed. This is important, because a rather complex network structure is required to represent the subtleties of musical style, but there is not enough data to learn an ordinary network of sufficient complexity. The shared weights technique also deals with the issue of variable input size. Simply put, while the number of edges is allowed to vary with the size of the input, the number of edge types — and, therefore, the number of phrases — is constant. Given a musical phrase of a certain length, the author simply grows a network of sufficient size to represent the entire phrase. The same thing will be learned for phrases of all different types, namely, the weights associated with each of the edge types. Thus, the ellipsis in the diagram can be filled with any number of measures, all with similar structure. To get at the multi level analysis, Buzzanca uses multiple hidden layers, each summarizing the previous level. The first hidden layer summarizes the information in a single beat. The second layer takes the output of the first layer and summarizes two beats. Note that the second layer actually has four inputs. That is because in the first layer two patterns are present: on-beat to off-beat and off-beat to on-beat. Similarly, at the second layer, strong-weak patterns and weak-strong patterns are seen. The theory is that a particular style might be characterized by what happens in either style. The hierarchy continues in a similar way: The third level

summarizes four beats, and so on. In his experiments, Buzzanca limits the network to three levels. To get at the directionality of music, the author took the output of the highest level of hierarchy and fed it to a directed network. The directed network consists of a sequence of nodes, from the beginning of the music until the end, where each node depends on the previous node in the sequence and on the corresponding top-level node in the hierarchy.

The diagram in Figure 1 is only a schema of the actual network. Each node in the diagram actually represents a set of nodes. Hidden layers typically use between two and six nodes at each point; for example, there may be four nodes representing the third-level strong-weak summary. Each such node has an input from all the relevant second-level hidden nodes. The input consists of four nodes for each unit of music. The unit chosen is half a beat (i.e., an eighth note for music in 4/4 time).[2] This is the smallest unit that occurs frequently in Palestrina, but the method adapts easily to smaller units. Although 16th notes do occur occasionally in Palestrina, building a network for quarter beat analysis would increase the complexity too much to justify modeling these occasional events accurately. As it turns out, it is still possible to deal with 16th notes quite well, as described next.

Out of the four input units for each half-beat, two encode rhythmic information and two encode melodic information: This encoding system can simply be thought of as a sampling of music every eighth note. The rhythm encoding is a generalization of the previously mentioned one due to Carpinteiro (1995).

Figure 1. Schema of the connectionist model by Buzzanca (2002). The sign [...] reveals that the model can be grown to the desired dimension. Letter 'I' stands for input; letter 'H' for hidden; letter 'O' for output. The arrow indicates the flow of information.

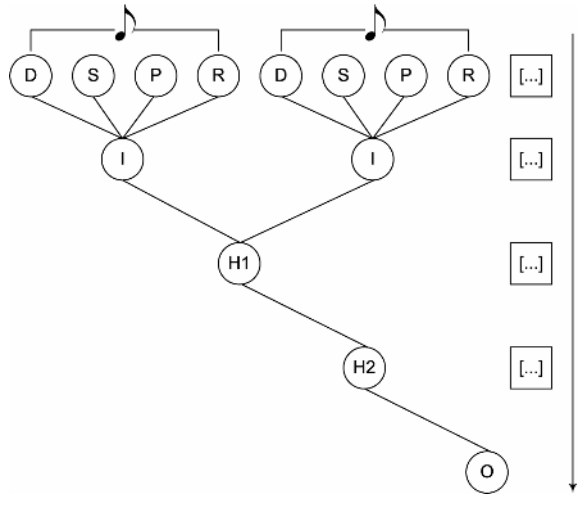
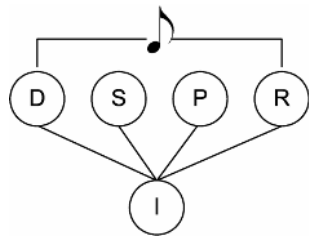


Figure 2. The four input units encode information relative to durations and pitches at an eighth note pace. The first one (D) encodes duration, the second (S) whether the note is sounded or not (i.e., if it is a note or a rest), the third (P) encodes pitch,and the forth (R) encodes range (i.e., the leap relative to the preceding note).



The first unit encodes the onset of a note. Its value is 0 if no note begins at that time, otherwise the value is the logarithm of the duration in half-beats (e.g., 1 for an eighth note, 2 for a quarter note, 3 for a half note, and so on). This representation allows us to encode duration, which is believed to be important in style identification, but in such a way as to minimize the impact of very long notes. The second rhythm unit corresponds to whether a note is sounded at a point. It is 1 for sounded notes and 0 for rests. Pitch is encoded with two nodes. The first encodes the absolute pitch in semitones, relative to the first note of the melody. The second is the relative pitch to the previous sounded note. While there is redundancy here, as either one could be deduced from the other, both are useful for style recognition. The first expresses aspects such as range and contour, while the second provides intervallic information. Indeed experience shows that representing the same data in multiple ways can be useful to neural network classifiers. The double encoding of pitch also allows us to deal with 16th notes. The first of the two 16th notes is usually the more important, so it is used to encode the pitch at the point of the 16th notes. However, for the interval input of the next note, the second of the two 16th notes is taken as the previous note, since that is the one defining the interval. In this way, it is possible to capture key aspects of both notes without having to greatly grow the network to accommodate a 16th-note scale.

Other Implementations

Self-Organizing Maps (SOMs)

The recognition of musical style has been also tackled using Kohonen self-organizing maps (SOM) (Kohonen, 1989). It is known that the SOMs are artificial neural networks that simulate the process of self-organization ascribed to the central nervous system with

a simple and effective numerical algorithm: A bidimensional array of simple processing units has a reference vector related to each unit. After training with the input vectors, the network nonlinearly maps the multidimensional input into the array, consequently placing vectors that are closer to one another in adjacent areas of the array. Ponce de León et al. (2004) have used this technique to distinguish between jazz and classical music. Once more, they encode music in MIDI format; first, the authors separate the melody line from the rest of musical information contained in the MIDI files (they keep just a sequence, either notes or rests; other MIDI events are just removed). The musical fragments are eight measures wide (in 4/4 time); the corpus includes 430 jazz and 522 classical melodic samples.

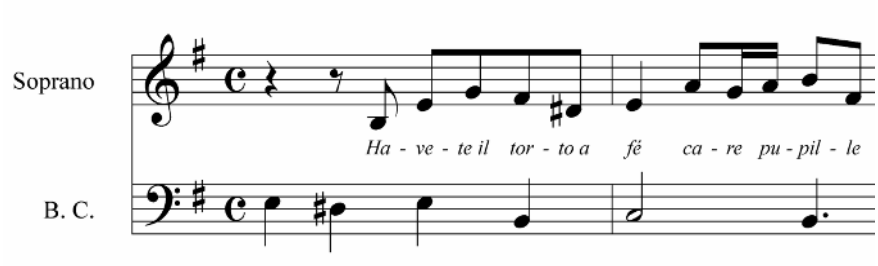
Other recent papers investigate the SOMs' ability in the field of stylistic recognition: A paper by Rauber and Frühwirth (2001) can be located within a wide area of studies called audio signal classification (ASC). ASC aims at the mining of relevant features for classification purposes (see, for example, Gerhard, 2003). The automatic classification of music into genres is becoming more and more important since there is an immense and relentlessly increasing amount of music available on the Web, which is the practical reason besides the "nobler" ones connected to the elaboration of realistic models of human cognition. To understand the proportions of the subject, it suffices to say that *mp3.com* keeps a record of 350 different musical genres.

Rauber and Frühwirth (2001) have implemented a model that can recognize a melody hummed by the user: This is not a trivial problem, since each user can potentially hum the melody in a different tonality (something similar happens with learning shift invariance with networks); transposition can thus hinder recognition. Music passages are divided into five seconds chunks; all beginnings and all ends are discarded to prevent fade in and fade out effects to impede recognition. Afterwards, they are fed to the SOM and distributed according to their degree of similarity.

Table 1. Classification results (success rates in percentages) for Ponce de León et al. (2004). The authors use a set of melodic, harmonic, and rhythmic descriptors. Results show a difference in performance with regard to the number of descriptors used. "Best" results for "Both" styles are averaged for jazz and classical styles. In the second row, the 16 x 8 and 30 x 12 refers to the map size.

	JAZZ				CLAS				BOTH			
	16x8		30x12		16x8		30x12		16x8		30x12	
Descr.	BEST	AVG.	BEST	AVG.	BEST	AVG.	BEST	AVG.	BEST	AVG.	BEST	AVG.
All	89.8	72.7	87.8	61.2	93.2	79.6	85.1	70.8	90.8	76.1	80.7	66
6	98	79.4	81.6	68.4	95.2	82.1	90.5	89.3	92.5	80.8	78.3	73.3
7	96	81.8	83.7	74.1	97.3	86.5	97.3	76.6	96	84.2	85.1	75.4
10	98	78.8	87.8	63.3	96	82.7	90.5	74.6	88.8	80.7	89.2	68.9
13	87.8	72	89.8	67.1	97.3	82.6	85.1	68.8	84.4	77.3	78	68

Figure 3. The aria is segmented in verse long phrases (i.e., musical phrases as long as the verses of the poem which was put into music). In this specific aria (n. 2), the first verse in the poem is Havete il torto a fè care pupille ("You are very wrong dear eyes"); thus, the segment fed to the network ends on the eighth note corresponding to the last syllable of the word pupille (i.e., '-le'). The length of the second musical phrase to be fed to the network will consequently be determined by the second verse and so forth..



Toiviainen and Eerola (2001) also have used an SOM to distinguish between folk songs from Germany and Northern China. The authors conclude that one possible application of their method is its employment to detect stylistic differences/similarities from regions of the world quite distant between them, to make this information available when creating hypothesis for cross-cultural comparisons.

We close this discussion quoting Birmingham et al. (2001): Much finer classification is desirable in the recognition of musical genre and musical style with neural networks. Nonetheless, light has been shed and new techniques have become available thanks to the growing interest in this field.

Figure 4. Recognition is split into two distinct subtasks (i.e., learning the soprano and the BC).

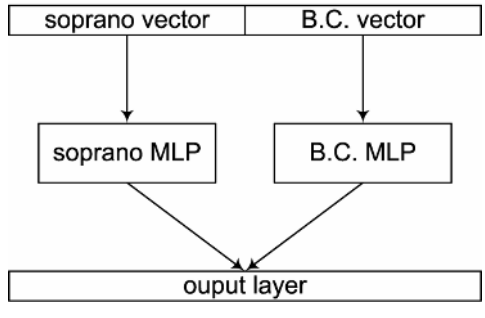


Table 2. Stylistic recognition: classification results (success rates in percentages). Results in this multiclass example are obtained setting aside 20% of exemplars for testing.

Composer	(%)
GABRIELLI	98.5
LEGRE	88.2
LEGRENZI	72.2
ROSSI	96.2
SCARLATTI	84.2
STRADELLA	89.1

Modular Neural Networks

Buzzanca (2003) has implemented a stylistic recognizer that uses a special class of multi-layer perceptrons (MLPs), also called modular neural networks (MNNs). These kinds of networks process their input using several parallel MLPs, and then recombine the results. The corpus analyzed includes several *arie da cantata da camera* by Giovanni Legrenzi (1626-1690) and a few other coeval composers: Alessandro Stradella (1639-1682), Alessandro Scarlatti (1660-1725), Luigi Rossi and Domenico Gabrielli (1651-1690), plus the automatic composer LEGRE (Baroni et al., 1999).

Buzzanca has already shown (2002a, 2002b) that shared weights technique greatly reduces the total number of different weights needed (therefore, the amount of training data). Thus, it is not a trivial improvement to design network models that do not learn by individual weight updating, but rather resolve the competition and the cooperation of the different modules. The implementation of the model is based on the *divide et impera* method: A task is broken down into smaller and less complex subtasks. This results in different experts (i.e., NN modules) learning each task. The learning achieved by each sub-module can than be re-applied to solve the whole problem. Each piece of music is segmented into several different pieces (called phrases); since the importance of the text is critical in this kind of music, each composition is segmented in phrases that have the same length of a verse (see Figure 3).[3]

Recognition is then split into two distinct subtasks: recognizing the Soprano and the B. C.; each task is trained off-line and later integrated in the global architecture according to the following schema (Figure 3). A piece of music like the one shown in Figure 3 is implemented so that the upper staff is put side by side with the lower staff; thus, we have two resulting vectors each corresponding to a staff (Figure 4)

From the schema (Figure 3), it is possible to see that each task is trained off-line and later integrated in the global architecture. This enables an acceleration of the learning; input layer has a sufficient number of input nodes, each one encoding the duration of an eight note, plus further information about pitch and duration. The output nodes are divided

into two subsets of nodes each responsible for the soprano task and the continuo task, respectively. Regarding the architecture of the modular network, the hidden layer is split in two and locally connected with the *soprano* output nodes and the *continuo* output other nodes. The best results are obtained when hidden nodes are used to solve locally the soprano subtask and hidden nodes to solve the continuo subtask. Those results are shown in Table 2.

Conclusions

We have shown that neural networks can perform well in complex tasks in the domain of music; also, they seem capable of inducing the hidden features of a domain that may be elusive to a rule based approach.

The flexible design that accompanies neural networks allows the encoding of musical structure in complex, yet powerful, models.

They have been shown to be equally successful in a number of musical tasks, such as music composition and improvisation and music analysis, including such tasks as the recognition of music and the modeling of musical cognition.

Acknowledgments

To my wife, Alessia, who has been a partner, friend, and encourager; to my daughter, Diletta, who has expressed pride in her dad, spurring his desire to write; last, but not least important, to my parents, Gianni and Anna Buzzanca, to whom I owe heartfelt thanks for their love, patience, and support over the years.

References

Baroni, M., Dalmonte, R., & Jacoboni, C. (1999). *Le regole della musica. Indagine sui meccanismi della comunicazione.* EDT.

Bharucha, J.J., & Todd, P.M. (1988). Connectionist learning of schematic musical expectancies. *Bulletin of the Psychonomic Society, 26*(6), 496.

Birmingham, W.P., Dannenberg, R.D., Wakefield, G.H., Bartsch, M., Bykowski, D., Mazzoni, D., Meek, C., Mellody, M., & Rand, W. (2001). MUSART: Music retrieval via aural queries. *Proceedings of ISMIR 2001.* Bloomington, IN (pp. 73-81).

Bosi, A., Buzzanca, A., Buzzanca, G., Gargiulo, A., & Lamanna, M. (2003). Melodic prediction with time-lagged feedforward networks. *Proceedings of the 3rd Conference Understanding and Creating Music*, Caserta, Italy (pp. 31-33).

Bresin, R. (2000). *Virtual virtuosity. Studies in automatic music performance.* Ph.D. thesis. Kungl Tekniska Högskolan.

Buzzanca, G. (2001).A rule based model for musical style recognition. *Proceedings of the 3rd Conference Understanding and Creating Music*, Caserta, Italy.

Buzzanca, G. (2002a). A supervised learning approach to musical style recognition. *Additional Proceedings of the Second International Conference ICMAI 2002*, Edinburgh, Scotland.

Buzzanca, G. (2002b).A neural network model for musical style recognition. *Proceedings of the 2nd Conference Understanding and Creating Music*, Caserta, Italy.

Buzzanca, G., (2003). Uncovering musical style through modular feed-forward networks. *Proceedings of the 3rd Conference Understanding and Creating Music*, Caserta, Italy (pp. 48-52).

Camilleri, L. (1992). Computational theories of music. In M. Leman (Ed.), *Computer representation and models in music*. San Diego, CA: Academic Press.

Carpinteiro, O. (1995). A neural model to segment musical pieces. In E. Miranda (Ed.), *Proceedings of the Second Brazilian Symposium on Computer Music, Fifteenth Congress of the Brazilian Computer Society* (pp. 114-120). Brazilian Computer Society.

Cosi, P., De Poli, G., & Prandoni, P. (1994). Timbre characterization with mel-cepstrum and neural nets. In *Proceedings of the 1994 International Computer Music Conference* (pp. 42-45). San Francisco: International Computer Music Association.

De Poli, G., & Tonella, P. (1993). Self-organizing neural network and Grey's timbre space. In *Proceedings of the 1993 International Computer Music Conference* (pp. 260-263). San Francisco: International Computer Music Association.

Desain, P., & Honing, H. (1989). The quantization of musical time: A connectionist approach. *Computer Music Journal, 13*(3), 56-66.

Gang, D., Lehman, D., & Wagner, N. (1998). Tuning a neural network for harmonizing melodies in real time. *Proceedings of the International Computer Music Conference,* Ann Arbor, Michigan.

Gerhard, D. (2003). *Audio signal classification: History and current techniques.* Technical Report TRCS2003 07.

Hörnel, D., & Degenhardt, P.(1997). A neural organist improvising baroque-style melodic variations. In *Proceedings of the 1997 International Computer Music Conference* (pp. 430-434). San Francisco: International Computer Music Association.

Hörnel, D., & Ragg, T. (1996). A connectionist model for the evolution of styles of harmonization. In *Proceedings of the Fourth International Conference on Music Perception and Cognition* (pp. 219-224).

Hörnel, D., & Ragg, T. (1996) Learning musical structure and style by recognition, prediction and evolution. In D. Rossiter (Ed.), *Proceedings of the 1996 International Computer Music Conference* (pp. 59-62). San Francisco: International Computer Music Association.

Hörnel, D. (1998). A multi-scale neural-network model for learning and reproducing chorale variations.*Computing in Musicology*, *11*.

Janata, P., & Grafton, S.T. (2003). Swinging in the brain: Shared neural substrates for behaviors related to sequencing and music. *Nature Neuroscience*, *6*, 682-687.

Kaipainen, M. (1994). *Dynamics of musical knowledge ecology. Knowing what and knowing how in the world of sounds.* Ph.D. dissertation, University of Helsinki, Finland.

Knopoff, L., & Hutchinson, W. (1978). An index of melodic activity. *Interface, 7,* 205-229.

Koelsch, S., Gunter, T.C., Cramon, D.Y., Zysset, S., Lohmann, G., & Friederici, A.D. (2002). Bach speaks: A cortical "language-network" serves the processing of music. *Neuroimage*, *17,* 956-966.

Koelsch, S., Kasper, E., Sammler, D., Schulze, K., Gunter, T., & Friederici, A.D. (2004). Music, language and meaning: Brain signatures of semantic processing. *Nature Neuroscience, 7*, 302-307.

Kohonen, T. (1989). *Self-organization and associative memory*. Springer Verlag.

Krumhansl, C.L., Toivanen, P., Eerola, T., Toivianinen, P., Järvinen, T., & Louhivuori, J. (2000). Cross-cultural music cognition: Cognitive methodology applied to North Sami yoiks. *Cognition, 76,* 13-58.

Large, E.W., & Kolen, J.F. (1994). Resonance and the perception of meter. *Connection Science, 6,* 177-208.

Large, E.W., Palmer, C., & Pollack, J.B. (1999). Reduced memory representations for music. In N. Griffith and P.M. Todd (Eds.), *Musical networks. Parallel distributed perception and performance*. Cambridge, MA: MIT Press.

Lawrence, S., Giles, C.L., Chung Tsoi, A., & Back, A.D.(1996). *Face recognition: A hybrid neural network approach.* Technical Report, UMIACS-TR-96-16 and CS-TR-3608. College Park, MD: Institute for Advanced Computer Studies University of Maryland.

LeCun, Y. (1989). Generalization and network design strategies. In R. Pfeifer, Z. Schreter, F. Fogelman, & Steels (Eds.), *Connectionism in perspective.* Zurich.

Leman, M. (1988). Sequential (musical) information processing with PDP-networks. In *Proceedings of the First Workshop on AI and Music* (pp. 163-172). American Association for Artificial Intelligence.

Leman, M. (1989). Artificial neural networks in music research. *Reports from the seminar of musicology. Institute for psychoacoustics and electronic music*. University of Ghent.

Leman, M. (1990). Emergent properties of tonality functions by self-organization. *Interface*, *19,* 85-106.

Leman, M. (1991a). Artificial neural networks in music research. In A. Marsden & A. Pople (Eds.), *Computer representations and models in music* (pp. 265-301). London: Academic Press.

Leman, M. (1991b). The ontogenesis of tonal semantics: Results of a computer study. In P.M. Todd & D.G. Loy (Eds.), *Music and connectionism* (pp. 100-127). Cambridge, MA: MIT Press.

Leman, M., & Carreras, F. (1996). The self-organization of stable maps in a realistic musical environment. *Proceeding of The Journées d' Informatique Musicale,* Caen, University of Caen/IRCAM (pp. 156-169).

Linster, C. (1990). A neural network that learns to play in different styles. *Proceedings of the 1990 International Computer Music Conference,* San Francisco (pp. 311-313).

Lischka, C. (1991). Understanding music cognition: A connectionist view. In G. De Poli, A. Piccialli, & C. Roads (Eds.), *Representations of musical signals* (pp. 417-445). Cambridge, MA: MIT Press.

Loy, D.G. (1991). Connectionism and musiconomy. In P.M. Todd & D.G. Loy (Eds.), *Music and connectionism* (pp. 20-38). Cambridge, MA: MIT Press.

Maess B., Koelsch S., Gunter T.C., & Friederici A.D. (2001). Musical syntax is processed in Broca's area: An MEG study. *Nature Neuroscience, 4*, 540-545.

Medler D.A. (1998). A Brief History of Connectionism. *Neural Computing Surveys, 1*, 61-101.

Mozer, M.C. (1994). Neural network music composition by prediction: Exploring the benefits of psychoacoustic constraints and multi-scale processing. *Connection Science*, *6*, 247-280.

Mozer, M.C. (1991). Connectionist music composition based on melodic, stylistic, and psychophysical constraints. In P.M. Todd & D.G. Loy (Eds.), *Music and connectionism* (pp. 195-211). Cambridge, MA: MIT Press.

Newell, A., & Simon, H.A. (1972). *Human problem solving*. Prentice-Hall.

Ovans, R., & Davison, R. (1995). An interactive constraint-based expert assistant for music composition. *Proceedings of the Ninth Canadian Conference on Artificial Intelligence,* Ottawa, Canada (pp. 76-81).

Page, M. P.A. (1994). Modelling the perception of musical sequences with self-organizing neural networks. *Connection Science*, *6*, 223-246.

Palmer, C., & Krumhansl, C.L. (1990). Mental representations of musical meter. *Journal of Experimental Psychology: Human Perception & Performance, 16,* 728-741.

Patel, A.D. (2003). Language, music, syntax and the brain. *Nature neuroscience*, *6,* 674-681.

Ponce de León, P.J, & Iñesta Quereda, J.M. (2003a). Musical style classification from symbolic data: A two-styles case study. *Proceedings of the CMMR* (pp. 167-178).

Ponce de León, P.J, & Iñesta Quereda, J.M. (2003b). Feature-driven recognition of music styles. *Proceedings of the IbPRIA* (pp. 773-777).

Ponce de León, P.J., Pérez-Sancho, C., & Iñesta Quereda, J.M. (2004). A shallow description framework for musical style recognition. *Proceedings of the SSPR/SPR 2004* (pp. 876-884).

Principe, J.C., Euliano, N.R., & Lefebvre, W.C. (1999). *Neural and adaptive systems.* New York: John Wiley & Sons.

Rauber, M., & Frühwirth (2001). Automatically analyzing and organizing music archives. *Proceedings of the 5th European Conference on Research and Advanced Technology for Digital Libraries (ECDL 2001)*, Darmstadt, Germany.

Reck, M.E. (2000). *Readings in music and artificial intelligence*. Harwood Academic Press.

Rumelhart, D.E., & McClelland, J.L. (1986). *Parallel distributed processing: Exploration in the microstructure of cognition*. Cambridge, MA: MIT Press.

Schenker, H. (1956). *Der freie Satz.* Universal Edition.

Selfridge-Field E. (Ed.) (1997). *Beyond MIDI. The handbook of musical codes.* Cambridge, MA: MIT Press.

Sloboda, J. (1985). *The musical mind: The cognitive psychology of music*. Clarendon Press.

Smoliar, S. (1992). Representing listening behavior: Problems and prospects. In M. Balaban, K. Ebcioglu, & O. Laske (Eds.), *Understanding music with AI.* Cambridge, MA: MIT Press.

Steinitz, P. (1991). La musica sacra tedesca. In *Storia della musica. The new Oxford history of music.* Milano, Italy: Feltrinelli – Garzanti.

Terhardt, E., Stoll, G., Seewann, M. (1982). *Algorithm for extraction of pitch and pitch salience from complex tonal signals. Journal of the Acoustical Society of America, 3,* 679-688.

Todd, P.M. (1991a). A connectionist approach to algorithmic composition. *Computer Music Journal, 13*(4), 27-43. Also in P.M. Todd & D.G. Loy (Eds.) (1991) *Music and Connectionism* (pp. 173-189). Cambridge, MA: MIT Press.

Todd, P.M. (1991b). Addendum to a connectionist approach to algorithmic composition. *Computer Music Journal, 13*(4), 27-43. Also in P.M. Todd & D.G. Loy (Eds.), *Music and Connectionism* (pp. 190-194). Cambridge, MA: MIT Press.

Todd, P.M., & Loy, D.G. (Eds.). (1991). *Music and connectionism*. Cambridge, MA: MIT Press.

Toiviainen P., & Eerola T. (2001). Method for comparative analysis of folk music based on musical feature extraction and neural networks. *Proceeding of the III International Conference on Cognitive Musicology*, Jyvskyl, Finland (pp. 41-45).

Vapnik, V., Levin, E., LeCun, Y. (1994). Measuring the VC-dimension of a learning machine. *Neural Computation, 6*, 851-876.

Velmans, M. (1991). Is human information processing conscious? *Behavioral Brain Sci., 14*, 651-726.

Endnotes

1 From now on, we will use all those terms as synonyms; nonetheless, a slight difference exists in our opinion between the use of the term connectionism (as a paradigm in the Kuhnian sense of the term) and neural networks (as a computational technique adoptable for example for machine learning).

2 The relative duration of musical notes is expressed in terms of a simple relationship they bear to one another; each represents a duration twice as long as the next smaller note. The duration of an 8th note is twice as long as a 16th note; for each note duration, there is a corresponding symbol called a rest, which indicates an equivalent duration of silence. In the case of neural networks, the fact that the input layer usually has a fixed number of units is a serious drawback when it comes to encode rhythm. In fact, even if an hypothetical input layer can process a number of notes, totaling for example a whole note, there are many subsets that sum a whole note: eight 8^{th} notes sum a whole note as well as 16 16^{th} notes. But this ultimately means that in the former case we need only eight input units while in the latter we need 16 units. This is the reason why encoding rhythm is a pretty subtle matter.

3 An aria is a composition that combines music with the verses of a poem, so that different melodies go together with different verses. Thus, there are many ways to segment an aria: segments as long as a verse, segment 8 eighth notes long, and so forth.

Chapter XIII

Connectionist Systems for Fishing Prediction

Alfonso Iglesias, University of A Coruña, Spain

Bernardino Arcay, University of A Coruña, Spain

José Manuel Cotos, University of Santiago de Compostela, Spain

Abstract

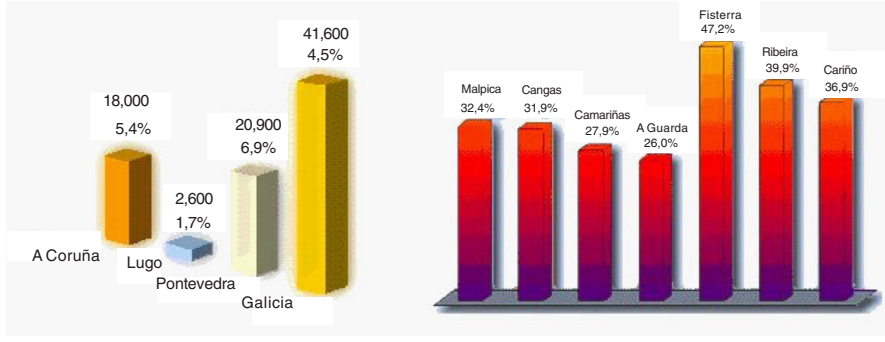
This chapter explains the foundations of a new support system for fisheries, based on connectionist techniques, digital image treatment, and fuzzy logic. The purpose of our system is to increase the output of the pelagic fisheries without endangering the natural balance of the fishing resources. It uses data from various remote sensors and the logbook of a collaborating fishing boat to improve the catches of the Prionace Glauca, a pelagic shark species also known as the blue shark.

Introduction

The Problem

Fishing, or the exploitation and optimisation of marine resources, is one of Galicia's main economical sectors and the basis of many industrial activities and services. According to the Fishing Council (Figure 1a), a total of 41,600 persons, or 4.5% of the autonomy's active population, are actively involved in the fishing sector.

Figure 1. The Galician population employed in the fishing sector. (a). The population in total numbers and in percentage of the active population for each province (Data provided by the Xunta de Galicia, 2002) (b). Relative importance of the fishing sector in various coastal areas (Data provided by the Xunta de Galicia, 2002.)



(a) *(b)*

Logically, this sector does not affect the whole Galician territory to the same extent: Places such as Fisterra, where 47.2% of the active population works for the fishing industry, are far above the Galician average (Figure 1b).

This percentage should include the jobs that are closely related to the fishing sector and increase the amount of generated jobs to a total of 120,000; This means that 12.2% of Galicia's employment is based either directly or indirectly on the fishing industry, which makes it the European region that most depends on this activity.

These data clearly show that the Galician fishing sector has grown from a local craft industry into the motor of the autonomy's economy. However, it is a sector that has recently been affected by ecological disasters, quota policies, biological stops, and the enlargement of the exclusive economical zones of riparian countries. Commercial agreements with other countries impose the import from foreign fish at very low prices. On the other hand, there is an increasing tendency in national and international organisms toward more protection and conservation of the natural resources and a sustainable development that preserves the oceans as an important provider of food for the entire world.

We believe that a fishing fleet can only stay competitive if it disposes of the most advanced technology and the largest amount of information possible. In this context, the new remote sensors are privileged tools when used as information sources of the biological conditions of the fishing environment. This work applies connectionist systems for the prediction and optimization of the available resources, based on the remotely gathered information. It is a method that enables the fisheries to reduce their

search time, save fuel, and increase the volume of the catches within legal limits. Apart from these economical motives, we also wish to put into practice new neural and neuro-fuzzy paradigms and the digital treatment of information that results from the launch of new satellites (Orbview 2).

Objectives

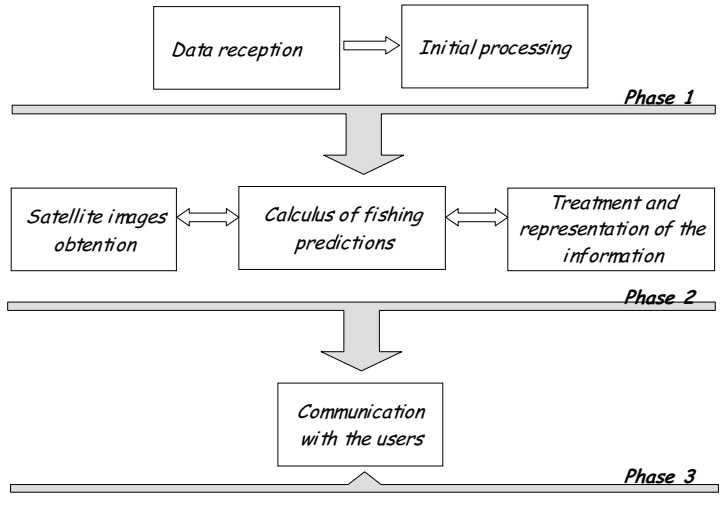
This work explains the foundations of a new support system for fisheries, which uses the data from remote sensors and the logbook of a collaborating boat to increase the catch probabilities of a pelagic shark, the Prionace Glauca, also known as the blue shark.

The developed system can be extended to any marine species that is sufficiently documented. Therefore, it is a useful tool for the optimization and management of the fishing resources, especially in geographical environments such as Galicia, whose economy clearly depends on the fishing sector. This introduction will provide economical data that demonstrate the importance of a sector that has suffered from ecological disasters and, in some cases, from an irrational over-exploitation of the available resources.

The presented system aims at increasing the benefits of its users through sustainable development.

As can be seen in Figure 2, the support system of the University of Santiago de Compostela consists of three different phases whose final purpose is to transmit those products that are apt to be sent to the boats.

Figure 2. Organigram of the support system for fisheries developed by the Systems Laboratory of the University of Santiago de Compostela



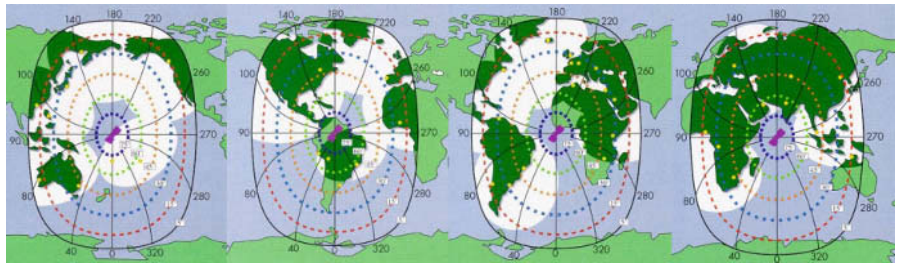
During *Phase 1*, the information that proceeds from various satellites is gathered and stored. The system consists of a set of antennas that receive information from satellites such as the NOAA (in high and low resolution), the OrbView-2 (images from the SeaWifs sensor), and Meteosat. The data are stored in a *backup* system and distributed to the computers of a local network, which visualize and process them. An exhaustive description of this phase can be found in Cotos (1994) and Trinanes (1998).

The purpose of this work is to use the data that result from phase 1 to elaborate *Phase 2*, which consists of the following steps:

- Apply digital processing techniques to the initial products in order to obtain new data with a biological meaning, for example, the following high-pass filters for the detection of thermic fronts:
  - DoG (see Wells, 1986).
  - Cluster-Shade (see Harlow, 1986).
- Visualize this information independently from the used platform.
- Study the sensitivity and correlation of the initial data, using:
  - the analysis of the main components
  - self-organising Kohonen maps
- Calculate the probability of obtaining fishing catches according to the environmental parameters obtained through teledetection. The applied techniques are:
  - networks trained with the backpropagation algorithm
  - radial basis function networks
  - functional networks
  - ANFIS (neuro-fuzzy inference system)
- Manage all the information efficiently and with centralized control by means of an appropriate database designed after the E-R model.

During *Phase 3*, the system communicates with the users. The information can be sent to the boats in many ways, for instance, by making use of the global network through the Inmarsat satellite (Figure 3). This satellite's network provides a whole range of

Figure 3. Coverage of the geo-stationary satellite network Inmarsat



services (telephone, data, fax, Web, etc.) and is used for maritime emergency communications, which makes it a service that is found on many boats.

Also, the connection with the Internet offers the possibility of communicating information through electronic mail. The embedded software TUNAFIS 4.2, developed by other members of the Systems Laboratory, manages user accounts and extracts the forwarded information for its subsequent use in the graphic interface.

Finally, the information also can be transmitted through the data services that are inherent of the mobile telephone GSM technology. When a boat is near the coast, it frequently disposes of coverage for mobile telephony and can therefore receive information by using a simple mobile GSM terminal connected to a personal computer on board.

Background

Data Sources

The data that are used by our system proceed from remote sensors in the satellites. *Remote sensing* can be defined as the set of techniques that allow us to obtain information on an object, area, or phenomenon through the analysis of data gathered from a device that is not in physical contact with that object, area, or phenomenon (Lillesand, 1987). In our particular case, in which the study object is the marine surface and the device a sensor on board of a specific satellite, we could define oceanographic remote sensing as the study of the oceanic surface through satellite images.

Remote sensing presents certain advantages with respect to the traditional sampling techniques (boats gathering data, for instance):

- global and periodic coverage of the surface
- panoramic and simultaneous vision of a large extension
- information from "invisible" areas (thermic infrared, medium infrared, and microwave)
- digital information

This work uses images from the NOAA, OrbView-2, and Topex-Poseidon series.

Remote sensing is very frequently complemented with other conventional techniques for information treatment such as geographic information systems (GIS) (Aronoff, 1991). The data that proceed from the processing of satellite images are complemented with field data that proceed from a collaborating fishing boat. The information consists of the geographical coordinates at the beginning and the end of the longline, which is the applied fishing technique, and of the catches of each species gathered on a daily basis during the years 1998 and 1999.

Neural Networks

Neural networks have come a long way since the first publications by McCulloch and Pitts: At present, they are established as an interdisciplinary research field that is deeply rooted in neuroscience, psychology, physics, and engineering. Many works before ours have described these networks as an adequate method to obtain reliable results in the analysis of fishing parameters through teledetection data. Famous examples are the following:

- *Komatsu, Aoki, Mitani, and Ishii* (1994) predict Japanese sardine catches with NN. They analyse the synaptic weights in order to determine which is the most important physical or biological factor. Previous intents with regression models did not provide satisfying results due to the correlation between the input variables.
- *Aurelle, Lek, Giraudel, and Berrebi* (1999) predict fishing data with a perceptron with three layers and two neurons in the hidden layer, trained with the error backpropagation algorithm.
- *Dreyfus-Leon* (1999) uses neural networks to predict the behaviour of the fisherman.
- *Aussem and Hill* (2000) predict the presence of a harmful green alga (Caulerpa taxifolia) with a multilayer perceptron and supervised training.
- *Brosse, Guegan, Tourenq, and Lek* (1999) predict the abundance of fish in lakes with neural networks. They compare multiple linear regression, neural networks, and main components analysis, and prove that the NN provides the most precise predictions.
- *Maas, Boulanger, and Thiria* (2000) predicts environmental parameters by using temporal series that correspond to the El Niño phenomenon.

An artificial neural network (ANN) can be defined as a machine that is designed to model the way in which the human brain carries out a task or a function. The neural network is usually implemented with electronic components or simulated by software, and uses massive interconnections between simple processing elements called neurons.

An artificial neural network also can be defined as a massively distributed parallel processer that has the natural capacity to store knowledge and have it available for future use. It resembles the human brain in two aspects:

- Knowledge is acquired by the network through a learning process.
- The connections between neurons, known as synaptic weights, are used to store the knowledge.

The highly connectionist structure of the networks, their parallel processing capacity, and their learning capacity make possible the previously expound analogy (Haykin, 1999).

The learning process is carried out by the learning rule or algorithm, which aims at a specific objective by modifying the synaptic weights in an orderly manner. This process can be supervised or non-supervised. The supervised learning process is controlled by the designer, who decides when the training is finished according to the error that exists between the network output and the training pattern output. Non-supervised learning takes place through the application of data to the inputs, without using the outputs as error indicators.

The computation power of the neural networks resides both in the large amount of parallel interconnections between their processing elements, as in their capacity to learn and to generalize — generalization being the ability of the neural networks to produce a reasonable output for inputs that were not present during the learning process. Thanks to these abilities, the neural networks can solve complex problems that are currently untreatable, although they cannot do it separately and need to be integrated in other systems.

In spite of this restriction, the use of ANN offers the following advantages:

- **Non-linearity:** Since neurons are non-linear elements, the neural networks, which are made of interconnected neurons, are also non-linear. This characteristic is special in that it is distributed over the entire network.
- **Associations between the input and the output:** The networks are able to learn complex associations between input and output by means of a training process that is either supervised or non-supervised.
- **Adaptability:** The ability to adapt the synaptic weights of the network to changes in the environment. A network that is trained to operate in a specific environment can easily be retrained when the conditions of that environment change.
- **Error tolerance:** A neural network is error-tolerant because the knowledge is divided over all the weights.
- **Neurobiological analogy:** The design of a neural network is motivated by the analogy with the human brain, which demonstrates that parallel processing is possible, fast, and powerful.

We have opted for neural networks trained with supervised algorithms because there is no predefined correspondence function between their input and output data. Several authors have successfully applied neural models in the field of remote sensing, as can be seen in Trinanes (1994), Benediktsson (1990), and Torres (2000).

Functional Networks

It was not until the mid 1980s that extensions of the neural network model started to appear. Some examples of extensions are the networks of high order, probabilistic neural networks (Specht, 1990), and neural networks based on "wavelets." These new models, however, still acted as black boxes that do not consider the functional structure and the properties of the object they model. An important characteristic of the functional networks is the possibility to treat with functional restrictions that are determined by the properties that may be known of the model. These restrictions will lead toward a specific network topology and therefore a system of functional equations.

Functional equations are introduced by Castillo and Gutiérrez (1998) as an extension of the neural networks. In a simple but rigorous way, we can define a *functional network* as a neural network in which the weights of the neurons are replaced by a set of functions. Some of its advantages are the following (Castillo, Cobo, Gutiérrez, & Pruneda, 1999):

1. Functional networks can reproduce certain physical properties that in a natural way lead toward the corresponding network, provided that we use an expression with a physical expression in the functions base. In our case, this advantage cannot be exploited because we do not dispose of that information.
2. The estimation of the network parameters can be obtained by solving a linear equations system, which constitutes a fast and unique solution, the global minimum of the error function.

The functional networks have been applied successfully to problems related to medical diagnoses and to the Bayesian distributions. A description of these problems, their solution through functional networks, and the entire formalism that is proper of the equations and the functional networks can be found in the book by Castillo et al. (1999).

A functional network consists of the following elements:

1. **An input layer of storage units.** This layer contains the input data.
2. **An output layer of storage units.** This layer contains the output data.
3. **One or various layers of processing units.** These units evaluate a set of input values that proceed from the previous layer (an intermediate layer or the input layer) and calculate values that will be considered for the next layer. Each neuron has its own neural function that can possess various arguments or inputs: This helps to introduce into each processing unit a part of the mathematical model that helps to explain our problem.
4. **None, one, or various layers of intermediate storage units.** These layers contain units that store intermediate information produced by the neural units. As such they allow us to compel the coincidence of the outputs of the processing units.
5. **A set of directed links.** They connect input units or intermediate layers to neural units, and neural units to intermediate or output units.

ANFIS

The use of neural networks facilitates the learning and minimization of an error function and the use of parallel computation. These techniques, however, provide minor results in representing knowledge and extracting linguistic rules. In a multilayer perceptron, for instance, the knowledge is distributed in a weights set that is difficult to interpret and to convert into rules. Fuzzy logic, on the contrary, translates the semantic terms of the rules into variables that are accepted without problems by the system.

The purpose of the neuro-fuzzy systems is to combine the benefits of both techniques. To this effect, the rules of the system are represented as fuzzy expressions, and the system is trained by using the learning schemes of the neural networks. The neuro-fuzzy systems allow the incorporation of linguistic and numeric data, and the extraction of rules from numeric data.

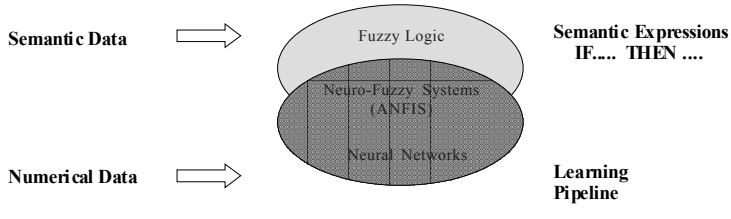
The neuro-fuzzy systems can be divided into two main groups:

1. neuro-fuzzy inference systems
2. fuzzy neural networks

The origin of the neuro-fuzzy inference systems is to incorporate concepts like learning and parallelism into the fuzzy inference systems. The neuro-fuzzy inference systems carry out fuzzy inference (Jang, 1993; Berenji, 1992) by adjusting internal rules parameters with neural networks learning algorithms. Each rule could be executed through a neural network, but it is also possible to apply the ANFIS architecture, as is the case in this work.

The fuzzy neural networks consist of two components in one subsystem, a neural network, and a fuzzy system. A possible combination of both components would consist in replacing the weighted sum of a neuron by a fuzzy operation, or even by an external block that would translate the linguistic information to the corresponding network input.

Figure 4. The neuro-fuzzy systems integrate fuzzy logic and neural networks



Results of the Applied Techniques

Training of Backpropagation Networks and Radial Basis Function

The problem consisted in determining the optimal conditions for a maximal amount of catches of the *Prionace Glauca*. After a study based on main components and Kohonen networks (SOM), we have determined that the set of input data, which consisted of five parameters, could be reduced to four parameters (temperature, heating-cooling of water, chlorophyll, and altimetry). The fifth variable was discarded because it did not provide new information. So, we have not only the data from the fishing logbook, but also a set of patterns that can be used for the training of connectionist systems. Once they are satisfactorily trained, neural networks are very successful prediction tools (Komatsu et al., 1994; Aurelle et al., 1999; Dreyfus-Leon, 1999; Aussem, 2000; Brosse et al., 1999; Maas et al., 2000).

The multilayer perceptron (MLP) trained with error backpropagation and the RBF networks are two well-known types of neural networks. The next section analyses their behaviour in a particular training set.

Backpropagation Networks

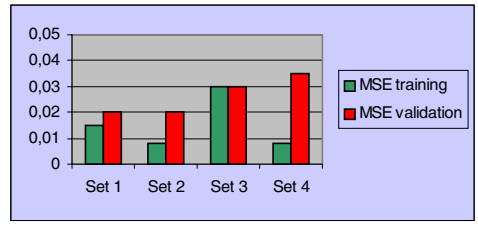
We created a multilayer perceptron with four input variables (SST, cooling-heating, chlorophyll concentration, and altimetry) and one output variable (Quenlla — a kind of shark — catches), and studied its behaviour according to the following criteria:

- the number of units in the hidden layer
- the number of patterns of the training set and the test set
- the variation of the parameters of the learning algorithm

Our purpose was to find the simplest possible network that allows us to determine the conditions under which the catches of the shipped unit can be maximized.

Table 1. Tested training sets and representation of the obtained errors

Set	Nº training patterns	Nº validation patterns
1	57	38
2	72	23
3	48	47
4	80	15



Once the network was trained, we observed that the number of neurons of the hidden layer hardly affects the error presented by the network. There are small initial differences that may be caused by the arbitrary initialization of the network's weights. In all the cases, the MSE of the training set remains close to 0.01, whereas the error of the validation set remains slightly below 0.02.

These results made us opt for the network that has only two neurons in its hidden layer and allows us to interpret more easily the information of the weights set.

Behaviour of the network after varying the number of patterns of the training set and the validation set.

The four different training and validation sets are used to train a network with two neurons in the hidden layer. Each set has the patterns shown in Table 1.

The training results are presented according to the applied training/validation set, and with the indicated parameters.

We can observe that Set 3 provides the worst results: Due to the low number of patterns that compose the training set, the MSE of these patterns reaches the 0.03 level.

Set 4 increases the validation error because the network loses learning capacity. We are trying to design a network that not only has a minimal error during the learning process but is also able to predict the output with the greatest possible precision, and this set is not giving us the desired results.

In Sets 1 and 2, the validation error is almost identical and therefore is not a valid criterium, but there is a small difference in the training error, which is lower for Set 2.

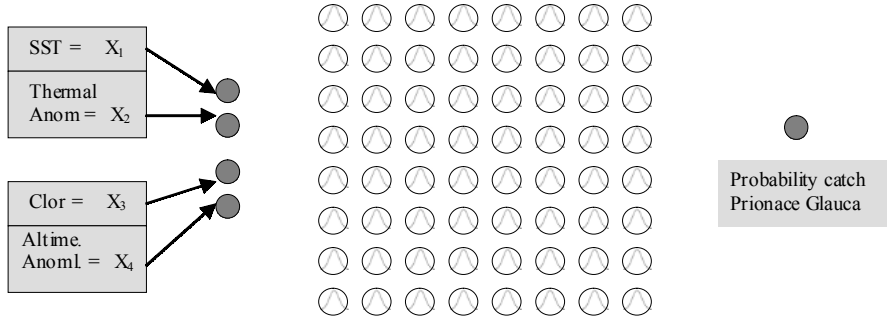
Considering the results in the error table, we can conclude that the speed of the training depends on the value of η. In Set 2, if η=1.0, the error reaches a constant value with 4,000 training cycles, but if η=2.0, the constant error reaches approximately 1,000 cycles. The results, therefore, agree with the theoretical version of the backpropagation method.

Choice of the Network

Finally, we have to select the most convincing result. On the basis of the previous conclusions, we opt for a network with the following characteristics:

1. Two neurons in the hidden layer. Since the number of neurons in the hidden layer does not affect the result, we chose the number according to the subsequent interpretation of the weights set.
2. Training Set 2, which has the most acceptable validation and training error, and has a rather balanced distribution of the different patterns. From now on, we will use this set to train the connectionist systems.
3. The parameters of the backpropagation algorithm will provide a more or less fast training, but will not affect the result.

Figure 5. Designed RBF network



RBF Networks

The obtained results will be compared with the following initialization methods:

1. selection of the centroids based on the training patterns
2. selection of the centroids with Kohonen networks

The network that was applied to carry out these tests used Gaussian functions as activation functions of the neurons of the hidden layer. This layer consisted in a bidimensional structure of 8×8 neurons, which were all connected to the four inputs and the only output (Figure 5).

For reasons of coherence, we chose the same dimensions for the hidden layer as in the previous study based on self-organising maps (SOM). One of the selected initialization methods uses Kohonen networks to choose the centroids values of the Gaussian functions.

These are the results for the two initialization types:

Procedure 1 (Selection of the centroids based on the training patterns)

This procedure selects the t_j centres of the training patterns in course and assigns them to the links between the input layers and the hidden layers (Figure 6).

Procedure 2 (Kohonen Networks)

In the self-organising method of the Kohonen maps, the appropriate centres are generated on the basis of the learning patterns. a is the learning speed: value 0 does not vary the centroid vector, whereas value 1 replaces the selected vector by the current

Figure 6. Representation of the MSE error according to the training cycles for initialization procedure 1. The line indicates the error of the validation set.

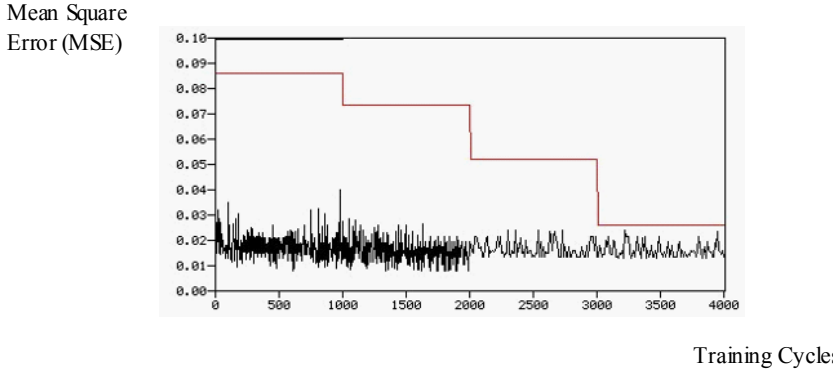
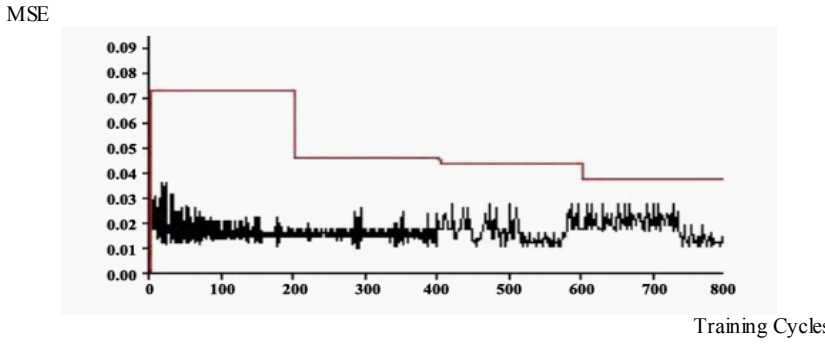


Table 2. Parameters used to initialize with Procedure 2

Learning Cycles	α	*Shuffle*
20,000	0.5	1

Figure 7. Representation of the MSE according to the training cycles for initialization Procedure 2. The line indicates the error of the validation set.



training pattern. The *Shuffle* parameter determines the selection of the initial centres vectors at the start of the procedure. With value 1, the centres vectors are arbitrarily selected among the training patterns set (Table 2).

Figures 6 and 7 show that the results of the training are similar for both initialization procedures, but that the initial error is smaller with Procedure 2. However, the prediction

of the validation data is slightly better for Procedure 1, which is why we will opt for this network.

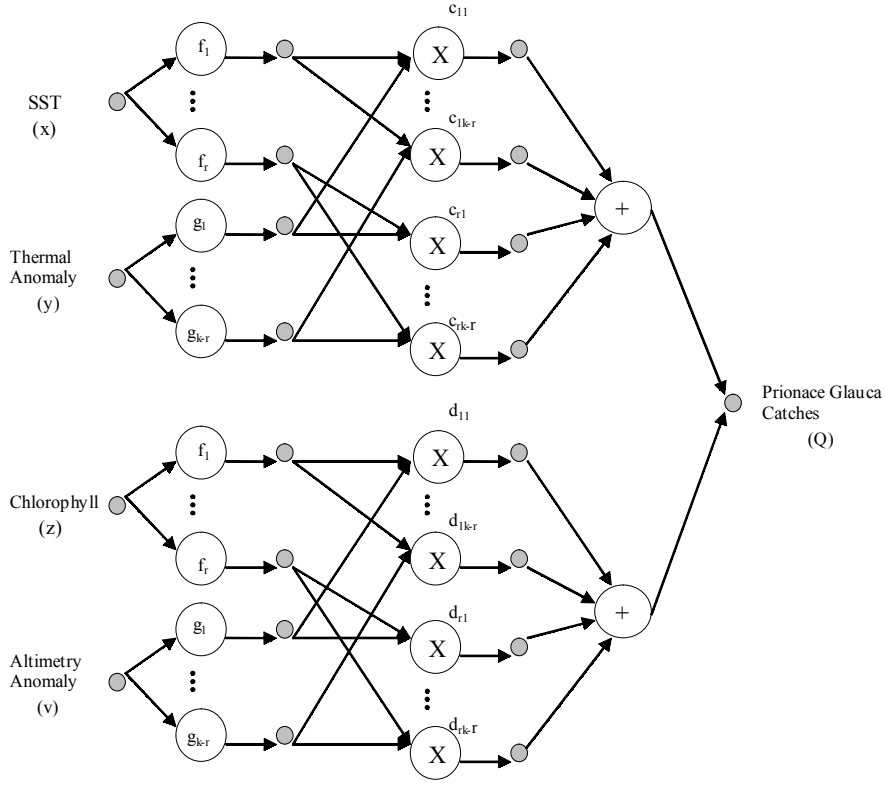
Results of the Functional Networks

We adapted the *separability* model, explained in Castillo and Gutiérrez (1998) and Castillo et al. (1999), to our problem. The topology of the proposed network for the prediction of Prionace Glauca catches is shown in Figure 8.

This model presupposes two functions families, $\{f_i \mid i=1,\ldots,r\}$ and $\{g_j \mid i=1,\ldots,k\text{-}r\}$, from which output Q can be calculated as follows:

$$Q = \sum_{i=1}^{r}\sum_{j=1}^{k-r} c_{ij} f_i(x) g_j(y) + \sum_{i=1}^{r}\sum_{j=1}^{k-r} d_{ij} f_i(z) g_j(v) \qquad (1)$$

Figure 8. Functional network for the prediction of catches, based on the separability model



To calculate the adequate coefficients c_{ij} and d_{ij}, we have to use the training set that was already used for the neural networks. The learning process of our network is described in the following formulas.

The error e_k of each pattern is defined as:

$$e_k = x_{0k} - Q = x_{0k} - \sum_{i=1}^{r}\sum_{j=1}^{k-r} c_{ij} f_i(x_k) g_j(y_k) + \sum_{i=1}^{r}\sum_{j=1}^{k-r} d_{ij} f_i(z_k) g_j(v_k) \quad (2)$$

The purpose of our training is to minimize the sum of the errors of all the patterns, that is, minimize E:

$$E = \sum_{k=1}^{n} e_k^2 \text{, where n is the number of patterns of the training set} \quad (3)$$

Using the method of the square minima, the set of parameters that minimize E must be the solution of the following equations system:

$$\begin{cases} \dfrac{\partial E}{\partial c_{pq}} = 2\sum_{k=1}^{n} e_k f_p(x_k) g_q(y_k) = 0 \\ \dfrac{\partial E}{\partial d_{pq}} = 2\sum_{k=1}^{n} e_k f_p(z_k) g_q(v_k) = 0 \end{cases} \text{ where p=1,…,r; q=1,…,r-s} \quad (4)$$

By replacing e_k by expression (3.2), we obtain the following equations system:

$$\begin{cases} \dfrac{\partial E}{\partial c_{pq}} = 2\sum_{k=1}^{n}\left(\sum_{i=1}^{r}\sum_{j=1}^{k-r} c_{ij} f_i(x_k) g_j(y_k) + \sum_{i=1}^{r}\sum_{j=1}^{k-r} d_{ij} f_i(z_k) g_j(v_k) \right) e_k f_p(x_k) g_q(y_k) = 0 \\ \dfrac{\partial E}{\partial d_{pq}} = 2\sum_{k=1}^{n}\left(\sum_{i=1}^{r}\sum_{j=1}^{k-r} c_{ij} f_i(x_k) g_j(y_k) + \sum_{i=1}^{r}\sum_{j=1}^{k-r} d_{ij} f_i(z_k) g_j(v_k) \right) e_k f_p(z_k) g_q(v_k) = 0 \end{cases}$$

where p=1,…,r; q=1,…,r-s (5)

For reasons of coherence, the training patterns are identical to those that trained the neural networks of the previous section. Functional networks are especially indicated to solve problems with mathematical models, but since our case does not present a model, we opt for the elementary polynomic family as the basis for the solution of the system.

The following functions families were tested:

- **Case 1:** $\{f_i\}=\{1,x,x^2,x^3\}$ and $\{g_i\}=\{x,x^2,x^3,x^4\}$.
- **Case 2:** $\{f_i\}=\{1,x,x^2,x^3,x^4\}$ and $\{g_i\}=\{x,x^2,x^3,x^4,x^5\}$.
- **Case 3:** $\{f_i\}=\{1,x,x^2,x^3,x^4,x^5\}$ and $\{g_i\}=\{x,x^2,x^3,x^4,x^5,x^6\}$.

We must be particularly careful when selecting the functions' families because the determinant of the coefficients matrix could have two identical columns. This is why the family $\{g_i\}$ does not contain the elemental function "1."

The next section reflects the results for each case.

We start by evaluating the success of the training and validation sets. Figure 9 shows that the best learning a priori corresponds to the functions of *Case 2*.

The success percentage is merely orientative: Even with a high success rate, the global error rate can be too high. Therefore, we measure the learning level of the studied systems with the mean square error (MSE).

Figure 10 represents the MSE for the training and validation sets in each of the three cases. We can observe that the training MSE is equal to and even superior to that of the neural networks: logical results, since the more functions are used, the closer the model that results from the functional network to the training data. However, our intention is to have the network extract the necessary knowledge to predict the outputs from the inputs, and control this fact by means of the results that were obtained through the validation set.

Figure 9. Comparison of the results of the training and validation sets

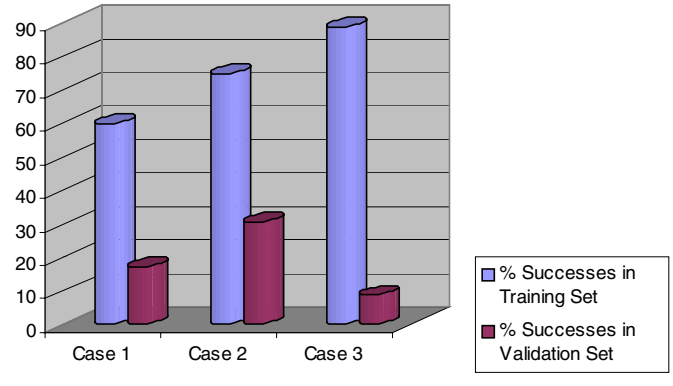
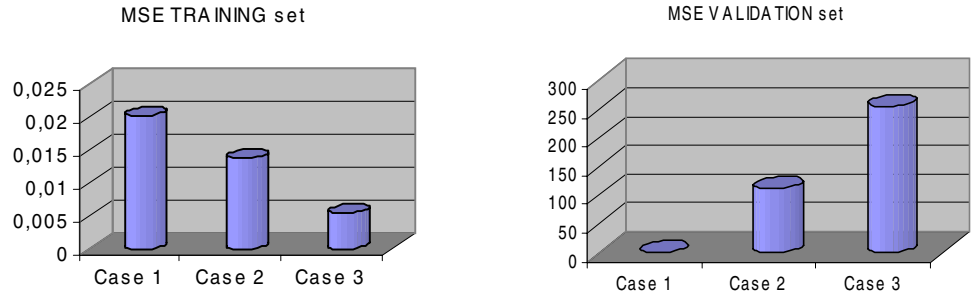


Figure 10. MSE of the training and validation sets



The lowest MSE of the validation sets is found in *Case 1* (0.404), whereas in Cases 2 and 3, the MSE reaches values that are hardly acceptable. This situation is not incoherent with Figure 9, where we observed important numeric differences between the real and the predicted values in the cases that were not considered successful. These differences became less important if we trained the functional network with the functions' base that corresponds to *Case 1*. We now find that the error in the validation set negatively affects the results obtained through the already mentioned neural networks. The fact that there is no explanatory mathematical model for the problem explains this phenomenon: Functional networks are especially effective in cases that do have this subjacent model. Nevertheless, it remains interesting to compare the behaviours of the modern and classical learning architectures.

ANFIS

We implemented ANFIS systems of type 3 (Jang, 1992, 1993; Takagi-Sugeno 1992, 1993) using the already applied patterns for neural and functional networks as training and validation sets. In order to discover the best topology for our system, we designed 10 ANFIS with the following characteristics:

- **Case 1:** two membership functions (or characteristic functions) for each variable. The output is of the order 0 (constant function).
- **Case 2:** four membership functions for each variable. The output is of the order 0 (constant function).
- **Case 3:** three membership functions for the variables SST and altimetric anomaly, and two for chlorophyll and thermic anomaly. The output is of the order 0 (constant function).

- **Case 4:** two membership functions for the variables SST and altimetric anomaly, and three for chlorophyll and thermic anomaly. The output is of the order 0 (constant function).
- **Case 5:** three membership functions for each variable. The output is of the order 0 (constant function).
- **Case 6:** two membership functions for each variable. The output is of the order 1 (linear function).
- **Case 7:** four membership functions for each variable. The output is of the order 1 (linear function).
- **Case 8:** three membership functions for the variables SST and altimetric anomaly, and two for chlorophyll and thermic anomaly. The output is of the order 1 (linear function).
- **Case 9:** two membership functions for the variables SST and altimetric anomaly, and three for chlorophyll and thermic anomaly. The output is of the order 1 (linear function).
- **Case 10:** three membership functions for each variable. The output is of the order 1 (linear function).

In all the cases, the developed systems have the following characteristics in common:

- Used AND rule = product.
- Used OR rule = probabilistic OR = a + b – ab.
- *Defuzzification* method => average weighted sum.
- Hybrid training method explained in next.
- Gaussian membership functions.

The training continued until the error of the training patterns remains constant. In a parallel way, the error in the validation set, and the success and error rate are being controlled. The prediction of our system is considered a *success* if it differs less than 10% of the maximum catches value. Since we have normalized all the catches data, the values are in the [0,1] interval, and our prediction is therefore a success if it differs less than 0.1 with respect to the real value of the boat catches.

According to this criterium, we obtained the following success and error rates in the training set (see Figure 11).

In Figure 11, we can observe that the success rate after the training is promising in several cases (2, 7, 8, 9, and 10). Only in Case 1, the number of errors ranks equally with that of the successes. This is, however, merely an orientative measure: If we want to know the learning capacity of our system, we must study the data of the validation set and in particular the respective MSE.

Figure 11. Comparison of the error and success rate in the training set

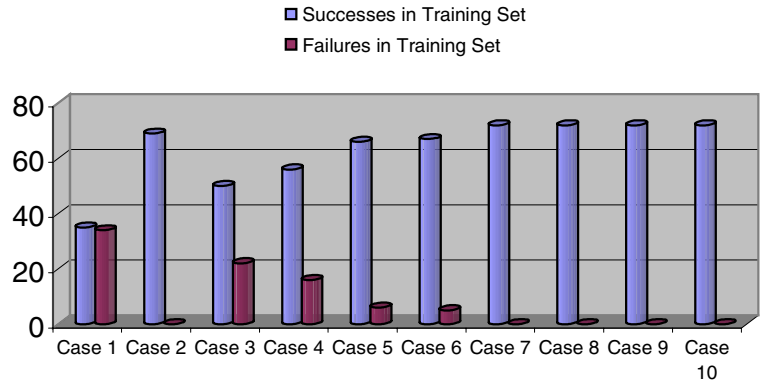
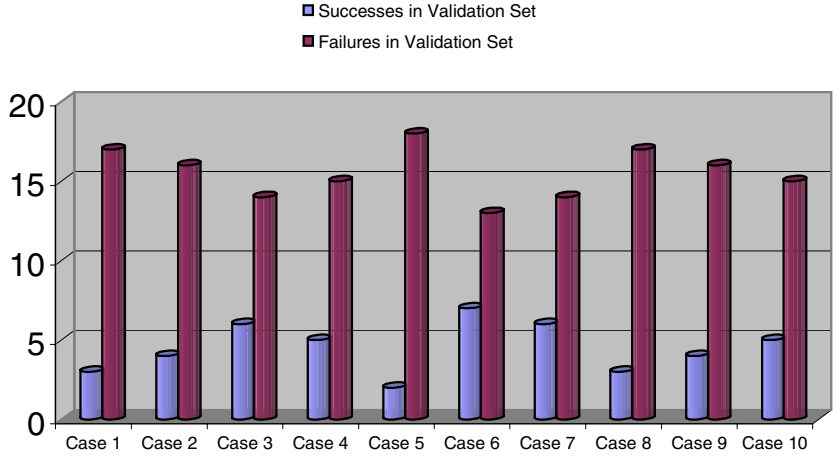


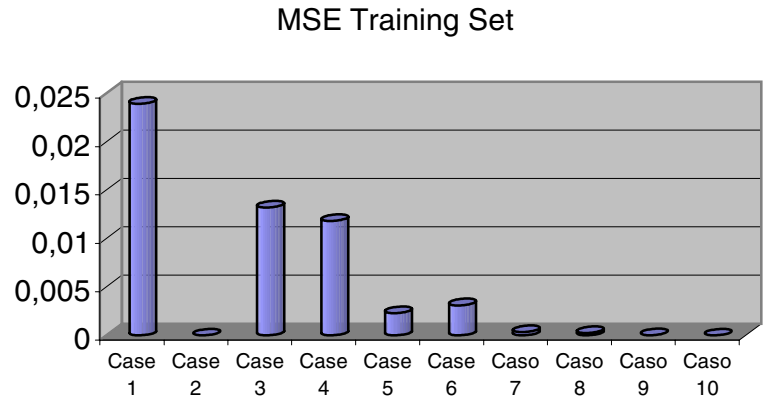
Figure 12. Comparison of the error rate and the success rate in the validation set



The results are less positive for the validation patterns set. Figure 12 shows that the success rate of the prediction is never superior to the error rate.

The MSE indicates the global deviation of the predictions set with respect to its real value. In Figure 13, we can observe that the error is comparable or smaller than that of the previously tested systems. The error of *Case 1* is obviously the highest and is comparable to that of the neural networks. As the system becomes more complex, it

Figure 13. Comparison of the MSE in the training set. In all the cases, the results of the neural networks are equalled or improved



increases its capacity to adjust the training parameters in such a way that the number of errors is minimal.

To test the learning capacity of these systems and compare it with the previously tested networks, we must check the behaviour of the validation set. A system with good generalization capacity will have a good MSE in the validation set and an acceptable MSE for the training patterns.

We may conclude from Figure 14 that *Case 1* is the only ANFIS system with an error comparable to that of the neural networks; in all the other cases, the error increases significantly. This proves the fact that a high success rate or a MSE in the training set does not guarantee good learning, and that this is a clear case of over-training.

We can resume the obtained results by stating that the system with the best generalization capacity is *Case 1*. The next section will give more details on its topology.

Figure 15 represents the characteristics and different layers of our ANFIS system type 3 of the order 0, with two typical functions for each input variable. Compared to the other studied systems, it is simple and offers more learning capacity.

The rules for the calculation of the output data are executed as in Figure 16.

Figure 16 shows how each rule gives a certain output. The proposed ANFIS, which disposes of four input variables with two membership grade functions each, can produce a total of 16 different rules. After *fuzzifying* the inputs and carrying out the corresponding operations, the result *defuzzifies* by means of an average weighted sum.

The learning has modified the parameters of the characteristic Gaussian membership functions, as can be seen in the graphics in Figure 17.

Figure 14. Comparison of the MSE in the validation set. Only in Case 1, the MSE is of the same order as the MSE obtained with neural networks.

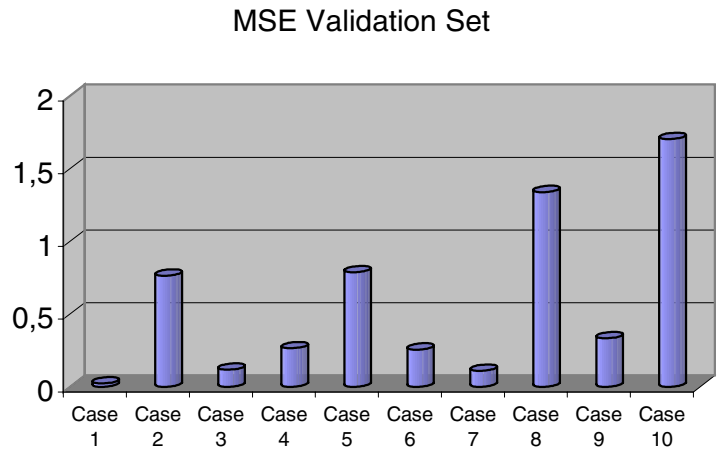


Figure 15. Topology of the ANFIS system of Case 1. For each variable, there are only two characteristic functions (membership functions or MF).

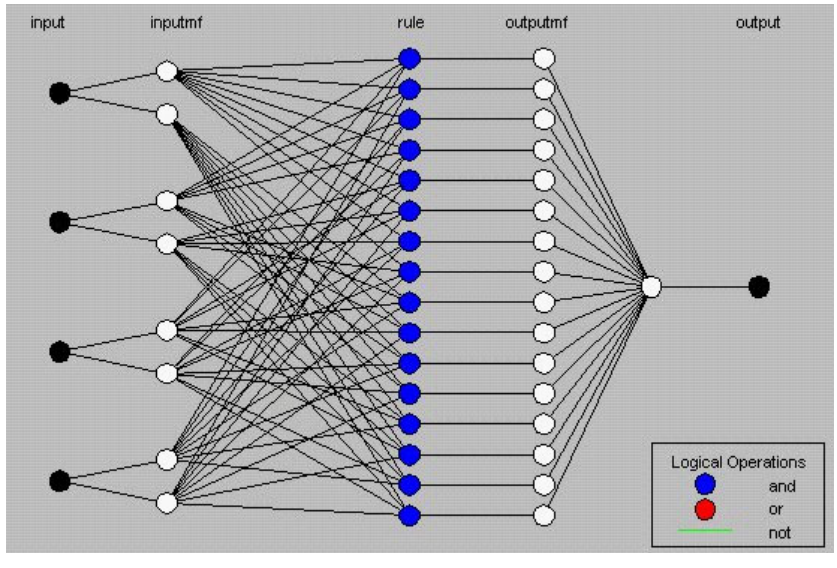


Figure 16. Behaviour of the system when it is presented with the following input pattern: SST=0.9, Thermic Anomaly=0.55, Chlorophyll=0.31, Altimetric Anomaly=0.48. The prediction of the system is 0.232. The real output is 0.22 and can be considered successful.

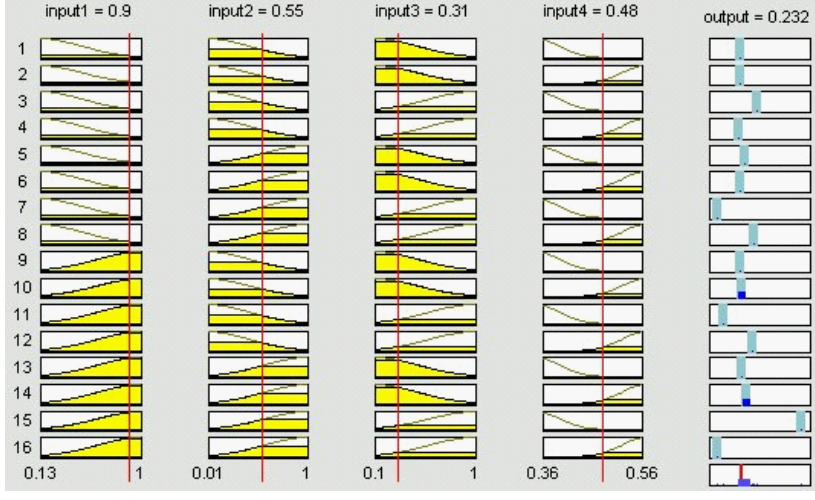


Figure 17. Membership grade functions after the training — (a) SST, (b) thermic anomaly, (c) chlorophyll concentration, (d) altimetric anomaly

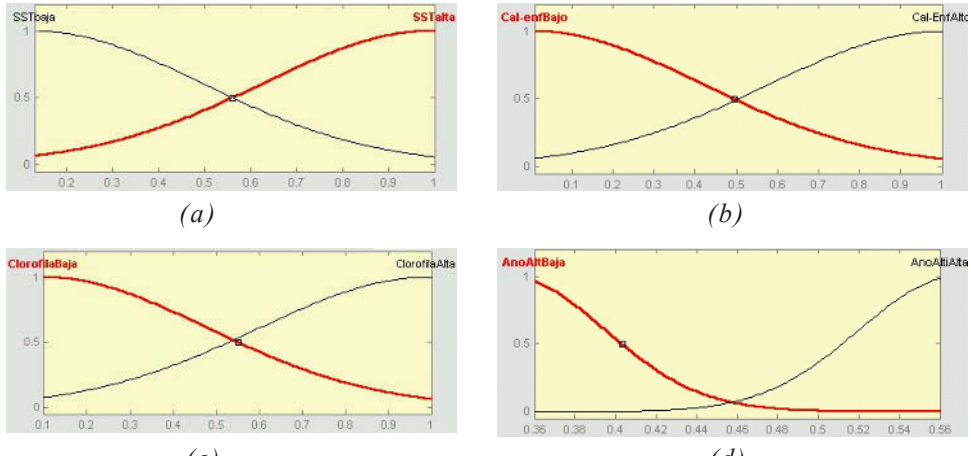


Figure 17d reflects the changes of the two functions according to the modifications of the parameters of the Gaussian functions. The changes of the remaining Gaussian functions are less important.

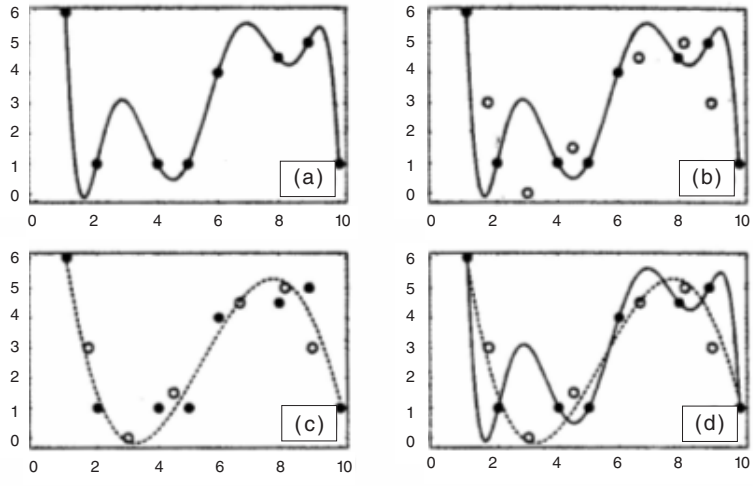
Results of the Algorithm: Discussion

The Problem of Over-Training

The studied algorithm can only be a good predictor if the training and validation set has a small error. If the validation error is much higher than the training error, there is a problem of over-adjustment or over-training. In statistics, it is well known that when we use a model with many parameters to adjust a dataset that proceeds from a process with little freedom, the obtained model may not discover the real tendencies of the original process, although it may present a small data adjustment error. In this case, the training is reduced to the interpolation of the data, including the noise, by means of a complicated sigmoidal function.

We illustrate this problem with the following example. The curve in Figure 18a passes through the training points (represented by black points). It is an interpolating polynomial of grade seven of the eight given points and is therefore a model with zero errors. However, if we consider an alternative set of control data, that same curve cannot predict the new data with precision, as is the case in Figure 18b, because of the excessive amount

Figure 18. An example of over-training — (a) model with too many parameters to adjust the data, (b) a high error in the control data, (c) a model with less parameters approaches in the same way the training data and the validation data, (d) comparison of both models (Adapted from Castillo et al., 1999)



of applied parameters (the eight coefficients of the polynome). A more realistic model can be obtained by considering a small number of parameters (e.g., the polynomial of grade three of Figure 18c). In that case, the approximation error is similar for the training data and the validation datasets and indicates that the real tendencies of the model were captured. Figure 18d shows the difference between both models.

The analysis of Figures 10, 13, and 14 reveals that this is a recurrent problem. Figure 10 represents the MSE for the training and validation sets of the functional network with three different functions bases. When the training MSE decreases, the validation MSE increases too strongly, except for Case 1, where both values simultaneously reach interesting quotas and coincide precisely with the simplest base (i.e., the base that has less functions and less parameters to adjust). Figures 13 and 14 compare several ANFIS structures. The MSE of the validation patterns increases considerably (Figure 14) when the MSE of the training patterns decreases (Figure 13), except for Case 1, where the errors are comparable to those of the neural networks, estimating the patterns of the training and validation set. Case 1 is, once again, the case with the simplest structure and the smallest number of parameters.

Results

We have compared the best results from the trained neural networks, functional networks, and ANFIS, avoiding all the cases that presented over-adjustment.
A first criterion in the choice for one of the four tested algorithm could be the number of successes and failures. Figure 19 shows the level of successes and failures in the training and validation sets.

Figure 19. Comparison of successes and failures in the training and validation sets for the four algorithms

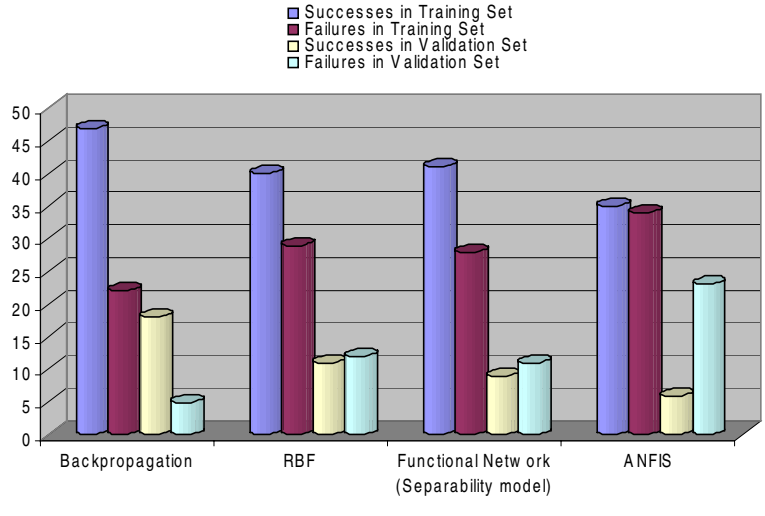
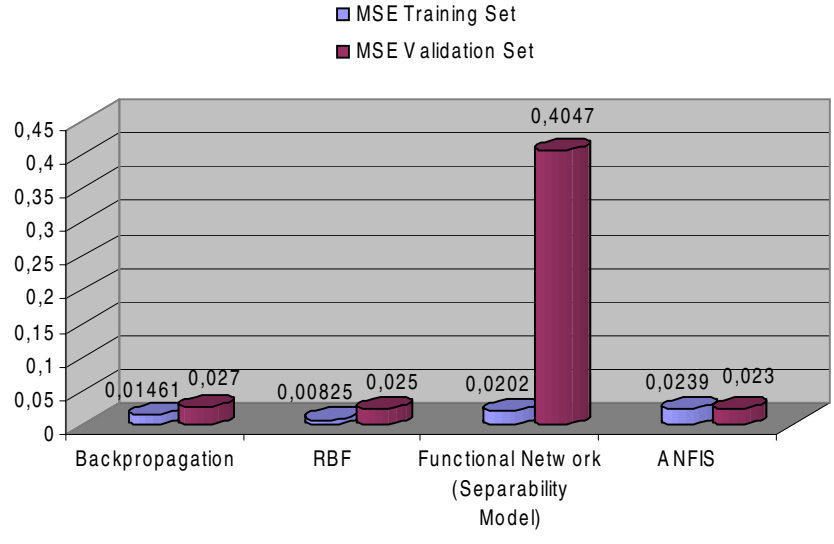


Figure 20. Comparison of the MSE of the different algorithms for the training and validation sets



The best results apparently correspond to the multi-layer perceptron, trained with the error backpropagation algorithm, but we must confirm this by checking the MSE of the training and validation sets (see Figure 20).

The errors in Figure 20 are all of the same order, except for the validation error of the functional network. This means that in the context of our problem, the generalization capacity of this algorithm is smaller, due to the absence of a mathematical model that explains the problem.

Finally, we opted for the implementation of the multi-layer perceptron trained with error backpropagation, which offers more simplicity than the ANFIS and the RBF network.

Generation of Fishing Probability Maps

The proposed system is fed with images from the OrbView-2, NOAA, and Topex-Poseidon satellites. As mentioned in previous sections, the products derived from the processing are maps with the following indicators:

- sea surface temperature (SST)
- weekly thermic anomaly
- thermic fronts

- chlorophyll concentration
- altimetric anomaly

Previous sensitivity studies have shown that the information from the weekly thermic anomaly and the thermic fronts is redundant. Therefore, we suppress the thermic fronts as input for our system, which leaves room for a larger number of training patterns with valid values.

Figure 21 shows four images derived from the processing that correspond to the August 7, 1998. Figures 21a and 21b proceed from the NOAA series, and Figures 21c and 21d from the Orbview-2 and Topex-Poseidon satellites, respectively. As mentioned before, the NOAA series uses data format *tdf*, Orbview-2 uses *hdf*, and the Topex-Poseidon images are in binary format. The programming languages were C for the NOAA images and IDL 5.2 for the Orbview-2 images. Since the images format is best unified before the application of the neural network, we have changed the *hdf* and *binary* images into *tdf* format so that later on they can be treated with codes generated in C. The tdf format has

Figure 21. Images obtained after the initial processing (7-8-1998). (a) sea surface temperature (SST, NOAA series), (b) weekly thermic anomaly (NOAA series), (c) chlorophyll concentration (Orbview-2), (d) altimetric anomaly map (Topex-Poseidon).

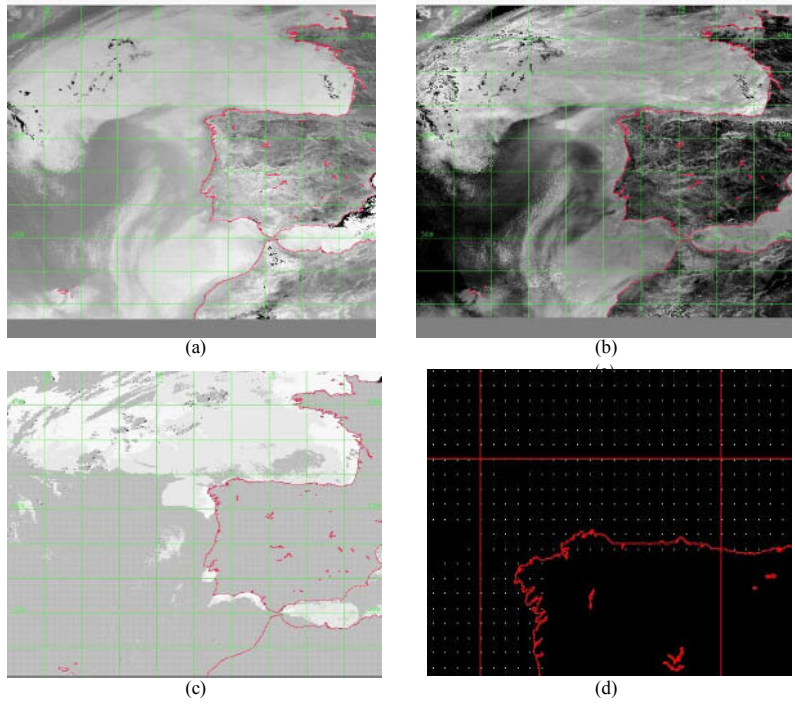
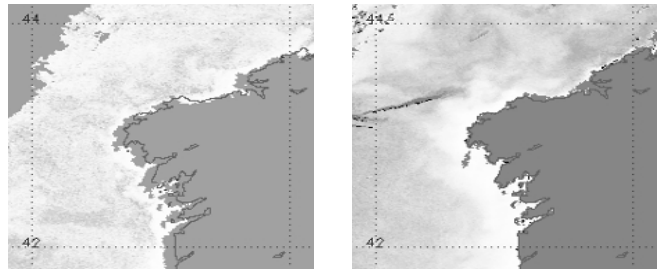


Figure 22. On the left, chlorophyll image from the OrbView-2 with georeferenciation errors due to cylindric projection; on the right, OrBiew-2 image with correct georeferenciation.



a wide range of functions that are especially indicated for the treatment of images in the mentioned format; nevertheless, we must remember to adjust the geographic correction of the images so that there can exist a pixel-to-pixel correspondence between them, and facilitate the application of the corresponding algorithm.

The next figure shows the example of a badly corrected image: The coast line is not adjusted to the real situation.

This problem was avoided by correcting the NOAA and OrbView-2 images with rectangular and cylindric equidistant corrections, respectively. Both geographic corrections are equivalent pixel to pixel if they possess the same spatial resolution. For the Topex-Poseidon image, we calculated the geographic coordinates of each point and afterwards placed them on an image without initial data; Figure 21 shows that the geographic correction of the four images is correct. The spatial resolution of Figure 21d differs from that of the others, which explains the isolated points and the absence of data (value 0 is represented in black): The spatial resolution of this image is 0.25º x 0.25º, that of the other images is 0.025º x 0.025º.

We are now ready to use the backpropagation network to calculate the probability map for the Prionace Glauca catches. The network inputs are the values of the pixels of each image. For the input that corresponds to the altimetric anomaly, we take the value that is not zero and that is closest to the pixel, if it does not already exist. With the output of the network, we finally generate a fishing probability map (see Figure 23), in which the light grey tones indicate the highest probabilities.

Due to the low spatial resolution of the MSLA maps that proceed from T/P, the probabilities grouped in squares of 0.25º x 0.25º.

Figures 23 and 24 are in *tdf* format, with the same geographic correction as the initial images, but if we change that into the well-known JPEG format, we can import it to a GIS. Figure 25 shows the result obtained with ArcView 3.2.

Figure 23. Probability map of Prionace Glauca captures of 7-8-1998, obtained by a neural network trained with the backpropagation algorithm. High grey levels (light tones) indicate high fishing probabilities.

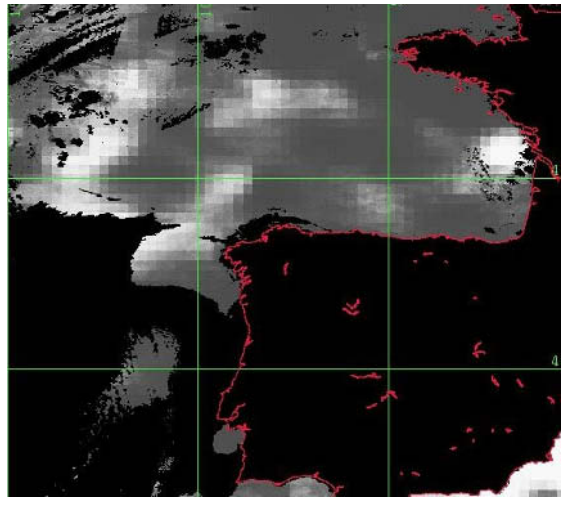


Figure 24. Detail of the probability map of Figure 24

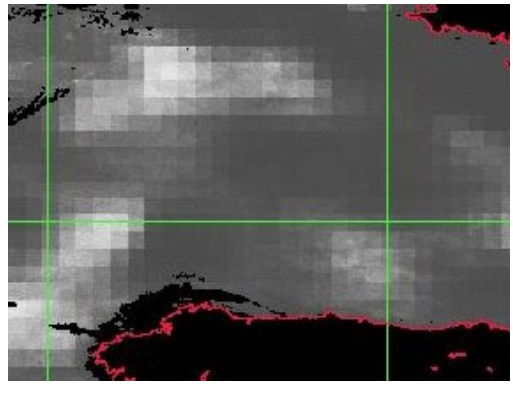
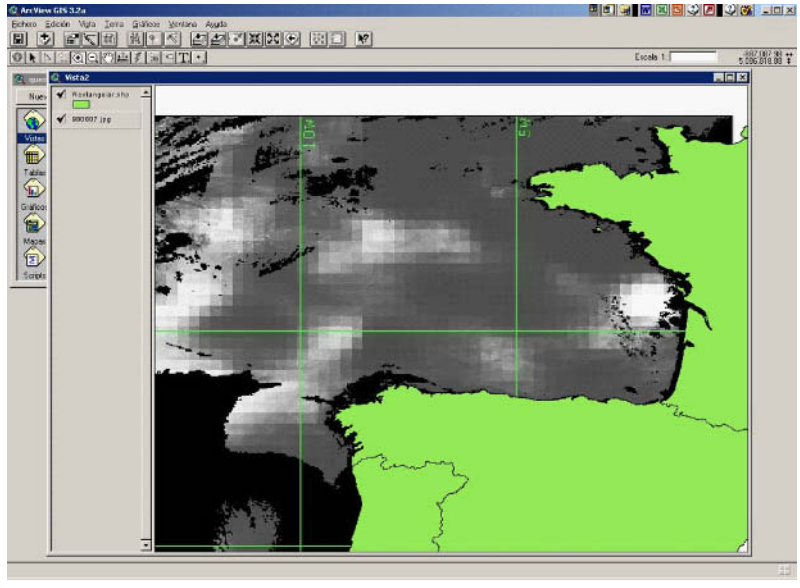


Figure 25. Image 3.19 exported with ArcView 3.2



Conclusions

The main conclusions of this work are the following:

- We have elaborated a support system for decision-making in the operational exploitation of fisheries, by integrating various artificial intelligence techniques into an acquisition system for data that proceed from remote sensors.
- On the basis of the images of two entire years (1998 and 1999) and many field data, we have generated a powerful information system that includes a relational database shaped by environmental parameters, geographic coordinates, and catches.
- A previous study of the available data, elaborated with Kohonen networks and the main components analysis, established that the variables *thermic anomaly* and *thermic fronts* contain redundant information. Therefore, we have opted for the *thermic anomaly* variable because it disposes of more patterns.
- We have designed a new decision support system based on the previously mentioned information system. Our problem is characterized by the absence of global and local models, the inexistence of a knowledge base, and variables that are hardly interrelated (if we exclude the anomalies and the thermic fronts).

Therefore, we have opted for algorithms of the connectionist tendency in the field of artificial intelligence.

- The resulting training set was used in the learning process of the recently created neural paradigm called functional networks (separability model). We have elaborated comparisons with the innovating, hybrid, neuro-fuzzy ANFIS system, as well as with the classical neural networks backpropagation and RBF. The tests with functional networks and ANFIS have revealed several cases of over-training. The best results were obtained with classical neural networks and simple hybrid ANFIS systems.
- We have been able to design several tools that can predict the optimal fishing areas according to information from various satellites. The obtained results were validated by means of patterns that differ from those used during the training.
- We have developed applications that integrate the neural networks and the digital treatment of the images after the previous unification of the distinct image formats of each satellite. The output of the system consists in a fishing probability map generated from the output of the used network. At the same time, we have implemented applications that enable us to access the information system and to calculate punctual predictions. These tools were developed in such a manner that the user can dispose of all the utilities in a comfortable and accessible environment.

References

Aronoff, S. (1991). *Geographic information systems: A management perspective* (2nd ed.). Ottawa, Ontario, Canada: WDL Publications.

Aurelle D., Lek, S., Giraudel, J., & Berrebi, P. (1999) Microsatellites and artificial neural networks: Tools for the discrimination between natural and hatchery brown trout (Salmo trutta, L.) in Atlantic populations. *Ecological Modelling, 120,* 313-324.

Aussem, A., & Hill, D. (2000). Neural-network metamodelling for the prediction of Caulerpa taxifolia development in the Mediterranean sea. *Neurocomputing, 30,* 71-78.

Benediktsson, H., Swain, P.H., & Ersoy, O.K. (1990). Neural network approaches versus statistical methods in classification of multisource remote sensing data. *IEEE Transactions on Geoscience and Remote Sensing, 28,* 540-551.

Berenji, H.R. (1992). Fuzzy logic controllers. In R.R. Yager & L.A. Zadeh (Eds.), *An introduction to fuzzy logic applications in intelligent systems* (pp. 69-96). Kluwer Academic Publishers.

Brosse, S., Guegan, J., Tourenq, J., & Lek, S. (1999). The use of artificial neural network to assess fish abundance and spacial occupancy in the litoral zone of a mesotropic lake. *Ecological Modelling, 120,* 299-311.

Castillo, E., & Gutiérrez, J.M. (1998). Nonlinear time series modeling and prediction using functional networks. Extracting information masked by chaos. *Physics Letters A, 244,* 71-84.

Castillo, E., Cobo, A., Gutiérrez, J.M., & Pruneda, E. (1999). *Introduction to functional networks with applications*. *A neural based paradigm*. Kluwer International Publishers.

Cotos, J.M. (1994, septiembre). *Dinámica y clasificación de estructuras oceánicas para aplicación operacional de pesquerías utilizando teledetección e ingeniería de conocimiento.* Tesis doctoral. Departamento de Física Aplicada, Facultad de Física, Universidad de Santiago de Compostela.

Dreyfus-Leon, M.J. (1999). Individual-based modelling of fishermen search behauviour with neural networks and reinforcement learning. *Ecological Modelling, 120,* 287-297.

Harlow, C.A., Trivedi, M.M., & Conners, R.W. (1986). Use of texture operators in segmentation. *Optical Engineering, 25*(11), 1200-1206.

Haykin, S. (1999). *Neural networks: A comprehensive foundation* (2nd ed.). Prentice Hall.

Jang, J.-S.R. (1992, July). *Neuro-fuzzy modeling: Architecture, analyses and applications.* Doctoral thesis, University of California, Department of Electrical Engineering and Computer Science.

Jang, J.-S.R. (1993). ANFIS: Adaptive-network-based fuzzy inference system. *IEEE Trans. Syst., Man., Cybern, 23*(5), 665-685.

Komatsu, T., Aoki, I., Mitani, I., & Ishii, T. (1994). Prediction o the catch o Japanese sardine larvae in Sagami Bay using a neural network. *Fisheries Science, 60*(4),385-391.

Lillesand, T.M., & Kiefer, R.N. (1987). *Remote sensing and image interpretation.* New York: John Wiley & Sons.

Maas, O., Boulanger, J., & Thiria, S. (2000). Use of neural networks for predictions using time series: Illustration with the El Niño Southern oscillation phenomenon. *Neurocomputing, 30,* 53-58.

Specht, D.F. (1990). Probabilistic neural networks. *Neural Networks, 3,* 109-118.

Takagi, T., & Sugeno, M. (1983, July). Derivation of fuzzy control rules from human operator‘s control actions. *Proceedings of the IFAC Symposium on Fuzzy Information, Knowledge Representation and Decision Analysis* (pp. 55-60).

Takagi, H., Suzuki, N., Koda, T., & Kojima Y. (1992). Neural networks desgned on approximate reasoning architecture and their applications. *IEEE Trans. Neural Networks, 3*(5), 752-760.

Torres, J.A., & Cantón, M. (2000). Detección y reconocimiento de estructuras oceánicas en imágenes AVHRR en el área de las Islas Canarias y NO de África mediante técnicas de computación conexionista. *Revista de la Asociación Española de Teledetección, 13,* 5-12.

Trinanes, J.A. (1998, febrero). *Sistema de información basado en teledetección para ayuda a la explotación operacional de pesquerías de túnidos y otras especies*

pelágicas. Tesis doctoral. Departamento Electrónica e Computación, Universidad de Santiago.

Trinanes, J.A., Torres, J., Tobar, A., & Hernández, C. (1994, noviembre). Clasificación de imágenes multiespectrales mediante redes neuronales. *Revista de la Asociación Española de Teledetección,* 46-54.

Wells, W.M. (March, 2, 1986). Efficient synthesis of Gaussian filters by cascaded uniform filters. *IEEE Trans. Pattern Analysis and Machine Intelligence, 9*(2).

Chapter XIV

A Neural Network Approach to Cost Minimization in a Production Scheduling Setting

Kun-Chang Lee, Sungkyunkwan University, Korea

Tae-Young Paik, Sungkyunkwan University, Korea

Abstract

Cost managers working in manufacturing firms have suffered from the difficulty of determining an optimal cost control strategy. Though the concept of ABC can provide a theoretically nice scheme for cost control, it has been widely known that cost managers have serious trouble comprehending the ABC scheme and applying it to real cost control situations. In this sense, proposing a heuristic method by which cost managers can obtain an approximate cost control strategy comparable to one obtained by ABC would be very meaningful from the view of both theory and practice. To fulfill this need, we suggest using a multi-layered perceptron (MLP) neural network model

with backpropagation learning algorithm, and then approximating the optimal cost control strategy of ABC. The primary concern of this study is to investigate whether such solutions approximated by the MLP would be valid from a statistical perspective. If true, it would mean that cost managers can depend on the neural network method to come up with an optimal cost control strategy comparable to applying ABC. To show the validity of the proposed cost control strategy by using the MLP, this study proposes to solve two problems within the context of a production scheduling situation, using ABC: (1) neural network-based total cost estimation (NNTCE); and (2) neural network-based cycle time estimation (NNCTE). For experimental setup, we assume that two products sharing five types of exogenous variables and three types of endogenous variables are manufactured at the same facility. The MLP neural network approach to NNTCE and NNCTE was generated with a set of 180 training data and 125 test data, all of which were proved to be statistically identical with the ABC results.

Introduction

In the face of increasingly fierce global competition, modern-day manufacturing operations must increase productivity and reduce costs. Because of this, it has become a strategic objective to estimate the various costs of manufacturing more accurately. While traditional cost systems tend to distort cost information by using traditional overhead allocation methods (relying on direct resources such as labor hours), activity-based costing (ABC) has gained a reputation for more accurate cost estimation and calculation methods. ABC traces costs via the activities performed on cost objectives (production or service activities) and results in more accurate and traceable cost information. ABC can help with classifying activities such as value-added and non-value-added, and allows for the elimination of the non-value-added activities (Gunasekaran & Sarhadi, 1998).

ABC was first introduced by Cooper and Kaplan as an alternative to traditional accounting techniques (Cooper & Kaplan, 1988a, 1988b), and has since been used increasingly in multi-level, complex manufacturing organizations (Koltai, Lozano, Guerrero, & Onieva, 2000). ABC models the relationships between products and the resources used in all stages of their production. It is preferable to classical cost calculations because ABC provides a more accurate and consistent way of calculating manufacturing costs (Andrea, Filho, Espozel, Maia, & Quassim, 1999), resulting in more accurate general cost calculations (Kee & Schmidt, 2000). ABC has been applied to various industries (Tsai, 1996), including electronics (Merz & Hardy, 1993), automotive (Miller, 1994), aerospace and defense (Soloway, 1993), airplane manufacturing (Haedicke & Feil, 1991), shipbuilding (Porter & Kehoe, 1994), telecommunications (Hodby, Thomson, & Sharman, 1994), and multi-level, highly automated complex manufacturing systems (Spedding & Sun, 1999; Koltai et al., 2000), among others.

However, one of the most critical problems with ABC is the well-known difficulty of applying it to real-world problems without the need to understand its theoretical complexities. As is often the case, cost managers working in manufacturing firms have

wrestled with the problem of finding the optimal cost control strategy. In the case of managers adopting ABC, it becomes more difficult for them to analyze cost drivers and build a sophisticated framework suitable for the basic philosophy of ABC, part of which often fails eventually, leading to poor results far from the expectations. Therefore, they need to find an alternative way, that is, a heuristic method guaranteeing a pseudo-optimal cost control strategy that would be comparable with one obtained by applying ABC.

In this sense, we suggest using a multi-layered perceptron (MLP) neural network (Rumelhart, Hinton, & Williams, 1986) which has been extensively utilized for a wide variety of application problems, in order to approximate the optimal cost control strategy of ABC. Research motivation here is to provide a practical solution for those cost managers who want to achieve the cost control strategy of ABC. Therefore, if we show how to successfully obtain the approximated cost control strategy of ABC by applying traditional MLP, then it can be surely claimed that the traditional neural network model, such as MLP trained by a backpropagation learning algorithm, has the desirable potential to be easily used by cost managers in the real environment who aim to achieve a cost control strategy having the same quality as an ABC-driven solution.

We assume that a manufacturing organization is to create two products using seven exogenous and four endogenous variables, and the cost manager aims to find out the optimal cost control strategy of ABC by applying the MLP neural network trained with a backpropagation learning algorithm. Based on this assumption, the main objectives of this chapter are twofold. The first objective is to describe manufacturing activities alongside a mathematical model which calculates the inventory carrying costs for each product, the total setup costs, and the total cost per hour for the two products using ABC analysis, under different scenarios combining seven exogenous and four endogenous variables, in order to gain insight into the effects of different planning and control strategies on costs. The second objective is to perform the neural network experiments to approximate the optimal cost control strategy of ABC, which is decomposed into two sub-parts: (1) a neural network-based total cost estimation (NNTCE) using a set of 180 training data and 125 test data generated by combining our exogenous and endogenous variables, and (2) a neural network-based cycle time estimation (NNCTE) using 60 different cases. Rigorous statistical tests are then performed in order to compare the NNTCE and NNCTE outputs with those of theoretical total cost and cycle time, highlighting the advantages of the MLP neural network approach trained with a backpropagation learning algorithm to approximate the optimal cost control strategy of ABC.

Section 2 of this chapter provides a review of background and pertinent literature, and Section 3 describes the methodology we use to produce the appropriate NNTCE and NNCTE outputs. Experiments and corresponding statistical tests are addressed in Section 4. Section 5 summarizes our findings and offers suggestions for future research.

Literature Review and Background

Cost Estimation Methods

Cost Estimation Methods Can Be Classified as Intuitive, Analogical, Parametric, or Analytical

Intuitive methods are based on the past experience of the estimator, and are therefore critically subject to the estimator's biases in judgment and calculation. Since professionally-skilled estimators also are very difficult to find, there are several hindrances to the adoption of intuitive methods as a systematic approach to cost estimation.

Analogical methods estimate the cost of products by comparing them to other, similar products whose costs are already known. This method provides a convenient and powerful approach to cost estimation in the early stages of costing policy.

Parametric methods estimate the costs of a product from the parameters used by its designers, which influence cost in a predictable way that can be represented using a simple equation. Parametric cost estimation parameters tend to characterize a product without describing it completely (Duverlie & Castelain, 1999). For example, Boothroyd and Reynolds (1989) demonstrated the use of the volume or weight of typical turned parts as parameters for approximating cost estimates.

There have been a number of studies focusing on analytical cost estimation methods. Luong and Spedding (1995) developed a generic knowledge-based system for process planning and cost estimation for hole-making, while Takakuwa (1997) utilized simulation to estimate cost for a flexible manufacturing system based on ABC analysis. Yang, Parsaei, Leep, and Wong (1998) integrated information from process planning, scheduling, and cost accounting to estimate cost in more detail. In order to determine overall cost for alternative process plans, Kiritsis, Neuendorf, and Xiruchakis (1999) used Petri net models to calculate the optimum process planning cost. ABC is also an analytical method, which decomposes the required work into elementary tasks, operations or activities with known — or easily calculated — costs (Copper & Kaplan, 1988a, 1988b). A review and comparison of traditional cost accounting and ABC analysis can be found in Park and Kim (1995). More details on ABC will be addressed in the sequel.

ABC

Overview

Activity-based costing (ABC) systems help designers to understand the parameters that affect indirect and support resources. ABC systems differ from traditional systems in that their cost pools are defined as activities rather than production cost centers, and the cost drivers used to assign their activity costs are structurally different from those used in

traditional cost systems (Lewis, 1995). The ABC method identifies items and activities that consume resources and drive costs, such as the number of units produced, labor hours, hours of equipment time, and the number of orders received. ABC assigns activity costs to units of production or to other cost objects such as customers, services, and so forth. Research about ABC systems has focused, accordingly, on two main streams: the selection of optimal cost drivers (Levitan & Gupta, 1996; Schniederjans & Garvin, 1997) and cost estimation (Bode, 1998a, 1998b).

Selection of Optimal Cost Drivers

Since the main objective of this chapter is to use the neural network, a heuristic search technique, we will first summarize the literature that proves the efficiency of heuristic search techniques to select optimal cost drivers (Babad & Balachandran, 1993; Levitan & Gupta, 1996). Babad and Balachandran (1993) attempted to optimize these drivers in ABC by using greedy algorithms, while Levitan and Gupta (1996) tried to optimize the selection of relevant cost drivers using the genetic algorithm in ABC systems. They found that the genetic algorithm not only reduces information gathering costs through fewer drivers, but also produces more accurate costs. Schniederjans and Garvin (1997) proposed the combined analytic hierarchy process (AHP) and zero-one goal programming to select relevant cost drivers in ABC, illustrating how AHP weighting can be combined in the zero-one goal programming model to include resource limitations in the cost driver selection process.

Cost Estimation

ABC generally assumes a linear cost function. Horngren, Foster, and Datar (1997) pointed out that, in practice, cost functions are not always linear, but in fact sometimes exhibit nonlinear behavior. They described a nonlinear function as a cost function in which the graph of total costs versus a single cost driver does not form a straight line within the relevant range. In this way, conventional ABC may distort product costs when a cost activity shows a nonlinear behavior.

The neural network has been used extensively in problem domains that show a high level of nonlinearity, such as bank failure prediction (Tam & Kiang, 1992; Sexton, Sriram, & Etheridge, 2003) and time series identification (Lee & Oh, 1996). Based on the empirical success they have achieved in nonlinearity studies, researchers have attempted to overcome the linearity of cost allocation by using conventional ABC and the neural network (Bode, 1998a, 1998b; Creese & Li, 1995; Garza & Rouhana, 1995; Smith & Mason, 1997).

Significance of the Study

As noted previously, the neural network has been useful in dealing with nonlinearities embedded in target problems. ABC systems that originally assumed linearity in cost behavior with regard to drivers were successfully supported by neural networks, which incorporated nonlinearity into cost estimations (Bode, 1998a, 1998b). The target problems of this chapter, NNTCE and NNCTE, are intended to contribute to the hypothesis that the neural network also can be used to provide support for decision-makers who want to predict total cost and cycle time in a production scheduling situation.

Neural network-based cost estimation in ABC, like NNTCE, has been extensively studied (Bode, 1998a, 1998b; Creese & Li, 1995; Garza & Rouhana, 1995; Smith & Mason, 1997), but no study has shown that neural networks can be used simultaneously for both cost estimation (NNTCE) and cycle time prediction (NNCTE).

Methodology

MLP Neural Network

The neural network model adopted in this study is MLP, and its learning algorithm is backpropagation (Rumelhart et al., 1986). The MLP neural network approach with a backpropagataion learning algorithm has been extensively applied to classification research in business. Examples include accounting inventory method choice (Liang, Chandler, Han, & Roan, 1992), bankruptcy prediction (Tam & Kiang, 1992), and bond rating (Surkan & Singleton, 1990). The MLP neural network model can handle noisy data and adaptively adjust the model parameters when encountering new environments. It is a computer-aided decision support tool that has gained prominence in business decision-making (Bell, Ribar, & Verchio, 1990; Etheridge, Sriram, & Hsu, 2000; Montagno, Sexton, & Smith, 2002), recognizing the nonlinear and linear patterns underlying datasets. Based on these patterns, the MLP neural network classifies data into discrete categories that assist experts in the decision-making process (Liang et al., 1992). Several attributes make the MLP neural network model an attractive decision-support tool. MLP models are classificatory and prediction techniques, but because they employ a backpropagation learning algorithm that approximates practical experience, their output can be used to support decision making. When chosen with care and used with adequate training, they are capable of helping managers recognizing trends, learning from data and making predictions. Qualities such as these increase the MLP neural network's reliability during classification and prediction (Etheridge & Sriram, 1997; Fanning & Cogger, 1994).

Table 1. Exogenous variables

	Product 1	Product 2
Demand per hour	d_1	d_2
Production per hour	p	p
Inventory carrying cost per hour	m_1	m_2
Setup cost per setup	S	S

Production Scheduling Setting

The production scheduling setting of this chapter is designed to include two products and the exogenous variables shown in Table 1, assuming that 1) products are manufactured sharing a common facility, 2) full capacity production is demanded: $d_1 + d_2 = p$, and 3) products cannot be produced at the same time but must take turns in production. Decision variable is cycle time T hours.

Based on those assumptions, we compute the total cost function. The total cost per hour (TC) is the sum of inventory carrying costs for Product 1 (IC_1), inventory carrying costs for Product 2 (IC_2), and setup costs (SC).

IC_i or IC per hour of product i ($i = 1, 2$) is computed as $IC_i = m_i \times AI_i$ where m_i denotes IC_i per hour and AI_i average inventory, *(1/2) × maximum inventory*. Maximum inventory is *(p – d_i) × production run time t_i (= d_i T / p)*. Therefore, IC_i is $m_i \times 1/2\ (p - d_i)\ d_i\ T / p$.

Total SC per hour for the two products is $2\ S/T$, which is a decrease in T. For a given period L (e.g., year), total SC is computed as $2\ S/T \times L = 2\ S \times (L/T) = 2\ S \times N$ where $N = (L/T)$ is the number of setups in the period L.

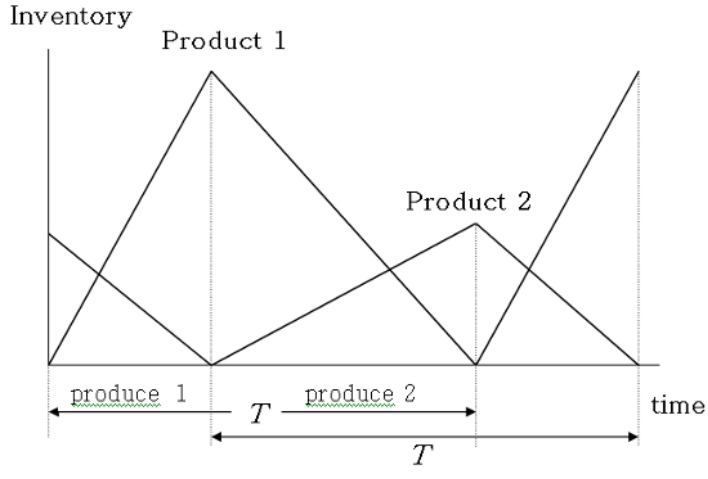
The total cost per hour for the two products (TC) is computed as $TC = IC_1 + IC_2 + SC$. It should be noted that trade-off exists between IC and SC. A graphic display, using two axes for inventory and time, is depicted in Figure 1.

The objective of the production scheduling setting is to minimize TC. To evaluate a neural network-based simulation, we compute the optimal value for cycle time T as a benchmark. Optimal T is computed as follows:

$$T = 2\sqrt{\frac{S}{m_1(1-d_1/p)d_1 + m_2(1-d_2/p)d_2}}$$

$$T = 2\sqrt{\frac{Sp}{m_1(p-d_1)d_1 + m_2(p-d_2)d_2}}$$

Figure 1. Relationship between inventory and time



Optimal AI_i is as follows:

$$AI_i = 1/2\ (p - d_i)\ d_i / p \times 2\ \sqrt{\frac{S}{m_1(1-d_1/p)d_1 + m_2(1-d_2/p)d_2}}$$

$$= (p - d_i)\ d_i \sqrt{\frac{S}{m_1(p-d_1)pd_1 + m_2(p-d_2)pd_2}}$$

Optimal N is as follows:

$$N = L/T = L/[2\sqrt{\frac{S}{m_1(1-d_1/p)d_1 + m_2(1-d_2/p)d_2}}$$

$$= (1/2) \times L \sqrt{\frac{m_1(1-d_1/p)d_1 + m_2(1-d_2/p)d_2}{S}}$$

Proposed Mechanism

Based on the stated advantages of neural networks, we propose using the following steps to solve the NNTCE and NNCTE problems:

Step 1: True Cost Function

The true cost function is denoted as $C = f^T(E,D)$ where E presents an exogenous variable vector (production and cost related parameters), and D a decision variable vector. The best decision $D^T(E)$ is defined as D minimizing $f^T(E,D)$ given E. The problem here is to approximate the best decision $D^T(E)$ by using the neural network.

Step 2: Neural Network Approximation

The neural network is applied to the true cost function, leading to an approximate function $C = f^N(E,D)$. The neural network-based best decision is denoted as $D^N(E)$ which is defined as D minimizing $f^N(E,D)$ given E.

Step 3: Comparison

First, we compare $D^T(E)$ and $D^N(E)$, then we compare the true minimum cost $C = f^T(E,D^T(E))$ and the neural network-based minimum cost $C = f^T(E,D^N(E))$.

Experiments

Neural network experiments are performed using hypothetical values for the seven exogenous and four endogenous variables summarized in Table 2. The MLP neural network used for experiments consists of 7 x 13 x 5 x 4, where 7 means seven neurons in the input layer, 13 means thirteen neurons in the first hidden layer, 5 means five neurons in the second hidden layer, and 4 means four neurons in the output layer, respectively. Input neurons represent seven exogenous variables, while output neurons represent four endogenous variables, as shown in Table 2.

Table 2. Variable list

Variable Type	Name (Parameter)	Value
Exogenous Variables	Production per hour (p)	20
	Demand per hour per Product 1 (d_1)	10
	Demand per hour per Product 2 ($d_2 = p-d_1$)	10
	Inventory carrying cost per hour per Product 1 (m_1)	1
	Inventory carrying cost per hour per Product 1 (m_2)	6
	Set-up cost per setup (S)	80
	Cycle time (T)	$2*SQRT(S/((m_1*(1-d_1/p)*d_1) + m_2*(1-d_2/p)*d_2))$
Endogenous Variables	Inventory carrying cost per hour for Product 1 (IC_1)	$m_1*(1-d_1/p)*d_1/2*T$
	Inventory carrying cost per hour for Product 2 (IC_2)	$m_2*(1-d_2/p)*d_2/2*T$
	Total Setup cost (SC)	$2*S/T$
	Total Cost (TC)	IC_1+IC_2+SC

NNTCE and NNCTE

All of the following experiments were conducted using the information in Table 2. Let us first focus on the NNTCE problem. Since TC is the sum of IC_1, IC_2, and SC, we need to check the performance of the MLP neural networks for each component of TC. *NeuroShell 2* software (www.wardsystems.com) was used to perform the MLP neural network experiments.

We calculated TC in accordance with changes in cycle time T, assuming the same exogenous variables. A set of 180 training data was randomly generated, changing the fixed level of production and demand per hour parameters of IC and SC. The MLP neural network model was developed using the 180 data set, identifying the appropriate prediction function against the unknown test data set.

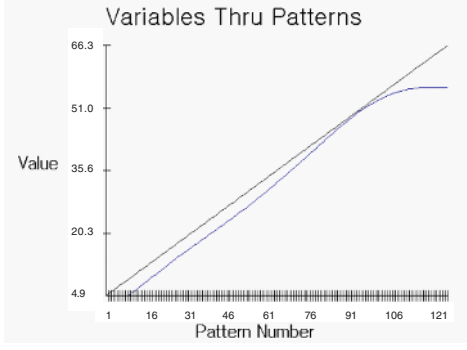
To prove the validity of the MLP neural network approach, a set of 125 test data was randomly generated using the same exogenous variables (except cycle time T) that were used to prepare the training dataset. Figure 2 depicts the MLP neural network-based cost estimation for IC_1, IC_2, and the total SC reveals that the MLP neural network simulation provides sufficient cost estimation power in the rather complicated production scheduling setting designed for this chapter.

Based on the estimated costs shown in Figure 2, NNTCE can be found by summing them. Figure 3 depicts the results of the real TC versus NNTCE.

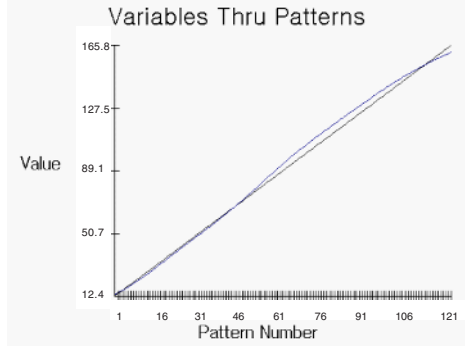
Let us discuss the cycle time T estimated in Figure 3. The NNTCE output is 77.106, at which the estimated cycle time is 14. In other words, the NNCTE output is 14. In contrast, the

Figure 2. MLP neural network-based cost estimation results for IC_1, IC_2, and SC

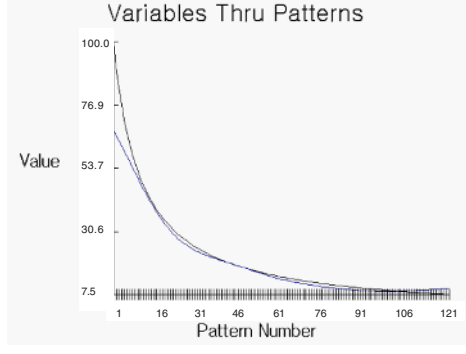
(a) Cost estimation for IC_1



(b) Cost estimation for IC_2



(c) Cost estimation for SC



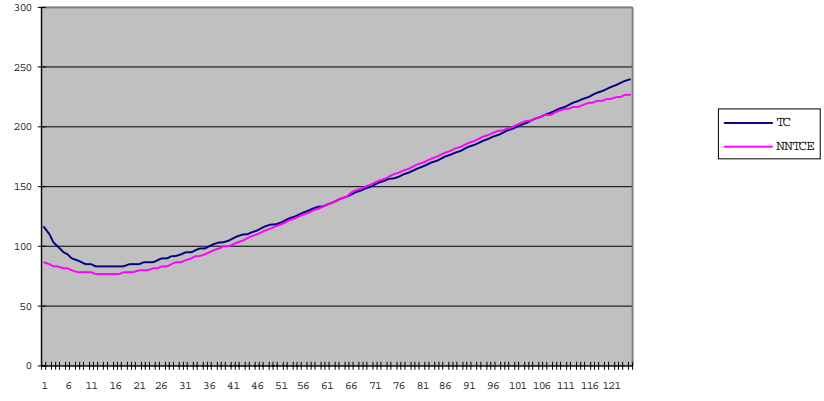
real *TC* value is 83.247 and the cycle time *T* is 15, results which are very close to the NNTCE and NNCTE outputs.

In order to ascertain that the MLP neural network can be used reliably as a tool for producing proxy values for real *TC* and *T*, we must conduct a statistical test.

Statistical Tests

By using the MLP neural network approach, we want to solve the NNTCE and NNCTE problems successfully. If this goal is accomplished, it will mean that the MLP neural network approach is a practical way of implementing ABC in a real production scheduling

Figure 3. Comparison of real TC and NNTCE results



situation. Since the MLP neural network approach requires only input-output variables to be specified, once the neural network model is properly trained, it is very easy for practitioners to get reasonable NNTCE and NNCTE results.

Before concluding that the MLP neural network approach yields statistically valid NNTCE and NNCTE results in complicated production scheduling situations, however, we must confirm the statistical validity of the MLP neural network approach to cost estimation and cycle time estimation. The following test cases were prepared for this purpose:

$p = 20$
$d_1 = 9, 10, 11, 12$
$d_2 = 11, 10, 9, 8$
$m_1 = 1, 2, 3, 4, 5$
$m_2 = 6, 5, 4, 3, 2$
$SC = 40, 50, 60$

With fixed p, we have 60 different cases, considering four cases for demand per hour (d), five cases for inventory carrying costs (m), and three cases for setup costs (SC). Therefore, the total cost TC and cycle time T can be estimated for each of 60 different cases, using the 180 training dataset and the 125 test dataset. Using these 60 cases, descriptive statistics comparing the NNTCE and NNCTE results with real values have been summarized in Table 3.

The NNTCE results are slightly lower than the real TC values. By contrast, the NNCTE results are slightly greater than the real T values. If the NNTCE and NNCTE results are

Table 3. Descriptive statistics for NNTCE and NNCTE results

		N	Mean	Standard Deviation	Standard Error	95% Confidence Interval for Mean		Minimum	Maximum
						Lower Limit	Upper Limit		
Total Cost (*TC*)	Estimated	60	80.5377	8.0838	1.0436	78.4494	82.6259	65.78	96.20
	Actual	60	82.0307	7.2872	.9408	80.1483	83.9132	73.32	91.65
	Total	120	81.2842	7.7000	.7029	79.8924	82.6760	65.78	96.20
Cycle (*T*)	Estimated	60	15.20	4.36	.56	14.07	16.33	3	24
	Actual	60	14.60	2.22	.29	14.03	15.17	12	18
	Total	120	14.90	3.46	.32	14.28	15.52	3	14

Table 4. t-test results for NNCTE and NNCTE

		N	Mean	Standard Deviation	Standard Error	95% Confidence Interval for Mean		Minimum	Maximum
						Lower Limit	Upper Limit		
Total Cost (*TC*)	Estimated	60	80.5377	8.0838	1.0436	78.4494	82.6259	65.78	96.20
	Actual	60	82.0307	7.2872	.9408	80.1483	83.9132	73.32	91.65
	Total	120	81.2842	7.7000	.7029	79.8924	82.6760	65.78	96.20
Cycle (*T*)	Estimated	60	15.20	4.36	.56	14.07	16.33	3	24
	Actual	60	14.60	2.22	.29	14.03	15.17	12	18
	Total	120	14.90	3.46	.32	14.28	15.52	3	14

virtually the same as the real *TC* and *T* values, respectively, we can suggest using the MLP neural network approach in production scheduling situations.

A t-test was applied for this purpose, and the results are shown in Table 4: the hypothesis NNTCE and NNCTE results were the same as real *TC* and *T* values were not rejected, with under a 95% level of confidence. Therefore, it can therefore be concluded safely and with statistical validity that the NNTCE and NNCTE results are virtually the same as the theoretical *TC* and *T* values.

Concluding Remarks

It is well known that cost managers who want to control total cost and cycle time at minimum levels in a specific production schedule are tempted to apply ABC in real settings. However, due to the technological difficulties as well as theoretical complexities of ABC, many cost managers have been frustrated. This study proposes a simple but robust approach to resolving this issue by applying the MLP neural network model.

Applying complicated theoretical frameworks in order to control total cost in a specific production schedule might seem like too much of a challenge for decision-makers who must monitor manufacturing operations on a real-time basis. But they have no choice but to rely on such theories if they want to operate the manufacturing facilities properly. This study organized two sub-problems of NNCTE and NNTCE to determine whether the MLP neural network approach to predicting total cost and cycle time in a production scheduling situation gives statistically identical results to real total cost and cycle time. If the MLP neural network does produce the estimated total cost and cycle time that are virtually the same as the ideal values obtainable by applying complicated models, cost managers in manufacturing organizations could be freed from struggling with sophisticated techniques to keep total cost at reasonable minimum levels. Statistical results show that the MLP neural network approach is capable of providing a safe and robust decision support platform for decision makers who want to minimize costs in a real manufacturing setting.

References

Andrea, M.C., Filho, R.C.P., Espozel, A.M., Maia, L.O.A., & Quassim, R.Y. (1999). Activity-based costing for production learning. *International Journal of Production Economics, 62,* 175-180.

Babad, Y.M., & Balachandran, B.V. (1993). Cost driver optimization in activity-based costing. *The Accounting Review, 68*(3), 563-575.

Bell, T. B., Ribar, G.S., & Verchio, J. (1990). Neural nets versus logistic regression: A comparison of each model's ability to predict commercial bank failures. In R.P. Srivastava (Ed.), *Proceedings of the 1990 Deloitte and Touche/University of Kansas Symposium of Auditing Problems,* Lawrence, KS (pp. 29-58).

Bode, J. (1998a). Neural networks for cost estimation. *Cost Engineering, 40*(1), 25-30.

Bode, J. (1998b). Decision support with neural networks in management of research and development: Concepts and application to cost estimation. *Information and Management, 34*(1), 33-40.

Boothroyd, G., & Reynolds, C. (1989). Approximate cost estimates for typical turned products. *Journal of Manufacturing Systems, 8*(3), 185-193.

Cooper, R., & Kaplan, R.S. (1988a). Measure costs right: Make the right decisions. *Harvard Business Review, 65*(5), 96-103.

Cooper, R., & Kaplan, R.S. (1988b). How cost accounting distorts product cost. *Management Accounting,* (April), 2002.

Creese, R.C., & Li, L. (1995). Cost estimation of timber bridges using neural networks. *Cost Engineering*, *37*(5), 17-22.

Duverlie, P., & Castelain, J.M. (1999). Cost estimation during design step: Parametric method versus case based reasoning method. *International Journal of Manufacturing Technology, 15,* 895-906.

Etheridge, H.L., & Sriram, R.S. (1997). A comparison of the relative costs of financial distress models: Artificial neural networks, logit and multivariate discriminant analysis. *Intelligent Systems in Accounting, Finance, and Management, 6,* 235-248.

Etheridge, H.L., Sriram, R.S., & Hsu, K. (2000). Artificial neural networks help auditors evaluate client financial viability. *Decision Sciences, 31*(2), 531-549.

Fanning, K., & Cogger, L.O. (1994). A comparative analysis of artificial neural networks using financial distress prediction. *Intelligent Systems in Accounting. Finance, and Management, 3*(4), 241-252.

Garza, J.M., & Rouhana, K.G. (1995). Neural networks versus parameter based applications in cost estimating. *Cost Engineering, 37*(2), 14-18.

Gunasekaran, A., & Sarhadi, M. (1998). Implementation of activity-based costing in manufacturing. *International Journal of Production Economics, 56-57,* 231-242.

Haedicke, J., & Feil, D. (1991). In a DOD environment: Hughes aircraft sets the standard for ABC. *Management Accounting, 72*(8), 29-33.

Hodby, T., Thomson, J., & Sharman, P. (1994). Activity-based management at AT&T. *Management Accounting, 75*(10), 35-39.

Horngren, C.T., Foster, G., & Datar, S. (1997). *Cost accounting: A managerial emphasis*. NJ: Prentice-Hall.

Kee, R., & Schmidt, C. (2000). A comparative analysis of utilizing activity-based costing and theory of constraints for making product-mix decision. *International Journal of Production Economics, 63,* 1-17.

Kiritsis, D., Neuendorf, K.P., & Xiruchakis, P. (1999). Petri net techniques for process planning cost estimation. *Advances in Engineering Software, 30,* 375-387.

Koltai, T., Lozano, S., Guerrero, F., & Onieva, L. (2000). A flexible costing system for flexible manufacturing systems using activity based costing. *International Journal of Production Research, 38*(7), 1615-1630.

Lee, K.C., & Oh, S.B. (1996). An intelligent decision support approach to time series identification by a neural network-driven decision tree classifier. *Decision Support Systems, 17,* 183-197.

Levitan, A., & Gupta, M. (1996). Using genetic algorithms to optimize the selection of cost drivers in activity-based costing. *International Journal of Intelligent Systems in Accounting, Finance and Management, 5*(3), 129-145.

Lewis, R.J., (1995). *Activity-based models for cost management systems*. Westport, CT: Quorum Books.

Liang, T. P., Chandler, J.S., Han, L., & Roan, J. (1992). An empirical investigation of some data effects on the classification accuracy of probit, ID3, and neural networks. *Contemporary Accounting Research, 9*(1), 306-328.

Luong, L.H.S., & Spedding, T. (1995). An integrated system for process planning and cost estimation in hole making. *International Journal of Advanced Manufacturing Technology, 10,* 411-415.

Merz, C.M., & Hardy, A. (1993). ABC puts accountants on design team at HP. *Management Accounting, 75*(3), 22-27.

Miller, S.H. (1994). The view from inside: GM's general auditor looks back. *Journal of Accountancy 177*(3), 44-46.

Montagno, R., Sexton, R.S., & Smith, B.N. (2002). Using neural networks for identifying organizational improvement strategies. *European Journal of Operational Research, 142*(2), 382-395.

Park, C.C., & Kim, G.T. (1995). An economic evaluation model for advanced manufacturing system using activity-based costing. *Journal of Manufacturing Systems, 14*(6), 439-451.

Porter, T.J., & Kehoe, J.G. (1994). Using activity-based costing and value analysis to take the pain out of downsizing at a naval shipyard. *National Productivity Review, 13*(1), 115-125.

Rumelhart, D.E., Hinton, G.E., & Williams, R.J. (1986). Learning internal representations by error propagation. In D.E. Rumelhartand & J.L. McClelland (Eds.), *Parallel distributed processing: Exploration in the microstructure of cognition*. Cambridge, MA: MIT Press.

Schniederjans, M.J., & Garvin, T. (1997). Using the analytic hierarchy process and multi-objective programming for the selection of cost drivers in activity-based costing. *European Journal of Operational Research, 100*(1), 72-80.

Sexton, R.S., Sriram, R.S., & Etheridge, H. (2003). Improving decision effectiveness of artificial neural networks: A modified genetic algorithm approach. *Decision Sciences, 34*(3), 421-442.

Smith, A.E., & Mason, A.K. (1997). Cost estimation predictive modeling: Regression versus neural network. *The Engineering Economist, 42*(2), 137-160.

Soloway, L.J. (1993). Using activity-based management systems in aerospace and defense companies. *Journal of Cost Management, 6*(4), 56-66.

Spedding, T.A., & Sun, G.Q. (1999). Application of discrete event simulation to the activity based costing of manufacturing systems. *International Journal of Production Economics, 58,* 289-301.

Surkan, A., & Singleton, J. (1990). Neural networks for bond rating improved by multiple hidden layers. *International Joint Conference on Neural Networks,* San Diego, CA (vol. 2, pp. 157-162).

Takakuwa, S. (1997, December 7-10). The use of simulation in activity-based costing for flexible manufacturing systems. *Proceedings of the 1997 Winter Simulation Conference,* Atlanta, GA (pp.793-800).

Tam, K.Y., & Kiang, M.Y. (1992). Managerial applications of neural networks: The case of bank failure predictions. *Management Science, 38*(7), 926-947.

Tsai, W.-H. (1996). Activity-based costing model for joint products. *Computer and Industrial Engineering, 31*(3/4), 725-729.

Yang, Y.N., Parsaei, H.R., Leep, H.R., & Wong, J.P. (1998). A manufacturing cost-estimation system using activity-based costing. *International Journal of Flexible Automation and Integrated Manufacturing, 6*(3), 223-243.

Chapter XV

Intrusion Detection Using Modern Techniques: Integration of Genetic Algorithms and Rough Sets with Neural Nets

Tarun Bhaskar, Indian Institute of Management, Calcutta, India

Narasimha Kamath B., Indian Institute of Management, Calcutta, India

Abstract

Intrusion detection system (IDS) is now becoming an integral part of the network security infrastructure. Data mining tools are widely used for developing an IDS. However, this requires an ability to find the mapping from the input space to the output space with the help of available data. Rough sets and neural networks are the best known data mining tools to analyze data and help solve this problem. This chapter proposes a novel hybrid method to integrate rough set theory, genetic algorithm (GA), and artificial neural network. Our method consists of two stages: First, rough set theory is applied to find the reduced dataset. Second, the results are used as inputs for the neural network, where a GA-based learning approach is used to train the intrusion

detection system. The method is characterized not only by using attribute reduction as a pre-processing technique of an artificial neural network but also by an improved learning algorithm. The effectiveness of the proposed method is demonstrated on the KDD cup data.

Introduction

The need for secured networked systems is now well established. With the widespread use of the Internet and other computer networks, both e-commerce and data communications depend on secure networks. Organizations have embraced information technology to share information and streamline their business operations. This makes it critical to have networks that function efficiently and reliably. E-business mode of operation is a necessity for today's global organizations. This also has brought with it the unwanted effect of security breaches.

A good intrusion detection system (IDS) is important to ensure the survivability of network systems. Intrusion detection is based on the fact that an intruder's behavior will be significantly different from that of a legitimate user. The number of intrusions is dramatically increasing and so are the costs associated with them. The number of incidents reported to Carnegie Melon's Computer Emergency Response Team/Coordination Center (CERT/CC) has increased from the range of 2,000 to 3,000 in the early and mid 1990s, to 52,658 in 2001, 82,094 in 2002, and 137,529 in 2003. It also reports that e-crime has cost the organizations approximately $666 million in 2003. The data published by the U.S. General Accounting Office shows about 250,000 attempts to attack the system and only 1 to 4% of those are detected. Rule mining can be used on large databases to generate learning algorithm to detect the attack on the site. This approach has tremendous thrust in numerous business applications such as e-commerce, recommender systems, supply chain management, group decision support systems, and so forth.

IDS helps network administrators prepare for and deal with network security attacks. It collects information from a variety of systems and network sources, which is then analyzed for signs of intrusion and misuse. Identifying the appropriate method is important in network intrusion since performance in terms of detection accuracy, false alarm rate, and detection time become critical for near real-time monitoring. Data mining is a very useful technique to extract meaningful information and improve the decision-making process. The extracted information is refined to gain useful knowledge, which is then used to predict, classify, model, and summarize the data being mined. Rule induction, neural networks, genetic algorithms, fuzzy logic, and rough sets are the widely used data mining techniques for industry classification and pattern recognition. The output of the IDS can be represented as shown in Table 1.

The traffic at a site can be broadly classified into normal and abnormal. We refer to abnormal traffic here as *attack*. A deployed IDS evaluates all the traffic that passes through it and classifies it as normal or attack. Proper detection of the attack situation is more important than a normal situation being classified as an attack. So, the emphasis

Table 1. Output of IDS

		Actuality	
		Attack	No Attack
Prediction by the IDS	Attack	Detected	False Alarm
	No Attack	Not Detected	Correct Prediction

here is on proper detection of the attacks, while keeping false alarms within acceptable level.

In order to develop better security and defense mechanisms against network attacks, it is important to investigate the patterns of attacks on network systems and sites. Data mining techniques have shown promising results when applied to such problems. In this chapter, we build on existing methods of IDS and evaluate the applicability of artificial neural networks and rough sets for the purposes of intrusion detection. A hybrid model is constructed with the integration of rough sets with neural networks by utilizing their complementary nature. Rough set is used as a preprocessing tool to eliminate the redundant data from the huge database. This reduces the learning time of the neural network. The changes of overcoming a local minimum is enhanced by going in for genetic algorithm- (GA) based learning which increases its prediction accuracy.

The purpose of this chapter is to develop neural network and rough sets to mine rules from large databases. The insight gained is utilized by developing a hybrid system based on the rough-neuro approach and the effectiveness is demonstrated on KDD cup data. The contributions of this study are as follows:

(i) Reduction of attributes from decision table by using rough sets

(ii) A rough-neuro method for intrusion classification

(iii) Statistical comparison of the means and standard deviations for the mentioned data mining techniques

This chapter is organized as follows. We first provide the literature survey followed by a hybrid model for IDS which uses GA for learning. Then, the basics of rough sets as a data mining tool are presented. This is further demonstrated by applying it to IDS. To improve the effectiveness of IDS, the synergies between neural network and rough sets are explored and the same is presented in rough-neuro method. We then present the effectiveness of the proposed approach in the results and discussion section, and finally conclude by giving directions for future research.

Literature Survey

Contemporary, large-scale, networked systems that are highly distributed improve the efficiency and effectiveness of organizations by permitting different levels of organiza-

tional integration. However, such integration is accompanied by elevated risks of intrusion and compromise. Incorporating IDS into an organization's systems can mitigate these risks. The literature in the field of IDS is very rich. In this section, we concentrate on the use of data mining techniques for IDS. In particular, our focus will be on neural networks, genetic algorithms, and rough sets.

One of the effective ways to enhance network security is to use an efficient intrusion detection system (Garfinkel & Spafford, 1991; Russel & Gangeni, 1991). IDS collects information from a variety of systems and network sources, and then analyzes the system for signs of intrusion and misuse. The major functions performed by an IDS are:

1. Monitoring and analyzing the server
2. Assessing the integrity of critical systems
3. Recognizing known attacks by activity patterns
4. Responding automatically to detected activity
5. Reporting the outcome of the detection process

An intuitive definition for a secured computer system is: "A computer system is secured if it can be depended upon to behave as it is expected to" (Garfinkel & Spafford, 1991). More formally, security is often defined in terms of confidentiality, integrity, and availability (Russel & Gangeni, 1991). Intrusion in computer networks is the inability of the network to work at its desired efficiency. An intrusion can be defined as "any set of actions that attempt to compromise the integrity, confidentiality, or availability of a resource" (Heady, Luger, Maccabe, & Servilla, 1990).

There are various types of intrusions on networks. A user running a program which executes actions in excess of the user's authorization is one type of intrusion. The user overloading the system and preventing other users from utilizing the system is another type of intrusion and is termed as denial-of-service attack. The detection and reporting of intrusion has become a very important task. The intrusion detection methods can be broadly classified into two categories:

- **Anomaly Detection:** Anomaly detection algorithm models normal behavior. Anomaly detection models compare sensor data from network traffic to normal patterns learned from the training data. If the sensor data deviates from normal behavior, the anomaly detection model classifies the data as having originated from an attack. Most of these algorithms require that the data used for training is purely normal and does not contain any attacks.
- **Misuse Detection:** Misuse detection algorithms model known attack behavior. They compare sensor data from network traffic to attack patterns learned from the training data. If the sensor data matches the pattern of some known attack data, the observed traffic is considered intrusive. Misuse models are typically obtained by training on a large set of data in which the attacks have been labeled.

So, to detect and report intrusions, a system must have some form of IDS installed. "An IDS must identify, preferably in real time, unauthorized use, misuse, and abuse of computer systems" (Mukherjee, Heberline, & Levitt, 1994).

There are two broad techniques for network security: protection and detection (Lunt, 1993). The protection technique tries to protect the system from attack. The most commonly used protection device is the firewall which allows only valid data to pass through it. Another approach is using an IDS, which collects information from a variety of systems and network sources and analyzes the data stream for signs of intrusion or misuse. The modeling of an IDS has always been an important problem for researchers in this area. Denning (1987) proposes an IDS model based on historical data, and Lunt (1993) provides a detailed survey of the work done on this topic. The success of an IDS is measured by the false positives and the true positives.

Researchers have shown that the efficiency of an IDS can be improved by using data mining techniques. Lee, Stolfo, and Mok (1999) present a data mining-based model. Chebrolu, Abraham, and Thomas (2004) investigate the performance of data mining techniques and propose a Bayesian network based technique. Zhu, Premkumar, Zhang, and Chu (2001) compare different data mining techniques and have found that neural networks are better at identifying malicious connections. Bonifacio, Cansian, and Carvalho (1997) apply neural networks for modeling an IDS. Lippmann and Cunningham (2000) state that the efficiency of an IDS improves by using neural networks. Most intrusion detection systems with a single-level structure can only detect either misuse or anomaly attacks. Some IDSs with multi-level structure or multi-classifier are proposed to detect both attacks. A multi-level hierarchical neural network is developed by Zhang, Jiang and Mohamed (2004) which is useful for adaptive learning. Joo, Hong, and Han (2003) insist that neural networks suit this domain. But the major problem with neural networks is that the training consumes much time and processing power because of the gradient-based learning algorithm (Hassoun, 1998).

Among the numerous neural networks present in literature, a feedforward neural network with backpropagation learning algorithm is probably the most popular and widely used. It uses supervised learning in which the network is trained using data for which inputs as well as desired outputs are known. The backpropagation approach uses gradient-descent algorithm to minimize the error on the training data by propagating errors backwards through the network starting at the output units and working backwards toward the input units. When applied to intrusion data there are high chances of it converging to a local minimum, not global minimum. Also, backprop requires computing derivatives for gradient direction which should be continuous, differentiable, non-decreasing, and easily computable. The backprop network is highly dependent on the choice of learning rate and proportionality constant of momentum. Zhu et al. (2001) did not obtain encouraging results applying the standard backprop neural network for intrusion detection. On account of the difficulties encountered, it is better to go for a non-gradient-based learning algorithm. The proposed method of GA-based learning for neural networks has the added advantage of self-adapting nature and also increased convergence speed.

GA also has been used for modeling an IDS. The concept of GA is given by Holland (1975) and is successfully used as an optimization method by Goldberg (1989). GAs have now

been established as an efficient optimization and data mining tool. Balajinath and Raghavan (2001) use genetic algorithms for learning the behavior of the computer user. Crosbie and Spafford (1995) use genetic programming based on suggestions from Koza (1992) and achieve good results.

Rule deduction has always been a problem with neural network-based data mining techniques. Rough set theory introduced by Pawlak (1982) is a data mining technique which provides a good method for rule deduction. This is a mathematical tool to deal with vagueness and uncertainty of the system. But the method became popular only after he published the book on rough sets (Pawlak, 1991). This tool is pretty useful in data analysis and rule generation. Since then, there has been a rapid progress in this field. The application of rough set method for knowledge discovery is also shown by Pawlak (1997). To date, there are around 1,500 research papers on the theory and applications of rough sets. For a detailed survey in this field, readers are referred to Pal and Skworon (1999).

Researchers also have started using rough sets in developing the intrusion detection systems. Cai, Guan, Shao, Peng, and Sun (2003) use this method to develop an IDS for anomaly detection. Zhu et al. (2001) have done a comparative study of different data mining tools for IDS and found that rough sets perform better than the other methods.

The complementary properties and similar usage of the two methods encouraged researchers to go for a hybrid of these two methods. The research work in this domain is still in its infancy. Yahia, Mahmod, Sulaiman, and Ahmad (2000) have developed a hybrid method for an expert system. To the best of our knowledge, only Li and Wang (2004) have used a rough-neuro hybrid method for data mining.

GA-Based Learning

Now, we explain and discuss the method proposed in Bhaskar, Kamath, and Moitra (2004) and develop a reliable and robust technique for IDS. The algorithm, the problem associated with it, and the possible solutions are the focus of discussion. The paper proposes a GA-based learning mechanism for neural network. The basic framework is borrowed from the field of cognitive science. The aim is to develop an IDS which adapts to the changes in the environment.

Artificial neural network is a method commonly used for learning. Since learning using neural network approach is a complex, time-consuming process, the connection between learning and evolution can be used to decrease the complexity of the problem and hence speed up the adaptation. Chalmers (1990) argues that evolution and learning are the two most fundamental processes of adaptation. The emergence of an adaptive system can be done using a connectionist as well as a genetic approach. According to Charles the usual learning process in ANN and is a connectionist approach in which the nodes in a layer are connected to those in a different layer. The kind of emergence found in genetically-based systems differs from that found in connectionist systems. Unlike connectionist systems that support synchronic emergence-emergence over levels, genetic-based systems support diachronic emergence-emergence over time. So, a method to achieve synchronic emergence through evolutionary methods which involves making an indirect connection between a genotype and a phenotype is used. The

genotype is the collection of genetic information passed on between generations (in GA it is a string of bits). The phenotype is the behavioral expression of the genotype, an entity that interacts with the environment, and is subject to selection by differential fitness. The benefit of using indirect mappings from genotype to phenotype allows for an open-ended space search. A feature of the existing genetic search is that a genotypic space is precisely specified in advance, and the search cannot go outside this space. Say, for example, we specify a genotype of 5 bits. By doing this we restrict the search space to vary from 0 (00000) to 32 (11111). Since it is difficult to know in advance precisely the low-level computations that would be appropriate for a specific high-level behavior, it makes sense to use genetic methods to search for an appropriate low-level computational form.

IDS Based on the Hybrid Model

We now discuss the details of the experiment performed to find the appropriate IDS model using the framework given by Chalmers. The data source is knowledge discovery in database (KDD) cup (http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html). This dataset is generated via a simulated U.S Air Force LAN at Lincoln Labs of Massachusetts Institute of Technology. A connection consists of 41 attributes and a final output (indicating whether the connection is a normal or a malicious connection). The original data classifies a particular connection as normal or malicious and gives the type of attack corresponding to the malicious connection. The output is binary, which indicates the connection as normal or malicious. As suggested by Williams and Sharda (1994) and ratified by Zhu et al. (2001), a balanced dataset (i.e., equal proportion of normal and abnormal data) is used. The simplest non-trivial topology of single-layer feedforward network with 41 input and 1 output nodes with supervised learning is used for the experiment.

The change in weights in the neural network is done using GA-based learning algorithm. For a given connection, from input unit i to output unit j, local information include four items:

- a_i the activation of the input unit i.
- o_j the activation of the output unit j.
- t_j the training signal on output unit j.
- w_{ij} the current value of the connection strength from input i to output j.

The genome must encode a function F, where

$$\Delta w_{ij} = F(a_i, o_j, t_i, w_{ij})$$

F is taken as a linear function of the four dependent variables and their six pair-wise products and is given by:

$$\Delta w_{ij} = k_0(k_1.w_{ij} + k_2.a_i + k_3.o_j + k_4.t_i + k_5.w_{ij} + k_6.w_{ij}.o_j + k_7.w_{ij}.t_i + k_8.a_i.o_j + k_9.a_i.t_i + k_{10}.o_j.t_i)$$

The genome consists of 35 bits in all. The first five bits code for the scale parameter k_0, which can take the values $0, \pm 1/256, \pm 1/128 \ldots \pm 32, \pm 64$, via exponential encoding. The first bit encodes the sign of k_0 (0 = negative, 1 = positive), and the next four bits encode the magnitude. If these four bits are interpreted as an integer *j* between 0 and 15, we have

$$|k_0| = \begin{cases} 0 & \text{if } j = 0 \\ 2^{j-9} & \text{if } j = 1,2,\ldots,15. \end{cases}$$

The other 30 bits encode the other 10 coefficients in groups of three. The first bit of each group expresses the sign, and the other two bits express a magnitude of 0, 1, 2, or 4 via a similar exponential encoding. If we interpret these two bits as an integer *j* between 0 and 3, then:

$$|k_i| = \begin{cases} 0 & \text{if } j = 0 \\ 2^{j-1} & \text{if } j = 1,2,3. \end{cases}$$

The selected data are divided into 30 datasets. Each dataset is called a task. Out of the 30 datasets, 20 are used for training and 10 are held back for testing. The chromosomes representing *k*'s values are evaluated. To evaluate a chromosome, appropriately sized networks are configured for each of the 20 tasks and the procedure is conducted for each task.

- For each epoch, the network is presented with all the training patterns, and the weights are updated according to the encoded learning rule. The absolute values of connection strengths are capped at 20 to prevent runaway learning rules.
- The network is presented with each pattern once more, and its outputs are recorded. If the desired and actual outputs are on opposite sides of 0.5, the response is counted as an error.
- Fitness is calculated as 100 * (1 – {number of errors}/{number of patterns}), yielding a percentage value between 0 and 100. This function is used for its simplicity.

The fitness of a chromosome is taken as the average fitness over all 20 tasks, and chromosomes are probabilistically selected for inclusion in the next generation based on

their cumulative fitness over generations. The selection mechanism is roulette selection with elitism (that is, the most-fit chromosome is always included in the next generation). The fitness of the generated rule is tested using the test set.

Results

A brief analysis of the results with illustration is provided here. A two-point cross-over and elitist selection is used. The best fitness of 92.4% is achieved with 55% cross-over rate and 1% mutation rate.

For illustration, we provide the two best results obtained during the experiment. The best fitness value is 92.4% and the respective values of *k's* are 2, 0, 0, 1, -1, 0, 0, 0, -2, 2, 0. The corresponding learning functions derived from this set of values are:

$$\begin{aligned}\Delta w_{ij} &= 2(o_i - t_i - 2.a_j.a_i + 2.a_j.t_i) \\ &= 2(o_i - t_i)(1 - 2.a_j) \\ &= 4(a_j - 0.5)(t_i - o_i)\end{aligned}$$

The second best fitness is 90.2% with *k* values: -1, 0, 0, 1, 1, 0, 0, 0, -2, 4, 0, respectively. So the corresponding learning rule is:

$$\begin{aligned}\Delta w_{ij} &= -1(o_i + t_i - 2.a_j.o_i + 4.a_j.t_i) \\ &= [2.a_j(o_i - t_i) - (o_i + t_i)]\end{aligned}$$

The performance of the IDS is evaluated based on the average efficiency. The average fitness over 10 epochs is 80.4%. This means that on an average the IDS can detect a malicious connection in about 80% of the cases.

Is this technique efficient? Rough sets are known to be an efficient data mining tool for problems of this type (Zhu et al., 2001). Efficiency and variance of classification are the two performance parameters of interest. We first discuss rough sets in general and move on to applying it to IDS. Then, the comparison of the two methods is presented.

Rough Sets for Data Mining

The rough set theory is based on the assumption that with every object of the universe of discourse we associate some information. If a student studying in a programme is an object, then the courses taken and the grades in the courses are the information related

Table 2. Dataset for rough classification

	Age	Fitness	Sportsman
x1	16-30	50	Yes
x2	16-30	0	No
x3	31-45	25	No
x4	31-45	25	Yes
x5	46-60	49	No
x6	16-30	49	Yes
x7	46-60	49	No

to the object. Objects characterized by the same information are indiscernible (similar) with respect to the information available about those objects. The indiscernibility relation generated in this way is the mathematical basis of rough set theory.

Any set of all indiscernible objects are called an elementary set and form a basic unit of knowledge called atom. Any union of some elementary set is called a crisp set, otherwise it is a rough set. Thus, we can say that every rough set has a hazy boundary. Unlike crisp sets, there are objects in the universe which cannot be classified with certainty to be an element of the rough set or its complement. So there are elements which may or may not be the element of the set. We consider an example to understand the concept. Suppose we want to find out whether a person can be a good sportsman or not based on age and fitness from Table 2.

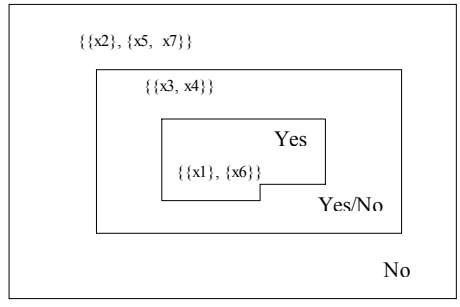
It can be easily noticed that the pairs x3, x4 and x5, x7 have the same set of input values, but the first pair has different outputs whereas the second pair has the same output. So, it is difficult to find a rule for the output based on the inputs. We try to find the indiscernibility relations for a particular output based on the inputs.

$$IND(\{\text{Age}, \text{Fitness}\}) = (\{x1\}, \{x2\}, \{x3, x4\}, \{x5, x7\}, \{x6\})$$

So, from the table we try to classify these subsets into some specific region (Yes/No). All the subsets, except {x3, x4} can be easily classified into one of the regions. But {x3, x4} is difficult to be classified. So if the age is in the 31 to 45 range and the fitness level is 25, the person may or may not be a good sportsman. This element comes in the rough area of the set. The regions and the elements belonging to that region have been shown in the Figure 1. The presence of the Yes/No region with non-zero cardinality shows that this set is a rough set and the elements in the rough area is {x3, x4}. The rough set theory provides us with methods to handle the vagueness available in the set.

The use of rough sets for data mining has been discussed in the literature survey section. Here, we explain the basic steps and terminologies used in rough set data analysis.

Figure 1. Diagrammatic representation of rough sets



Data Table

The data is generally given in the form of a table out of which we are expected to extract some rules. Formally, a data table is a 4-tuple $S = < U, Q, V, f >$, where U is a finite set of objects (universe), $Q = \{q_1, q_2, ..., q_m\}$ is a finite set of attributes, V_q is the domain of the attribute q and $f: U \times Q \rightarrow V$ is called information function such that $f(x,q) \in V_q$. To every (non-empty) subset of attributes P is associated an indiscernibility relation on U denoted by I_P.

$$I_P = \{(x, y) \in U \times U : f(y,q) \ \ \forall q \in P$$

If $(x,y) \in I_P$, it is said that the objects x and y are P-indiscernible.

Approximation

Let S be a data table, X a non-empty subset of U and $\phi \neq P \subseteq Q$. The P-lower approximation and the P-upper approximation of X in S are defined respectively as:

$$\underline{P(X)} = \{x \in U : I_P(x) \subseteq X\}$$

$$\overline{P(X)} = \cup_{x \in X} I_P(X)$$

So, the elements of $\underline{P(X)}$ are all and only those objects which definitely belong to the set and $\overline{P(X)}$ are all those elements which may belong to the set. In the set shown in Figure 1, {x1, x6} are the elements of $\underline{P(x)}$ and {x1, x3, x4, x6} are the elements of $\overline{P(X)}$.

Dependence and Reducts

A set of attributes $T \subset Q$ totally depends on a set of attributes $P \subset Q$ if all the values of the attributes from T are uniquely determined by the values of the attributes from P, that is, if a functional dependence exists between evaluation of attributes from P and by the attributes from T.

Superfluous data is a very important issue of concern in rule extraction. Let $P \subset Q$ and $p \in P$. It is said that attribute p is superfluous in P if $I_P = I_{P-\{p\}}$; otherwise, p is indispensable in P.

The set P is independent if all its attributes are indispensable. The subset P' of P is a reduct of P (denoted by *Red(P)*) if P' is independent and $I_{P'} = I_P$. If we have more than one reducts, then the intersection of the sets of all reducts is called the core of the dataset.

Decision Rules

The attributes of the decision table can be divided into condition attributes and the decision attributes. The relationship between the condition attributes and the decision attributes is called rule extraction. These rules are logical statements (implications) of the type "*if....., then.....,"* where the antecedent specifies values assumed by one or more condition attributes and the consequence implies assignment to one or more decision class. So, the syntax of the rule is the following:

$$if\ f(x,q_1) \text{ is equal to } r_{q_1}\ and\ f(x,q_2) \text{ is equal to } r_{q_2}\ and\ .....\ f(x,q_n) \text{ is equal to } r_{q_n},$$

$$then\ x \text{ belongs to } Y_{j_1}\ or\ Y_{j_2}\ or\ ....Y_{j_k}$$

An object $x \in U$ supports decision rule r if its description is matching both the condition part and the decision part of the rule. We also say that decision rule r covers object x if it matches at least the condition part of the rule. Each decision is characterized by its strength, defined as the number of objects supporting the rule.

Rough Set Method for IDS

Our interest lied in coming up with a good neural network for intrusion detection. The GA-based neural network showed promising results. Rough set is a well-established method for data mining, and hence we have compared these two methods. First, we evaluated the dataset with rough sets and a brief analysis of this method is presented in this section.

Data and Preprocessing

The dataset consists of 41 input attributes and one binary output. The pre-processing is conducted in a similar manner as explained earlier under hybrid model for IDS. So the dataset used in this experiment is same as that used in the ANN model.

Finding the Reducts

The reduct is the set of most useful and meaningful attributes in the dataset. We have used the *Indiscernibility Matrix* concept for finding the reduct. Its implementation is based on a structure called indiscernibility matrix. It takes the dissimilarity in the dataset for a particular output to find the reduct. By using the method on our dataset, we got 12 reducts with three attributes each. So, we need to concentrate on these reducts only for the rough set experiment.

Rule Reduction

After finding the reduct, we need to find the rules according to which the classification would be made. For rule reduction, we have used basic minimal covering. This uses the minimal covering algorithm (minimal number of possibly shortest rules covering all the examples). Nine rules are formulated. The best rule has strength of 79 and support of 79 out of 150 data points.

Validation

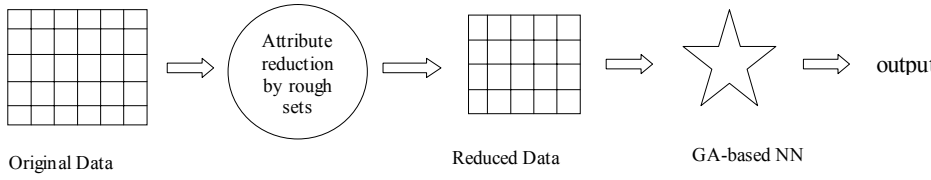
Any rule formulated by a particular approach needs to be validated. We also use the basic minimal covering for validation. The results are tabulated in Table 3. The binary output 0 represents a normal connection, whereas the output 1 represents a malicious connection.

Table 3 shows that the overall efficiency of the model is 80% and detected a malicious connection in 83% of the cases, but the deviation is very high (30%). The comparison of results obtained by the neural network and rough sets show that these methods have

Table 3. Efficiency of the rough set based IDS

	Correct	Incorrect	None
Total	80.00±11.16	18.67±10.24	1.33±02.67
0	78.18±12.08	21.11±11.87	0.71±02.14
1	83.33±30.73	13.33±30.55	3.33±10.00

Figure 2. Rough-neuro hybrid approach for output classification



high variance which is a potential danger for the system. So, there is a need for a method that reduces the variance of prediction. We utilize the synergism of the two methods and propose the use of the integrated method called rough-neuro for improving the precision of the system.

Rough-Neuro Approach for IDS

In this section, we propose an integrated model for IDS called rough-neuro. The focus is on retaining the mean and decreasing the variance of prediction (see Figure 2).

We use a reduced dataset for analysis with the proposed neural network model. The reduction of dataset is done using rough set. So the steps involved in this hybrid model are:

- Reduce the dataset using rough set theory. Find all the reducts of the dataset and obtain the union of all the sets of the reducts. We take the attributes of this set as the attributes of the reduced dataset.
- Use the neural network model on the reduced dataset to develop an IDS.

The results show that the best efficiency of the hybrid model is 86%, whereas the average efficiency is 79.4%. There is a slight reduction in the best case efficiency because some information is lost since we use reducts to get a reduced set of data. There is no statistical difference in the average efficiency as compared to the other methods. The noticeable point is the reduction in the standard deviation to 3.15%. This is due to the elimination of unwanted information, by enriching the training data by using reducts.

Results and Discussion

In this section, we compare the results obtained by the three methods, GA-based neural network, rough set, and rough-neuro methods. This includes analysis and interpretation of the results.

Table 4. Efficiency comparison

Method	Best	Average	Worst	Std dev
1	94.00	80.00	62.00	11.160
2	92.00	80.40	64.00	08.356
3	86.00	79.60	74.00	3.627

The important performance parameters of IDS are efficiency and its standard deviation. If the system is mission critical, then the worst case efficiency of an IDS also is considered as an important parameter. The results of each method are tabulated in Table 4.

We have given a number to each method for our convenience of easy representation. The numbers given are:

1. Rough Set Method
2. Neural Network Method
3. Rough-Neuro Method

We first compare the efficiency of the methods based on its best case, worst case, and average case. As discussed, we take standard deviation as a parameter of comparison.

We can see that the GA-based neural network has the highest average efficiency. But at the same time, it comes with the undesired effect of high variance in the prediction. The rough set is a better approach if we consider only the best case efficiency. The network systems security people would go in for the rough-neuro method because of the low variation and a good average case efficiency performance.

Table 5 gives the error in the classification, which is commonly called as Type I error (normal connection is interpreted as malicious one) and Type II error (malicious connection is interpreted as normal). It is clearly evident that for a secured system, Type II error is more harmful than Type I error.

Table 5. Comparison of errors

Method	Type I Error		Type II Error	
	Mean	Std. Dev.	Mean	Std. Dev.
1	21.11	11.87	13.33	30.55
2	22.45	10.85	12.35	18.50
3	20.45	8.54	10.58	12.50

Table 6. Comparison of efficiency - Z test

Methods	Z (calc.)	Z (table)	Verdict
1 & 2	0.366	1.96 ($\alpha = 0.05$)	The comparison of means indicate no significant difference
1 & 3	0.280		
2 & 3	0.723		

Table 7. Comparison of standard deviation - F test

Methods	F (calc.)	F (table)	Verdict
1 & 2	1.782	1.861 ($df = 58$) ($\alpha = 0.01$)	Comparison of variance prove a significant difference in favour of rough-neuro
1 & 3	9.610		
2 & 3	5.393		

We should have a very small Type II error, since the opportunity cost of the system compromise is much more than that of the system being not available to a normal user. Rough-neuro method again is the best performer if we choose this criterion. Also, the standard deviation for this method is the lowest. Comparison of the average efficiency for different methods is presented in Table 6.

But what really determines the intrusion detection is the variation that each method has to offer. So we analyze the deviations, which are caused by each method. This is presented in Table 7.

We have used the best methods available for intrusion detection, and statistically we cannot say that there is a difference in the average efficiency of any of these three methods. The variation of rough-neuro method is significantly less than that of the other two methods at 99% confidence level. The rough-neuro method outperforms the other methods with respect to the worst case efficiency and standard deviation, and hence should be the chosen one in intrusion detection systems.

Conclusion and Future Research

In this chapter, we have discussed the data mining approach for IDS. We have given a brief introduction, with proper references, to each of the fields mentioned to help the readers brush up their concepts. This is in no way complete, and readers should refer to books in each particular area to know the intricacies of such approaches.

The neural network can perform better by choosing a suitable architecture and learning algorithm. The focus of this paper is to come up with a non-gradient learning algorithm so that the chances of overcoming the local minimum are enhanced. A single layer feedforward network with GA-based supervised learning provides promising results. The comparison with rough sets shows similar results. Utilizing the complementary

characteristics, we have shown how an integrated approach can reduce the attributes and generate a better solution.

Data mining techniques are data-dependent, and a good set of training data is necessary for effective decision-making. We have used the KDD cup data and compared the proposed methods. It would be good to check the variability of these techniques across varied sets of data. Application of the proposed methods with the necessary modification to other fields would be a good extension of this work. One also can come up with efficient algorithms that relax the burden of neural network training. Building robust IDS by evaluating the approach for different noisy level conditions is an important issue for future study.

References

Balajinath, B., & Raghavan, S.V. (2001). Intrusion detection through learning behavior model. *Computer Communications, 24*(12), 1202-1212.

Bhaskar, T., Kamath, N., & Moitra, S.D. (2004). A hybrid model for network security systems: Integrating intrusion detection system with survivability. *Working Paper Series 507/2004.* Indian Institute of Management Calcutta.

Bonifacio, J.M., Cansian, A.M., & Carvalho, A.C. (1997). Neural networks applied to intrusion detection systems. In *Proceedings of the International Conference on Computational Intelligence and Multimedia Application* (pp. 276-280).

Cai, Z., Guan, X., Shao, P., Peng, O., & Sun, G. (2003). A rough set theory based method for anomaly intrusion detection in computer network systems. *Expert Systems, 20*(5), 251-259.

Chalmers, D. (1990). The evolution of learning: An experiment in genetic connectionism. In *Proceedings of 1990 Connectionist Model Summer School.* Morgan Kaufmann.

Chebrolu S., Abraham, A., & Thomas, J.P. (2004). Feature deduction and ensemble design of intrusion detection system. *Computers and Security* (forthcoming).

Crosbie, M., & Spafford, G. (1995). Applying genetic programming to intrusion detection. In *Proceedings of AAAI 1995 Fall Symposium.*

Denning, D.E. (1987). An intrusion detection model. *IEEE Transactions on Software Engineering, 13*(2), 222-232.

Garfinkel S., & Spafford, E. (1991). *Practical UNIX security.* O'Reilly and Associates Inc.

Goldberg, D.E. (1989). *Genetic algorithms in search, optimization and machine learning.* Addison-Wesley.

Hassoun, M.H. (1998). *Fundamentals of artificial neural networks.* India: Prentice Hall of India.

Heady R., Luger, G., Maccabe, A., & Servilla, M. (1990). *The architecture of network-level intrusion detection system, technical report, CS90-20.* University of Mexico.

Holland, J.H. (1975). *Adaptation in natural and artificial systems.* Ann Arbor: University of Michigan Press.

Joo, D., Hong, T., & Han, I. (2003). The neural network models for IDS based on the asymmetric costs of false negative errors and false positive errors. Expert *Systems with Applications, 25*(1), 69-75.

Koza, J.R. (1992). *Genetic programming: On programming of computers by means of natural selection.* Cambridge, MA: The MIT Press.

Lee, W., Stolfo, S.J., & Mok, K. (1999). A datamining framework for building intrusion detection model. In *IEEE Symposium on Security and Privacy,* 120-132.

Li, R., & Wang, Z. (2004). Mining classification rules using rough sets and neural networks. *European Journal of Operational Research. 157,* 439-448.

Lippmann, R., & Cunningham, R.K. (2000). Improving intrusion detection performance using keyword selection and neural networks. *Computer Networks, 34,* 597-603.

Lunt, T.F. (1993). A survey of intrusion detection techniques. *Computer and Security, 12*(4), 405-418.

Mukherjee, B., Heberline, L.T., & Levitt, K.N. (1994). Network intrusion detection. *IEEE Network, 8*(3), 26-41.

Pal, S.K., & Skworon, A. (Eds.) (1999). *Rough fuzzy hybridization: A new trend in decision making.* Springer.

Pawlak, Z. (1982). Rough sets. *International Journal of Computers and Informations Sciences, 11,* 341-356.

Pawlak, Z. (1991). *Rough sets: Theoretical aspects and reasoning about data.* Kluwer Academic Publisher.

Pawlak, Z. (1997). Rough set approach to knowledge based decision support. *European Journal of Operational Research, 99,* 48-57.

Russel, D., & Gangeni, Sr., G. (1991). *Computer security basics.* O'Reilly and Associates.

Williams, R.L., & Sharda, R. (1994). Bankruptcy prediction using neural networks. *Decision Support Systems, 11,* 545-557.

Yahia, M., Mahmod, R., Sulaiman, N., & Ahmad, F. (2000). Rough neural expert systems. *Expert Systems with Applications, 18*(2), 87-99.

Zhang, C., Jiang, J., & Mohamed, K. (2004). Intrusion detection using hierarchical neural networks. *Pattern Recognition Letters* (forthcoming).

Zhu, D., Premkumar, G., Zhang, X., & Chu, C.-H. (2001). Data mining for intrusion detection: A comparison of alternative methods. *Decision Sciences, 32*(4), 635-660.

Chapter XVI

Cooperative AI Techniques for Stellar Spectra Classification: A Hybrid Strategy

Alejandra Rodríguez, University of A Coruña, Spain

Carlos Dafonte, University of A Coruña, Spain

Bernardino Arcay, University of A Coruña, Spain

Iciar Carricajo, University of A Coruña, Spain

Minia Manteiga, University of A Coruña, Spain

Abstract

This chapter describes a hybrid approach to the unattended classification of low-resolution optical spectra of stars. The classification of stars in the standard MK system constitutes an important problem in the astrophysics area, since it helps to carry out proper stellar evolution studies. Manual methods, based on the visual study of stellar spectra, have been frequently and successfully used by researchers for many years, but they are no longer viable because of the spectacular advances of the objects collection technologies, which gather a huge amount of spectral data in a relatively short time. Therefore, we propose a cooperative system that is capable of classifying stars automatically and efficiently, by applying to each spectrum the most appropriate

method or combined methods, which guarantees a reliable, consistent, and adapted classification. Our final objective is the integration of several artificial intelligence techniques in a unique hybrid system.

Introduction

This chapter is part of a global project devoted to the study of the last phases of stellar evolution. Evolutionary studies are an essential part of astrophysics because they allow us to discover and follow the temporal changes of the physical and chemical conditions of the stars. The general objective of our project is the development of an automatic system for the determination of the physical and chemical stellar parameters (spectral type, luminosity, temperature, metallicity, etc.) through optical spectroscopy and artificial intelligence techniques.

Spectroscopy is among the most powerful, currently available techniques for the study of stars and, in particular, their physical conditions (temperature, pressure, density, etc.) and chemical components (H, He, Ca, K, etc.). In general terms, a stellar spectrum consists of a black body continuum light distribution, distorted by the interstellar absorption and reemission of light, and by the presence of absorption lines, emission lines, and molecular bands (Zombeck, 1990).

The stellar spectra are collected from telescopes with appropriate spectrographs and detectors. Observers collect the flux distribution of each object and reduce these data to obtain a one-dimensional spectrum calibrated in energy flux ($erg^{-1}cms^{-2}s^{-1}Å^{-1}$) and wavelength (Å).

As part of the global project, we have collected a large sample of optical stellar spectra from astronomical observations carried out at several telescopes. Once the spectra of a homogeneous sample of stars are collected and reduced, the study of the distribution of spectral types and the analysis of spectral data can help to understand the temporary change of the physical conditions of stars from a statistical point of view, and therefore, to learn about their evolution. This is why spectral classification is one of the fundamental aspects of the evolutionary study of stars, and a phase that must be carried out in a fast, efficient, and accurate way.

In order to extract useful information from the individual spectra and to study the stellar evolution in the whole sample, we must complete a solid and systematic spectral classification of our collected spectra in the current Morgan-Keenan system (MK).

The MK classification system was firstly proposed in 1943 by Morgan, Keenan, and Kellman and has experienced many revisions ever since (Morgan, 1943). This two-dimensional system is the only one that is widely used for stellar classification. One of its main advantages is that MK classifications are often static because they are based on the visual study of the spectra and on a set of standard criteria. However, the same spectra can be classified differently by different experts and even differently by the same person at different times. This classification system quantifies stellar temperatures and levels of luminosity. Stars are divided into groups (i.e., spectral types) that are mainly

based on the strength of the hydrogen absorption lines and on the presence or absence of some significant lines of Ca, He, Fe, and molecular bands. The temperature of the stars is divided in a sequence called OBAFGKM, ranging from the hottest (type O) to the coolest (type M) stars. These spectral types are further subdivided by a decimal system, ranging from 0 (hottest) to 9.5 (coolest). In addition, a luminosity class (from I to V) is assigned to the star, which depends on the intrinsic stellar brightness. That is, the hottest star of the MK system would be of spectral type O0 and the coldest would be a M9 star.

Table 1 illustrates the main properties of each spectral type in the MK standard classification system.

Any classification system should hold a compromise between maintaining the full information of the spectra and the need for a compact summary of this information. An optimal summary is obviously obtained by a study of the spectral type and luminosity in the MK system.

The estimation of the spectral type and luminosity of stars is often carried out by human experts, who analyze the spectra by hand, with no more help than their own experience. These manual analyses usually lead to a MK classification of the spectra. The manual classification techniques are often based on the visual study of the spectra and on a set of standard criteria (Zombeck, 1990).

The human classifiers select a reference catalogue of previously classified spectra that is used as a reliable guide in the whole process. In order to compare the stars that are going to be classified with those of the reference catalogue, it is essential to normalise all the spectra and isolate the continuous component (affected by interstellar reddening). Only then, the comparison can be focused on the information of the spectral lines. At this point, the standardised spectra must be scaled to make their magnitudes equivalent to the stars of the reference catalogue.

On the basis of the scaled and normalised spectra, human experts try to determine the spectral type and the luminosity in the MK system. They measure and study the relation between some absorption lines and the depth of certain relevant molecular bands, as a result they obtain the first classification of each star.

Interstellar reddening complicates the use of the continuum shape to determine the photosphere effective temperature. Temperatures are measured by calibrating the relative strength of individual spectral lines, mostly from atomic H and He energy

Table 1. Main spectral features in the MK system

Type	Color	Prominent Lines
O	Bluest	Ionized He
B	Bluish	Neutral He, Neutral H
A	Blue-white	Neutral H
F	White	Neutral H, Ionized Ca
G	Yellow-white	Neutral H, Strongest Ionized Ca
K	Orange	Neutral Metals (Ca, Fe), Ionized Ca
M	Red	Molecules and Neutral Metals

transitions. Additionally, various groups of lines (e.g., sulphur or nitrogen) can be used to determine electron densities/pressures and other physical properties. Line widths reflect rotational velocities, velocity dispersion, or microturbulence, and line displacements by the Doppler effect show the stellar radial velocity.

The initial MK classification is refined by superposing the unclassified spectra and those of the reference catalogue that correspond to the same spectral type, until the spectral subtype is finally determined.

Although this manual method of classification has been used successfully for many years, it is no longer feasible, since the current collection technologies allow us to obtain a huge amount of spectral data in a relatively short time. The manual classification of all the spectra that are currently available would involve a considerable increase in time and human resources; therefore, it is highly advisable to optimize the manual procedure by means of automatic, fast, and efficient computational techniques.

In the course of the last few years, research in the field of spectral classification has been either focused on the need for the development of automatic tools or on the revision and improvement of the manual techniques.

As for the application of artificial intelligence techniques to the design of automatic classification systems, some well-known previous works also have applied artificial neural networks to the problem of stellar classification (Weaber, 1995; Gulati, 1998), obtaining classifications with diverse resolution grades. Our research team has contributed to this research line with the development of various fuzzy experts systems for the classification of super giant, giant, and dwarf stars. A complete description of our previous works can be found in Rodriguez (2004). The obtained results led us to extend these systems to stars of different luminosities and to add new computational techniques, such as artificial neural networks and clustering algorithms, in order to refine the automatic processing of spectra.

Our current objective is not to test models or techniques that have already demonstrated their suitability in this problem, but rather to combine all the techniques in a unique classification tool. We intend to formalize a hybrid system that is able to determine the most appropriate method for each spectrum type and to obtain online MK classifications through an Internet Stellar Database (http://starmind.tic.udc.es).

The following sections start by describing the spectral data that were used to design and test the automatic classification techniques. Then, we describe the morphological analysis algorithms that were applied to the spectra before presenting them to the automatic techniques. Finally, we present the different artificial intelligence models that were implemented and we contrast their results.

Astronomical Data

We have chosen a complete and consistent set of 258 spectra that cover all the types and luminosities of the MK system, in order to design the artificial models that will be applied to the classification problem. This set is sufficiently representative because it offers a

continuous transition of the spectral features between each spectral type and its adjacent types. The selected spectra were previously analyzed and corrected by human experts who collaborate in the project. We have used the public catalogues of Silva (1992) — 28 spectra sampled in the range of 3,500 to 8,900 Å with 5 Å of spectral resolution — Pickles (1998) — 97 spectra sampled in the range of 1,150 to 25,000 Å with 5 Å of spectral resolution — and Jacoby (1984) — 133 spectra sampled in the range of 3,510 to 7,426 Å with 1.4 Å of spectral resolution.

In order to guarantee the generalization of the designed and implemented models, we have built the training or design set with approximately 50% of the spectra of each spectral type, leaving around 15% for validation and the remaining 35% to evaluate the classification capability of each model.

The artificial intelligent techniques of this experimentation have been designed and tested so as to consider both full spectra and spectral parameters as input patterns. Before presenting the spectra to the automatic techniques, we carry out a morphological analysis of all the spectra in order to obtain the values of the parameters that characterize each spectrum separately.

Morphological Analysis

The patterns that are presented to the selected models were obtained automatically by means of signal processing techniques that measure the spectral peculiarities (absorption and emission lines, spectral energy, molecular bands, etc.).

In particular, we measure the 25 spectral features that are described in Table 2. These spectral parameters can be grouped into three general types:

- **Absorption and emission lines:** including hydrogen, helium and metallic lines (Ca, K, Fe, etc.).
- **Molecular bands:** hydrogen and carbon absorption bands.
- **Rates between lines:** CH-K rates, He-H rates, and so forth.

The signal processing algorithms used to obtain the spectral parameters are mainly based on the spectral continuum estimation and the energy measurement.

From a morphological point of view, an absorption line is a descending (ascending for emission) deep peak that appears in an established wavelength zone. To accurately calculate the intensity of each line, we carry out an estimation of the local spectral continuum. We smoothen the signal with a low pass filter, excluding the peaks in an interval around the sample where the line was detected. This filter is implemented by a five-point moving average method that selects the five more stable fluxes. That is

Table 2. Spectral classification parameters

Parameter	Description
Band 1	5005 ± 055 Å
Band 2	6225 ± 150 Å
Band 3	4435 ± 070 Å
Band 4	5622 ± 180 Å
Band 5	5940 ± 135 Å
Band 6	6245 ± 040 Å
Band 7	6262 ± 130 Å
Band 8	6745 ± 100 Å
Band 9	7100 ± 050 Å
Line Ca II (K)	3933 Å
Line Ca II (H)	3968 Å
Line CH band	4300 Å
Line H Iγ	4340 Å
Line H Iδ	4102 Å
Line He I	4026 Å
Line He II	4471 Å
Line H Iβ	4861 Å
Line H Iα	6563 Å
Main Bands	$\sum_{i=1}^{i=2} Band_i$
Secondary Bands	$\sum_{i=3}^{i=9} Band_i$
Rate K-H	Ca II K / Ca II H
Rate CH- H Iγ	CH band / H Iγ
Rate H Iδ - HeI	H Iδ / He I
Rate H Iδ - HeII	H Iδ /He II
Energy	Flux Integral

$$C_j = \left(\frac{\sum_{j-n}^{j+n} E_i * X_i}{N} \right) \tag{1}$$

where Cj is the estimation of the continuum for sample j, Ei is the flux in sample i, N is the number of values used in the moving average method to calculate the local spectral continuum, and X is a binary vector that indicates the representative fluxes of the spectral continuum in the zone. This means that $Xi = 1$ if Ei is a flux value representative of the local spectral continuum, and $Xi = 0$ if Ei is a peak. The intensity is positive for the absorption lines and negative for the emission lines.

A molecular band is a spectral zone where the flux suddenly decreases from the local continuum during a wide lambda interval. For the molecular bands this means that we only have to measure their energy to decide if they are significant enough. In this case, the upper threshold line for each band is calculated by means of linear interpolation between the fluxes in the limits of the interval defined for each band. The area between this line and the axis of abscissas is then calculated with a discrete integral, and the area that

surrounds each band is calculated by integrating the flux signal between the extremes of the band. Finally, the flux of the band is obtained by subtracting both calculated energies. That is

$$B_{lr} = \int_{l}^{r} L(\lambda_i) - \int_{l}^{r} E(\lambda_i) \quad (2)$$

where *Blr* is the flux of the band between the samples *l* and *r*, *L* is the projection line, *E* is the flux function, *ë* the wavelength, *l* the left limit of the band and *r* the right limit. Since the obtained value becomes more negative as the band becomes deeper and wider, positive or negative values close to zero are not considered as bands.

Classification Techniques

Among the existing techniques of artificial intelligence, expert systems (ES) and artificial neural networks (ANN) seem to be most appropriate to approach the problem of stellar classification. Expert systems can reproduce the reasoning of experts in order to classify spectra; neural networks, capable of learning the intrinsic relations of the patterns with which they were trained, have already proven their efficiency in classification problems (Bazoon, 1994). We also have implemented clustering algorithms to perform the sensibility analysis of the input spectra, although this technique is not currently used to obtain MK classifications of stars.

Our main objective is to integrate all the designed and implemented techniques in a unique hybrid system capable of applying the best classification method to each input spectrum. The developed system includes two different tools: a spectral analyzer and a stellar classifier.

The spectral analyzer makes an exhaustive morphological analysis of the spectra, using the described algorithms to obtain a numerical parameterisation. It is developed in Builde C++ (Hollingworth, 2000) and integrates ad hoc ActiveX components for the visualization of spectra.

The analyzer retrieves the spectral data from a relational database that stores and structures the information from human and bibliographic sources. The stellar database is implemented by means of the PostgreSQL Database Management System running under Linux (Momjiam, 2000). At present, approximately 400 spectra of our survey are stored in the database, and they will be soon available via the Internet.

The stellar classifier is based on the development of the different artificial models that were chosen to approach the MK classification of stars. We used both the spectral parameters obtained by the spectral analyzer and the full spectral data to build the input patterns of the artificial intelligence techniques. The neural networks were implemented with the Stuttgart Neural Network Simulator (SNNS, 2001), the clustering algorithms were

developed with MATLAB v.6.5.1 (Hahn, 2002), and the expert systems were implemented in OPS/R2 (Forgy, 1986).

At present, we are developing a Web site that will make the stellar classifier available through the Internet. Our main purpose is to allow users worldwide to classify their stellar spectra online in a fast, efficient, and comfortable way.

After analyzing the performance of each technique separately, we implemented the best models in C++ by integrating them with the spectral analyzer, which provides us with a unique tool for the processing and classification of the optical spectra of stars.

The next sections describe the different models that are integrated into the stellar spectral classifier.

Expert Systems

This first approach proposes the implementation of a knowledge-based system that provides the user with a comfortable tool for the processing of stellar spectra . We have integrated signal processing, knowledge-based, and fuzzy techniques, obtaining a very satisfactory emulation of the current manual process. Our final system is able to classify stars with a success rate very similar to the agreement percentage between experts in the field (approximately 80%), and allows two classification modalities: spectra with no given luminosity class, and spectra of stars with a well-known luminosity level.

As a previous step toward the design of the expert system, we carried out a sensibility analysis of the classification parameters in order to define the different fuzzy sets, variables, and membership functions. In this study, we have analyzed the parameters of the spectra from the reference catalogue, using the aforementioned algorithms and determining the different spectral types that each parameter discriminates. Some parameters that seemed to be suitable were discarded, whereas others, which are not explicitly considered in the manual classification, were included, for example, the additions of band fluxes: no molecular band, by itself, was found suitable to determine the global temperature (early, intermediate, late) for all the stars in the reference catalogue; however, we found a good discriminant between early, intermediate, and late stars, which is the addition of several relevant bands. This new parameter can divide the stars from the catalogue into the three global temperature groups: Since some stars that belong to the same group present a greater value in some bands, and in other stars the highest value corresponds to a different band, the addition solves these problems.

As a final result of this analysis, we have defined as many fuzzy variables as classification levels (global, type, and subtype) for each luminosity class; we also have defined the fuzzy sets and membership functions determined by the values of the spectral features in the guiding catalogue spectra.

The developed expert system stores the information that is necessary to initiate the reasoning process in the facts base. This descriptive knowledge of the spectra is represented by means of frames (Sowa, 1999), that is, objects and properties structured by levels. This model was chosen because it is the simplest and most adequate to transfer the analysis data to the classification module and allows us to establish the equivalence

between analysis data and knowledge. The knowledge of the facts base includes general information, such as the names of the stars and the results of the morphological analysis, that is, the values of the classification parameters.

The real parameters of spectral classification and the limit values of each type and subtype were included in the expert system in the shape of fuzzy rules. The rules base is that part of the system where the human classification criteria are reproduced. We have adopted IF-THEN production rules for the implementation of this module, because they allow us to manage the uncertainty and imprecision that characterize human reasoning in this field.

The conditions of these rules refer to the values of the parameters stored in the current facts base (working memory). The conclusions allude to three levels of spectral classification: global (late, intermediate, early), spectral type, and luminosity, and as such, the module communicates actively with the facts base.

To decide what rule to apply at each moment, we used the means-end analysis strategy (MEA) (Valette-Florence, 1994): Basically, among the rules that were incorporated last into the working memory, this strategy chooses the not executed rule that has the largest number of patterns. The production rules are linked in a forward reasoning, guided by objectives. The strategy used for the reasoning process combines guided reasoning methods with a method based on truth values. The rules also have associated credibility factors that were obtained from interviews with experts and from the bibliography of this field.

We used the Shortliffe and Buchanan methodology (Buchanan, 1984) to create an evolution that includes fuzzy sets and membership functions that are contextualized for each spectral type and allow superposition between them. The applied inference method is Max-product, which combines the influence of all the active rules and produces a smooth, continuous output. In our approach, the credibility factor of each rule has also been considered as another truth value. The defuzzification of the data into a crisp output was accomplished by the fuzzy-centroid method (Mendel, 1995). With this mixed strategy, we achieved a remarkable adaptation to human reasoning, able to successfully handle the imprecision and uncertainty implicit in the manual classification process. In addition, we obtained the spectral classification of stars with a probability value that indicates the grade of confidence.

This part of the spectral classifier was developed in OPS/R2 (Forgy, 1986) and integrated with the analyzer by means of dynamic link libraries (DLL).

An additional research topic consisted in improving the implemented system by applying the results of the best neural models, and will be described in the next sections. The weights of the output layer units were analyzed so as to determine, for each spectral type, which input parameters have more influence on the output. The normalized values of the higher weights were included in the expert system in the shape of credibility factors of the rules that correspond to the most influential parameters for each spectral type. This modification of the reasoning rules (using the weights values of the trained neural networks) resulted in a slightly significant improvement of the performance of the original expert systems (around 2%).

Artificial Neural Networks

The neural networks of this approach are based on both supervised and non-supervised learning models (Haykin, 1994). In particular, backpropagation, Kohonen, and radial basis functions (RBF) networks were implemented.

We have tested three backpropagation learning algorithms (standard, momentum, and quick) for the spectral types, spectral subtypes, and luminosity classes.

We also have tested RBF networks; networks based on radial basis functions combine non-supervised learning for hidden units and supervised learning in the output layer. The hidden neurons apply a radial function (generally Gaussian) to the distance that separates the input vector and the weight vector that each one stores, called centroid.

Finally, we also have implemented Kohonen networks. The self-organizing map (SOM) algorithm of Kohonen is based on non-supervised learning. SOMs are a unique class of neural networks, since they construct topology-preserving mappings of the training data where the location of a unit carries semantic information (Kohonen, 2001).

The training, validation, and testing patterns that are presented to the neural networks were obtained automatically by adding the necessary functions to the spectral analyzer developed in the expert systems approach. Once the input values are obtained by the spectral analyzer, they must be normalized and presented to the neural networks. Our study standardizes the inputs of the network by means of a contextualised and specific sigmoidal function for each parameter. This function normalizes the input parameters in the [0, 1] interval and centers and scales the distribution function of each parameter properly. The different topologies that were implemented for the three learning algorithms are shown in Table 3.

Our observations of the networks' behaviour show that they converge when the mean square error (MSE) is equal or inferior to 0.05 and the net becomes stable. If the training continues after having reached this rate of MSE, the net is over-trained and its performance decreases. In the SNNS simulator used to train the networks, an output greater than 0.5 is equivalent to 1, otherwise to 0. In the analysis of the results, we have not considered the outputs near 0.5 as successes (from 0.45 to 0.55).

The backpropagation topology that has resulted in a better performance corresponds to a network trained with 25 spectral parameters as input layer and three hidden layers of 10, 5, and 3 units. The best results for Kohonen networks were achieved by maps of 12x12 units. As for the RBF networks, the best topology corresponds to a network trained with 25 spectral parameters as input layer and 8 neurons in the hidden layer.

Clustering Techniques

In order to refine the classifications made by artificial neural networks and expert systems, we have implemented statistical clustering techniques and applied them to the problem of spectral classification. In particular, we have implemented the K-means, Max-Min, and Isodata non-hierarchical clustering methods (Kaufman, 1990).

Table 3. Topologies tested for backpropagation, Kohonen, and RBF networks (shows best networks)*

Network	Input Patterns	Hidden Layer
BP Type	Spectral parameters	10
BP Type	Spectral parameters	5x5
BP Type	Spectral parameters	10x10
BP Type	Spectral parameters	10x5x3*
BP Type	659 flux values	100x50x10x3
BP Luminosity	Spectral parameters	10x10
BP Luminosity	659 flux values	100x50x10x3
RBF Type	Spectral parameters	16
RBF Type	Spectral parameters	8*
RBF Type	Spectral parameters	4
RBF Type	659 flux values	124
RBF Luminosity	Spectral parameters	8
RBF Luminosity	659 flux values	124
Kohonen Type	Spectral parameters	2x2
Kohonen Type	Spectral parameters	12x12*
Kohonen Type	Spectral parameters	24x24
Kohonen Luminosity	Spectral parameters	2x2

This approach uses the spectral parameters, obtained through the morphological analysis algorithms, as well as the full spectra. In addition, two different versions of each algorithm with 6 and 12 initial clusters were implemented.

Although the implemented clustering methods have achieved remarkable success rates in stellar spectra classification, this technique was mainly applied to analyze the sensibility of the spectral parameters that were used to classify the stellar spectra.

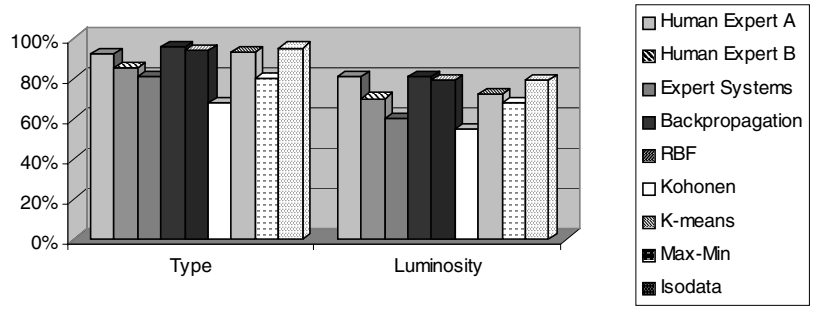
Results

This section makes the final comparison between the three applied global approaches: expert systems, clustering techniques, and artificial neural networks. We selected the neural models of each type with the best performance and classified, by means of clustering algorithms and expert systems, the 100 spectra that were used to test these networks. Figure 1 contrasts the behaviour of the automatic techniques and that of two human experts who collaborated on this project.

The backpropagation and RBF networks, as well as K-means and isodata algorithms, obtained a high success rate of approximately 95%. The Kohonen model obtained a low success rate in all its implementations, which could be due to the size of the training set: This network type must cluster the data and therefore needs a training set that is large enough to extract similarities and group the data.

Although the final results for the proposed classification methods seem to be similar, an exhaustive study has revealed some interesting peculiarities; for example, we have observed that each technique reached its worst results for the B and M spectral types,

Figure 1. Final performance for 100 testing spectra



that is, the hottest and coolest stars, respectively, and indeed, most of the grouping algorithms include these spectra in the same cluster. This fact led us to review the spectral parameters that were being used to design the models: We discovered that B stars usually present great emission lines in zones where a molecular band is expected, which means that the automatic techniques are unable to differentiate between them. Our hybrid approach tries to solve these problems by making a previous global classification of the star and then selecting the best method to classify the spectra.

Hybrid Strategy

Our hybrid approach consists of choosing, among all the described techniques, those methods that present the best performance for each classification level. The final system is mainly based on an expert system that determines the global type of each star and that, according to the type, sends the spectra to different neural networks or clustering algorithms in order to obtain their spectral type as well as their luminosity level. The expert system classification also is used, as an additional information, for those cases in which the other methods are unable to discriminate. The final system conceptual model is shown in Figure 2.

The final implemented system includes a friendly interface, which is shown in Figure 3. It allows the users to select the spectra, visualize them, perform different analyses, and classify as many spectra as they want in a fast, comfortable, and simple way, which is the global objective of this computational approach.

Figure 2. Final system scheme

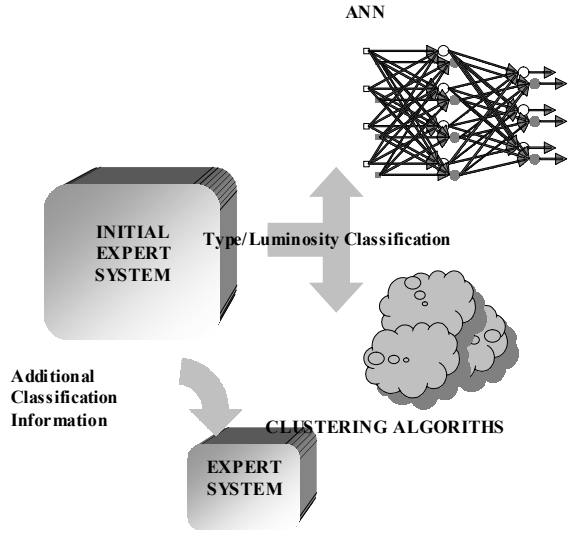
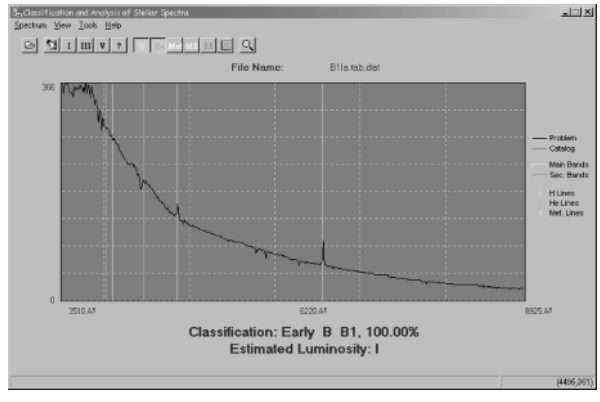


Figure 3. Final system interface



Conclusion

This chapter has proposed a hybrid and cooperative approach to the problem of MK classification of stars. Our contribution is based on the development of an unattended system that morphologically analyzes and automatically classifies the optical spectra of stars.

We described several artificial intelligence models and analyzed their performance and results to discover the best approach to the classification of each type of spectrum. In

our research, we combined signal processing techniques, expert systems, artificial neural networks, and clustering algorithms.

The best techniques reached a success rate of approximately 95% for a sample of 100 testing spectra, which, compared to manual classifications, corresponds to a performance increase of approximately 10% (since the experts reached an agreement percentage of approximately 87% of the spectra).

By using the additional classification information provided by the clustering techniques, we have opened a new research line for the refinement of automatic classifications, especially for spectral types B and M; the implemented clustering techniques allow us to perform a sensibility analysis of the spectral parameters used to classify stellar spectra in the neural networks and expert systems approach.

Finally, all the artificial intelligence techniques were integrated into a hybrid system that has resulted in a versatile and flexible automatic method for the classification of stellar spectra. In this way, the proposed system can achieve a better adaptation to the classification problem, since each spectrum is processed with the most appropriate technique according to its specific features.

For the evaluation period of the proposed models, we could count on the essential collaboration of experts from the areas of astronomy and astrophysics of the University of A Coruña.

At present, we are analyzing functional networks in order to determine the suitability of this artificial technique in stellar classification. We also are working on our stellar database and are developing a Web site to make the automatic classification system available through the Internet so as to allow world-wide users to analyze and classify stellar spectra online in a fast, efficient, and comfortable way.

The authors acknowledge support from grants AYA2000-1691 and AYA2003-09499, extended by the Spanish *Ministerio de Ciencia y Tecnología*.

References

Bazoon, M., Stacey, D., Cui, C., & Harauz, G. (1994). A hierarchical artificial neural network system for the classification of cervical cells. *Proceedings of the IEEE International Conference on Neural Networks (ICNN '94).*

Buchanan, B., & Shortliffe, E. (1984). *Ruled-based expert systems.* Adisson-Wesley.

Forgy, C.L. (1986). *The OPS user's manual, system version 2.2.* Production Systems Technologies.

Gulati, R.K., & Singh, H.P. (1998). Stellar classification using principal component analysis and artificial neural networks. *Monthly Notices Royal Astronomical Society, 295,* 312.

Hahn, B. (2002). *Essential MATLAB for scientists and engineers.* University of Cape Town.

Haykin, S. (1994). *Neural networks. A comprehensive foundation.* MacMillan College Pub.

Hollingworth, J., Butterfield, D., Swart, B., & Allsop, J. (2000). *C++ builder 5.0. Developer's guide.* SAMS.

Jacoby, G.H., Hunter, D.A., & Christian, C.A. (1984). A library of stellar spectra. *The Astrophysical Journal Suppl., 56,* 257-281.

Kaufman, L., & Rousseuw, P.J. (1990). *Finding groups in data.* John Wiley.

Kohonen, T. (2001). *Self-organizing maps* (3rd ed.). Springer.

Mendel, J.M. (1995). Fuzzy logic systems for engineering: A tutorial. *Proceedings of the IEEE, 83*(3), 345-377.

Momjiam, B. (2000). *PostgreSQL: Introduction and concepts.* Adisson-Wesley.

Morgan, W.W., Keenan, P.C., & Kellman, E. (1943). *An atlas of stellar spectra with an outline of spectral classification.* Chicago: University of Chicago Press.

Pickles, A.J. (1998). A stellar spectral flux library. 1150-25000 A. *Publications of the Astronomical Society of the Pacific, 110,* 863-878.

Rodriguez, A., Arcay, B., Dafonte, C., Manteiga, M., & Carricajo, I. (2004). Automated knowledge-based analysis and classification of stellar spectra using fuzzy reasoning. *Expert Systems with Applications, 27*(2), 237-244.

Silva, D.R., & Cornell, M.E. (1992). A new library of stellar optical spectra. *The Astrophysical Journal Suppl, 81*(2), 865-881.

Sowa, J.F. (1999). *Knowledge representation: Logical and computational.* Brooks Cole Publishing.

Stuttgart Neural Network Simulator. (2001). Retrievd from www-ra.informatik.uni-tuebingen.de/SNNS/

Valette-Florence, P. (1994). Introduction to means-end chain analysis. *Rech. Appl. Mark, 9,* 93-117.

Weaber, W., & Torres-Dodgen, A. (1995). Neural networks classification of the near-infrared spectra of A-type Stars. *The Astrophysical Journal, 446,* 300-317.

Zombeck, M.V. (1990). *Handbook of astronomy and astrophysics* (2nd ed.). Cambridge University Press.

Glossary

Accuracy: The measurement (or a prediction) error that is repeatable from trial to trial. Accuracy is limited by systematic (repeatable) errors.

Action Potential: Electric signal propagated over long distances by excitable cells (e.g., nervous and muscular ones); it is characterized by an all-or-none reversal of the membrane potential in which the inside of the cell temporarily becomes positive with regards to the outside; it has a threshold and it is conducted without decrement. It is also known as nerve impulse.

Activation Function: The time-varying value that is the output of a neuron.

Adaptive Learning Rate: A learning rate that is adjusted according to an algorithm during the training to minimise training time.

Aggregate: It is the inert filler material in concrete that permits good physical properties at a low cost.

Apoptosis: Form of programmed cell death caused by the activation of endogenous molecules leading to the fragmentation of DNA.

Architecture: A description of the number of layers in an artificial neural network, the activation function of each neuron, the number of neurons per layer, and the connections between layers.

Artificial Intelligence (AI): Artificial intelligence is most often defined as "the ability of a computer to perform activities that are normally thought to require intelligence." As science, AI is often branched into several lines of research based on the methods used to achieve the "intelligence," including expert systems, fuzzy logic, and artificial neural networks.

Artificial Neural Network (ANN): An artificial neural network is an information-processing system that is based on generalizations of human cognition or neural biology and are electronic or computational models based on the neural structure of the brain. The brain basically learns from experience. There are two main types of ANN, the first one with only feedforward connections is called feedforward ANN, and the second one with arbitrary connections without any direction, are often called recurrent ANN (RANN). The most common type of ANN consists on different layers, with some neurons on each of them connected with feedforward connections and trained with the backpropagation algorithm.

Astrocyte (Astroglia): A star-shaped cell, especially a neuroglial cell of nervous tissue that supports the neurons.

Autocorrelation: A signal correlated with itself. It is useful because the Fourier transform of the autocorrelation is the power spectrum of the original signal.

Autonomic nervous system: Part of the vertebrate nervous system that regulates involuntary action, such as the intestines, heart, and glands. It is divided into the sympathetic nervous system and the parasympathetic nervous system.

Axon: The usually long process of a nerve fiber that generally conducts neuron impulses away from the body of the nerve cell.

Backpropagation (Generalised Delta-Rule): A learning rule in which weights and biases are adjusted by error-derivate (delta) vectors backpropagated through the network. Backpropagation is commonly applied to feedforward multilayer networks. Sometimes this rule is called the generalized delta rule.

Basis Functions: The set of waveforms used by decomposition. For instance, the basic functions for the Fourier decomposition are unity amplitude sine and cosine waves.

Bayesian Framework: Assumes that the weights and biases of the network are random variables with specified distributions.

Bias: A neuron parameter summed with the neuron's weighted inputs and passed through the neuron's transfer function to generate the neuron's output.

Bias Vector: A column vector of bias values for a layer of neurons.

Blood-Brain Barrier (BBB): A physiological mechanism that alters the permeability of brain capillaries, so that some substances, such as certain drugs, are prevented from entering brain tissue, while other substances are allowed to enter freely.

Bond: A debt instrument that pays a set amount of interest on a regular basis. The amount of debt is known as the principal, and the compensation given to lenders for making such funds available is typically in the form of interest payments. There are three major types of bonds: corporate, government, and municipal. A corporate bond with a low credit rating is called a high-yield or junk bond.

Bond Rating: An assessment of the likelihood that a bond issuer will pay the interest on its debt on time. Bond ratings are assigned by independent agencies, such as Moody's Investors Service and Standard & Poor's. Ratings range from AAA or Aaa (highest) to D (in default). Bonds rated below B are not investment grade and are called high-yield or junk bonds. Since the likelihood of default is greater on such bonds, issues are forced to pay higher interest rates to attract investors.

Cell Body (Soma): The portion of a nerve cell that contains the nucleus but does not incorporate the dendrites or axon.

Cement: It is a kiln-dried and finely pulverized mixture of natural earth materials used as a bonding ingredient in concrete.

Central Nervous System: The portion of the vertebrate nervous system consisting of the brain and spinal cord.

Cepstrum: A rearrangement of "spectrum." Used in homomorphic processing to describe the spectrum when the time and frequency domains are switched.

Civil Engineering: A broad field of engineering that deals with the planning, construction, and maintenance of fixed structures, or public works, as they related to earth, water, or civilization and their processes. Most civil engineering today deals with roads, structures, water supply, sewer, flood control, or traffic.

Classification: An association of an input vector with a particular target vector.

Clustering: Clustering algorithms find groups of items that are similar. For example, clustering could be used by an insurance company to group customers according to income, age, types of policies purchased, and prior claims experience. It divides a set of data so that records with similar content are in the same group, and groups are as different as possible from each other. Since the categories are unspecified, this is sometimes referred to as unsupervised learning.

Competitive Layer: A layer of neurons in which only the neuron with maximum net input has an output of 1 and all other neurons have an output of 0. Neurons compete with each other for the right to respond to a given input vector.

Competitive Learning: The unsupervised training of a competitive layer with the *instar* rule or Kohonen rule. Individual neurons learn to become feature detectors. After training, the layer categorizes input vectors among its neurons.

Competitive Transfer Function: Accepts a net input vector for a layer and returns neuron outputs of 0 for all neurons except for the "winner," the neuron associated with the most positive element of the net input n.

Concrete: It is a mass of sand, gravel, crushed rock, or other aggregate bonded together by a hardened paste of hydraulic cement and water.

Conjugate Gradient Algorithm: In the conjugate gradient algorithms, a search is performed along conjugate directions, which produces generally faster convergence than a search along the steepest descent directions.

Connection: A one-way link between neurons in a network.

Connection weight: The weight of a link between two neurons in a network. The weight determines the effect that one neuron has on another.

Connectivity: The interaction level within a system, the structure of the weights in a neural network, or the relative number of edges in a graph.

Cortex: Outer layer of grey matter that covers the surface of each cerebral hemisphere.

Credit rating: A credit rating is an assessment by a third party of the creditworthiness of an issuer of financial securities. It tells investors the likelihood of default, or non-payment, of the issuer financial obligations.

Crossover: Genetic process by which two chromosomes paired up exchange a distal portion of their DNA.

Data clustering: A common technique for data analysis, which is used in many fields, including machine learning, data mining, pattern recognition, image analysis, bioinformatics, and search engines.

Data mining – Knowledge discovery in databases (KDD): An information extraction activity whose goal is to discover hidden facts contained in databases. Using a combination of machine learning, statistical analysis, modeling techniques, and database technology, data mining finds patterns and relationships of the data and infers rules that allow the prediction of future results. Typical applications include market segmentation, customer profiling, fraud detection, evaluation of retail promotions, and credit risk analysis.

Decision boundary: A line, determined by the weight and the bias vector, for which the net input n is zero.

Decomposition: The process of breaking a signal into two or more additive components. Often it is referred specifically to the *forward Fourier transform*, breaking a signal into sinusoids.

Delta vector: The delta vector for an ANN layer is the derivate of the network's output error with respect to that layer's net input vector.

Dendrite: Short fiber that conducts information toward the cell body of the neuron.

Distance function: A particular way of calculating distance, such as the Euclidean distance between two vectors.

Epoch: The presentation of the set of training (input and/or target) vectors to an ANN and the calculation of new weights and biases. Note that training vectors can be presented one at a time or all together in a batch.

Error ratio: A training parameter used with adaptive learning rate and momentum training of backpropagation ANNs.

Error vector: The difference between an ANN's output vector in response to an input vector and an associated target output vector.

Evolutionary Computation (EC): Encompasses methods of simulating evolution on a computer. The term represents an effort bring together researchers who have been working in closely related fields but following different paradigms.

Expert Systems: A branch of artificial intelligence, expert systems use a complex hierarchy of rules to perform "intelligent tasks" based on available data. Examples include disease diagnosis and financial analysis.

False Negative: One of four possible outcomes of a target detection trial. The target is present, but incorrectly indicated to be not present.

False Positive: One of four possible outcomes of a target detection trial. The target is not present, but incorrectly indicated to be present.

Feedback Network: An ANN with connections from a layer's output to that layer's input. The feedback connection can be direct or by pass several layers.

Feedforward Network: A layered ANN in which each layer only receives inputs from previous layers.

Forecasting: Making predictions about future performance on the basis of historical and current conditions data.

Function Approximation: The task performed by an ANN trained to respond to inputs with an approximation of a desired function.

Fuzzy Logic: A branch of artificial intelligence, whereas most computer systems use boolean logic where everything is made with 1 and 0, true and false, fuzzy logic uses "partial truth." This helps in dealing with some human concepts, such as the meaning of natural language (this 84% certainly means X) and quantification (someone who is 190 centimeters tall is 70% certainly "tall," someone who is 210 centimeters is by all measures 100% certainly tall).

Generalisation: A measure of how well an ANN can respond to new input information on which it has not been trained but which is related in some way to the training patterns. An ability to generalise is crucial to the decision-making ability of the ANN. An attribute of an ANN whose output for a new input vector tends to be close to outputs for similar input vectors in its training set.

Genetic Algorithm (GA): An algorithm used to find approximate solutions to difficult-to-solve problems through the application of the principles of evolutionary biology to computer science.

Genetic Programming (GP): A type of programming that uses the same properties of natural selection found in biological evolution. The general idea behind genetic programming is to start with a collections of functions and randomly combine them into programs; then run the programs and see which gives the best results; Keep the best ones (natural selection), mutate some of the others, and test the new generation; repeat this process until a clear best program emerges.

Gland: Organ that produces a secretion for use elsewhere in the body or in a body cavity or to be eliminated by the body.

Glia: Supporting tissue that surrounds and supports neurons in the central nervous system; glial and neural cells together compose the tissue of the central nervous system. Also named neuroglia, glial cells do not conduct electrical impulses, unlike neurons.

Global Minimum: The lowest value of a function over the entire range of its input parameters. Gradient descent methods adjust weights and biases in order to find the global minimum of error for an ANN.

Gradient Descent: The process of making changes to weights and biases, where the changes are proportional to the derivatives of the ANN error with respect to those weights and biases. This is done to minimise the ANN error.

Hebb Learning Rule: Historically, the first proposed learning rule for neurons. Weights are adjusted proportional to the product of the outputs of pre- and post-weight neurons.

Hebbian Learning: ANNs process that strengthens the association between two simultaneously active neurons.

Hidden Layer: A layer of an ANN that is not the ANN input or output.

Hopfield ANN: A particular example of an ANN capable of storing and recalling memories or patterns. All nodes in the ANN feed signals to all others.

Hydrology: Is the study of the occurrence, distribution, and movement of water on, in, and above the earth.

Immune System: Integrated body system of organs, tissues, cells, and cell products such as antibodies that differentiates self from no self and neutralises potentially pathogenic micro-organisms or substances.

Initialisation: The process of setting the network weights and biases to their original values.

Input Layer: A layer of neurons that receive inputs directly from outside the ANN.

Input Space: The range of all possible input vectors.

Input Vector: A vector of inputs presented to the ANN.

Input Weights: The weights connecting ANN inputs to the input layer.

Intrusion Detection System (IDS): A software tool used to detect unauthorised access to a computer system or network. This may take the form of attacks by skilled malicious hackers or Script kiddies using automated tools.

Kohonen Learning Rule: A learning rule that trains selected neuron's weight vectors to take on the values of the current input vector, in Kohonen ANN.

Layer: A group of neurons that have a specific function and are processed as a whole. The most common example is in a feedforward ANN that has an input layer, an output layer, and one or more hidden layers.

Layer Diagram: An ANN architecture figure showing the layers and the weight matrices connecting them. Each layer's transfer function is indicated with a symbol. Sizes of input, output, bias, and weight matrices are shown. Individual neurons and connections are not shown.

Layer Weights: The weights connecting layers to other layers. Such weights need to have non-zero delays if they form a recurrent connection.

Learning Algorithms (Supervised, Unsupervised): An adaptation process whereby synapses, weights of ANNs, classifiers strengths, or some other set of adjustable parameters is automatically modified so that some objective is more readily achieved. The backpropagation and bucket brigade algorithms are two types of learning processes.

Learning Rate: A training parameter that controls the size of weight and bias changes during the ANN training process.

Learning Rule: The algorithm used for modifying the connection weights in response to training patterns while training is being carried out.

Local Minimum: The minimum of a function over a limited range of input values. A local minimum may not be the global minimum.

Mean Squared Error (MSE) Function: The performance function that calculates the average squared error between the ANN outputs and the target outputs.

MK System: The standard classification system, the Morgan-Keenan (MK) system was developed in the 1940s and, because photographic film was used, depended on a small region in the blue part of the stellar spectrum for classification. For the last 20 years, astronomers have relied on electronic detectors, which are more sensitive in the red regions of the spectrum.

Momentum: A technique often used to make it less likely for a backpropagation ANN to get caught in a shallow minima.

Momentum Constant: A training parameter that controls how much "momentum" is used.

Multi-Layer Perceptron (MLP): A type of feedforward ANN that is an extension of the perceptron in that it has at least one hidden layer of neurons. Layers are updated by starting at the inputs and ending with the outputs. Each neuron computes a weighted sum of the incoming signals, to yield a net input, and passes this value through its sigmoidal activation function to yield the neuron's activation value. Unlike the perceptron, an MLP can solve linearly inseparable problems.

Mutation: Genetic operation that makes a change of a gene of a chromosome of an organism resulting in the creation of a new individual not found in the parental type.

Natural Selection: Process in nature by which, according to Darwin's Theory of Evolution, only the organism most adapted to their environment tend to survive and transmit their genetic characteristics in increasing numbers to succeeding generations while those less adapted tend to be eliminated.

Neighborhood: A group of neurons within a specified distance of a particular neuron.

Nervous System: The specialized coordinating system of cells, tissues, and organs that endows animals with sensation and volition. In vertebrates, it is often divided into two systems: the central (brain and spinal cord), and the peripheral (somatic and autonomic nervous system).

Neuroglia: Network of branched cells and fibers that support the tissue of the central nervous system. It is also called glia.

Neuron: A simple computational unit that performs a weighted sum on incoming signals, adds a threshold or bias term to this value to yield a net input, and maps this last value through an activation function to compute its own activation. Some neurons, such as those found in feedback or Hopfield networks, will retain a portion of their previous activation.

Neurotransmitters: A group of substances that are released on excitation from the axon terminal of a presynaptic neuron of the central or peripheral nervous system and travel across the synaptic cleft to either excite or inhibit the target cell. Among the many substances that have the properties of a neurotransmitter are acetylcholine, noradrenaline, adrenaline, dopamine, glycine, glutamic acid, enkephalins, endorphins, and serotonin.

Organism: Individual form of life, such as a plant, animal, bacterium, protist, or fungus; a body made up of organs, organelles, or other parts that work together to carry on the various processes of life.

Output Layer: A layer whose output is passed to the world outside the ANN.

Output Vector: The output of an ANN. Each element of the output vector is the output of a neuron.

Outstar Learning Rule: A learning rule that trains a neuron's output weight vector to take on the values of the current output vector of the post-weight layer. Changes in the weights are proportional to the neuron's output.

Overfitting: A case in which the error on the training set is driven to a very small value, but when new data is presented to the ANN, the error is large.

Pathology: Study of the nature of disease and its causes, processes, development, and consequences; anatomic or functional manifestation of a disease.

Pattern Association: The task performed by an ANN trained to respond with the correct output vector for each presented input vector.

Pattern Recognition: The task performed by an ANN trained to respond when an input vector close to a learned vector is presented. The network "recognizes" the input as one of the original target vectors. This can be carried out by the ANN even in the presence of noise or when some data is missing.

Perceptron: An ANN capable of simple pattern recognition and classification tasks. It is composed of two layers where signals only pass forward from nodes in the input layer to nodes in the output layer. There are no connections within a layer. This ANN is often trained with the perceptron learning rule.

Perceptron Learning Rule: A learning rule for training single-layer hard-limit ANN. It is guaranteed to result in a perfectly functioning ANN in finite time, given that the ANN is capable of doing so.

Performance: The behavior of an ANN.

Performance Function: Commonly, the mean squared error (MSE) of the ANN outputs.

Phenotype: The physical or biochemical expression of an individuals' genotype; observable expressed traits, such as eye and skin color.

Phylogenetic Learning: Learning process that takes place over millions of years, as species evolve governed by natural selection. It is here that the abilities to perceive the most important features of the world around us — at our own scales of size and time — and to survive in it, are selected for and perfected.

Physiology: Study of the functions of living organisms and their parts.

Pitch: Human perception of the fundamental frequency of a continuous tone.

Principal Component Analysis (PCA): Orthogonalize the components of ANN input vectors. This procedure also can reduce the dimension of the input vectors by eliminating redundant components.

Radial Basis Network: An ANN that can be designed directly by fitting special response elements where they will do the most good.

Receptor: Cell surface molecule that binds specifically to particular proteins or peptides.

Resilient Backpropagation: A training algorithm that eliminates the harmful effect of having a small slope at the extreme ends of the sigmoid, "squashing" transfer functions.

Rheology/Rheological Behaviour: It is the science of the deformation and flow of matter, and the emphasis on flow means that it is concerned with the relationships between stress, strain, rate of strain, and time.

Rough Set Theory (RST): Can be approached as an extension of the Classical Set Theory, for use when representing incomplete knowledge. Rough sets can be considered sets with fuzzy boundaries sets that cannot be precisely characterized using the available set of attributes

Scaled Conjugate Gradient Algorithm: Avoids the time-consuming line search of the standard conjugate gradient algorithm, for the training process of an ANN.

Self-Organizing: An ANN is called self-organizing if it is capable of changing its connections so as to produce useful responses for input patterns without the instruction of a smart teacher.

Sigmoid Function: A function that is often used as an activation function in an ANN.

Slump Test: The slump test is the most well-known and widely used test method to characterize the workability of fresh concrete. The apparatus consists of a mold in the shape of a frustum of a cone. The mold is filled with concrete. The slump cone mold is lifted vertically upward and the change in height of the concrete is measured.

Soma: The neuron cell body that contains the nucleus.

Spectral Analysis: In astronomy, the study of the composition and structure of bodies via spectroscopy.

Spectrum: A graph or plot of intensity versus wavelength or frequency.

Spread Constant: The distance an input vector must be from a neuron's weight vector to produce an output of 0.5.

Stem Cell: Cell that gives rise to a lineage of cells. Particularly, used to describe the most primitive cells in the bone marrow from which all the various types of blood cells are derived.

Stimulus: A factor that can be detected by a receptor, which in turn produces a response.

Sum-Squared Error: The sum of squared differences between the ANN targets and actual outputs for a given input vector or set of vectors.

Supervised Learning: A learning process in which changes in an ANN's weights and biases are due to the intervention of any external teacher. The teacher typically provides output targets.

Synapse: Space in which a signal passes from one neuron to another.

Target Vector: The desired output vector for a given input vector.

Test Vectors: A set of input vectors (not used directly in training) that is used to test the trained ANN.

Threshold: A quantity added to (or subtracted from) the weighted sum of inputs into a neuron, which forms the neuron's net input. Intuitively, the net input (or bias) is proportional to the amount that the incoming neural activations must exceed in order to activate a neuron.

Timbre: The human perception of harmonics in sound.

Time Series: A series of measurements taken at consecutive points in time. Data mining products which handle time series incorporate time-related operators such as moving average.

Tissue: Organization of differentiated cells of a similar type. There are four basic types of tissue: muscle, nerve, epidermal, and connective.

Topology Functions: Ways to arrange the neurons in a grid, box, hexagonal, or random topology.

Training: A procedure where an ANN is adjusted to do a particular job. Commonly viewed as an "off-line" job, as opposed to an adjustment made during each time interval as is done in adaptive training.

Training Set: An ANN is trained using a training set. A training set comprises information about the problem to be solved as input.

Training Vector: An input and/or target vector used to train an ANN.

Trait: Genetically determined characteristic or condition.

Tuning Phase: Period of self-organizing feature maps (SOFM) training during which weights are expected to spread out relatively evenly over the input space while retaining their topological order found during the ordering phase.

Unsupervised Learning: A learning process in which changes in an ANN's weights and biases are not due to the intervention of any external teacher. Commonly, changes are a function of the current ANN input vectors, output vectors, and previous weights and biases.

Update: Make a change in weights and biases. The update can occur after presentation of a single input vector or after accumulating changes over several input vectors.

Validation Vectors: A set of input vectors (not used directly in training) that is used to monitor training progress so as to keep the ANN from overfitting.

Voiced: Human speech sound that originates as pulses of air passing the vocal cords. Vowels are an example of voiced sounds.

Weight: In an ANN, the strength of a synapse (or connection) between two neurons. Weights may be positive (excitatory) or negative (inhibitory). The thresholds of a neuron also are considered weights, since they undergo adaptation by a learning algorithm.

Weight Function: Weight functions apply weights to an input to get weighted inputs as specified by a particular function.

Weight Matrix: A matrix containing connection weights from a layer's inputs to its neurons. The element w_{ij} of a weight matrix W refers to the connection strength from neuron j to neuron i.

Weighted Input Vector: The result of applying a weight to a layer's input, whether it is an ANN input or the output of another layer.

Workability: ACI defines workability as "that property of freshly mixed concrete which determines the ease and homogeneity with which it can be mixed, placed, consolidated, and finished." Workability is affected by every component of concrete and essentially every condition under which concrete is made. A list of factors includes the properties and the amount of the cement; grading, shape, angularity, and surface texture of fine and coarse aggregates; proportion of aggregates; amount of air entrained; type and amount of pozzolan; types and amounts of chemical admixtures; temperature of the concrete; mixing time and method; and time since water and cement made contact.

About the Authors

Juan R. Rabuñal is an assistant professor in the Department of Information and Communications Technologies, University of A Coruña (Spain). He finished his studies in computer engineering in 1996, and in 2002, he became a doctor of computer science with his thesis "Methodology for the Development of Knowledge Extraction Systems in ANNs." He has worked on several Spanish and European projects and has published many books and papers in several international journals. He is currently working in the areas of evolutionary computation, artificial neural networks, and knowledge extraction systems.

Julián Dorado is associate professor of computer science at the University of A Coruña (Spain). He finished his graduate in computer science in 1994. In 1999, he earned a PhD, with a special mention of European Doctor. In 2004, he finished his graduate degree in biology. He has worked as a teacher at the university for more than eight years. He has published many books and papers in several journals and international congresses. Currently, he is working in the areas of bioinformatics, evolutionary computing, artificial neural networks, computer graphics, and data mining.

* * *

Alfonso Araque has a degree in biological sciences from the Universidad Complutense de Madrid (Spain) (1988). He earned a doctorate in biological sciences from the Universidad Complutense de Madrid (1993), working at the Instituto Cajal, CSIC, Madrid. From 1993 to 1995, he was involved with postdoctoral research at the Instituto Cajal, Madrid. His work has been funded by Fundación Ramón Areces. From 1996 to 1998, he was involved

with postdoctoral research at the Iowa State University, Ames. His work was funded by NATO Scientific Program and Human Frontier Science Program. From 1999 to 2001, he was a research assistant at the Instituto Cajal, CSIC, Madrid. Since 2001, he has been a staff scientist at the Instituto Cajal, CSIC, Madrid. He has published more than 30 works in prestigious international scientific journals and three chapters in scientific books. He has presented 34 communications at international neuroscience meetings. He has dictated 19 invited talks in both national and international universities and scientific centers.

Bernardino Arcay graduated from the electronics program at the Faculty of Physics, University of Santiago, Spain. Since 1986, he has been a member of the Laboratory of Applied Biophysics in Artificial Intelligence in the same faculty. He received a PhD in applied physics from the University of Santiago (1990). He currently is a full professor in the Department of Information and Communications Technologies, University of A Coruña, Spain. His current research areas are intelligent monitoring research, real-time systems, tele-control systems, and signal processing.

Tarun Bhaskar is pursuing his doctoral studies at the Indian Institute of Management, Calcutta (India), in the field of operations research and systems analysis. He obtained his bachelor's degree (Honors) in mechanical engineering from Bihar University, India. His main research interests are soft computing techniques, network security, and combinatorial optimization problems.

Anthony Brabazon currently lectures at University College Dublin, Ireland. His research interests include mathematical decision models, evolutionary computation, and the application of biologically-inspired algorithms to the domains of business and finance. He has published widely in these fields, and has been a member of the programme committee at multiple international conferences, and acts as reviewer for several journals. Prior to joining UCD he worked in the banking sector, and for KPMG.

Giuseppe Buzzanca received a diploma (BA) in piano performance at the State Conservatory of Music in Bari (Italy), Laurea (master's) in musicology at the University of Bologna. He holds a PhD from the University of Bologna with a dissertation on music and AI. He has been awarded a Fulbright Senior Research Fellowship at Harvard University. Founder of the Italian Society for Music and Artificial Intelligence (ISMAI), he is a member of the scientific committee of the International Conference *Understanding Music* (University of Napoli2). His publications include articles in music and AI (his main fields of interest). He is full professor in the State Conservatory of Music in Bari and chair of the post-graduate course in music and technologies.

Alberto Cancela Carollo is a PhD student in the Department of Information and Communications Technologies, University of A Coruña (Spain). He finished his studies in computer engineering in 2003. Since then, he has participated in several research

projects and has published several papers. His actual research interests are focused in evolutionary computation (multimodal and multiobjective optimization) and image segmentation and its use in medical diagnosis.

Iciar Carricajo graduated with a degree in physics from the University of La Laguna, Spain (2000), and earned her first postgraduate degree for research in classification of stellar spectra in 2004. She is developing her PhD on automatic technique for the classification of stellar spectra in the Department of Navigation and Earth Science, University of A Coruña (Spain). Her main research interests include classification of stellar spectra, stellar evolution, and techniques of artificial intelligence .

Diego Carro, civil engineer (University of A Coruña, Spain, April 2004), specialist in structures and railways. After obtaining the degree, he has worked on town-planning on a consulting company (until October 2004). He started working on research on beach dynamics and sediment transport while he was obtaining his degree. At present, he is earning his doctorate in civil engineering and developing his thesis directed by Fernando Martínez Abella. He synchronizes his studies with the research on materials science and on the construction uses of the oily waste from the Prestige tank. He has collaborated on several R&D projects with the construction engineering team.

Paulo Cortez was born in Braga, Portugal. He studied at the University of Minho, where he obtained an engineering degree (1995), an MSc (1998), and a PhD (2002) in computer science. Currently, he is a lecturer at the Department of Information Systems and a researcher at the ALGORITMI center. He is co-author of more than 30 publications in international conferences and journals (e.g., IEEE or Springer). Since 2002, he has been a regular reviewer of the *Neural Processing Letters* journal. His research interests include: data mining and machine learning; neural and evolutionary computation; and forecasting. For more information, visit www.dsi.uminho.pt/~pcortez/.

José Manuel Cotos graduated from the electronics program at the Faculty of Physics, University of Santiago (Spain). Since 1990, he has been a member of the Laboratory of Systems in Remote Sensing and Geographic Information Systems Group, at the same faculty. He received a PhD in applied physics from the University of Santiago (1994). He is currently a professor in the Department of Computer Science, University of Santiago, Spain. His current research area is geospatial databases and applications.

Carlos Dafonte graduated with a degree in computer science from the University of A Coruña, Spain (1994) and received his first postgraduate degree for research in computer science in 1995. He received a PhD in computer science from the University of A Coruña (2000). He is currently an assistant professor in the Department of Information and Communications Technologies, University of A Coruña. His research interest includes information systems, intelligent systems for real-time control, and signal processing.

José Manuel Andrade Garda graduated with a degree in chemistry from the University of Santiago of Compostela in 1990 and earned his PhD in 1995 from the University of A Coruña (Spain). His thesis focused on the application of multivariate chemometric methods for quality management systems in laboratories (sponsored by the Spanish Government). Since 1997, he has held a position at the Department of Analytical Chemistry, University of A Coruña, where he teaches about the basics of the instrumental measurements, quality and environmental management systems, and chemometric applications. His research covers infrared and atomic spectrometry, coupled to chemometric techniques (multivariate regression, selection of variables, artificial neural networks) and environmental studies (pattern recognition). At present he is involved in some projects to monitor the environmental consequences of the Prestige's wreck (2002) along the Galician coastline.

María Paz Gómez-Carracedo graduated with a degree in chemical sciences from the University of Santiago de Compostela (1998). In 2005, she obtained her PhD by a work where multivariate methods were applied to two different fields of consumers' concern: authentication of fruit (apple) juice and quality control of aviation jet fuel. She has participated in several R&D projects with governmental support. At present, she works at the University of A Coruña, Department of Analytical Chemistry, to investigate the consequences of Prestige's oil spill.

Belén González earned a doctorate in civil engineering with the qualification of Outstanding Cum Laude by the University of A Coruña (Spain). He was an assistant lecturer at the School of Civil Engineering, Department of Construction Technology, University of A Coruña. Since February 2005, Fonteboa has been academic secretary at the School of Civil Engineering, University of A Coruña. Since July of 1999, Fonteboa has been an investigator with the Department of Construction Technology, University of A Coruña. Since November 2002, Fonteboa has been a member of the working group "Recycled Concrete"; the purpose of this group is to develop a Spanish code that contains specific recommendations on the utilization of the recycled concrete aggregates in structural concrete.

Alfonso Iglesias graduated from the electronics program at the Faculty of Physics, University of Santiago, Spain. Since 1996, he has been a member of the Laboratory of Systems in Remote Sensing and Geographic Information Systems Group, at the same faculty. He received a PhD in artificial intelligence and advanced computation from the University of Santiago (2003). He currently is a researcher in the Department of Information and Communications Technologies, University of A Coruña, Spain. His current research area is artificial intelligence and remote sensing data.

Narasimha Kamath B. is pursuing his doctoral studies, funded by Infosys Technologies, at the Indian Institute of Management, Calcutta (India). He obtained his bachelor's degree (Honors) in mechanical from Regional Engineering College, Surathkal, India. He was the lead system designer at GE Medical Systems, South Asia, and was involved in

research work for Hindustan Lever Limited and General Electric. Supply chain management, intrusion detection and decision making are his current research interests.

Kun-Chang Lee (leekc@skku.ac.kr) is a full professor of MIS at Sungkyunkwan University in Seoul, Korea. He received his PhD in MIS from Korea Advanced Institute of Science and Technology (KAIST), a Master's of Science in MIS from KAIST, and a Bachelor's of Arts in business administration from Sungkyunkwan University, Seoul, Korea. His research focuses on decision analysis involved in electronic commerce management. Recently, he has been developing several working papers specializing in knowledge management, artificial intelligence-based analysis of IS performance, and schema-based decisions. His research findings have been published in *Journal of Management Information Systems*, *Decision Support Systems, International Journal of Production Research, Expert Systems with Applications, Fuzzy Sets and Systems, Intelligent Systems in Accounting Finance and Management, Computers & OR, Computers & IE, Simulation, Expert Systems*, among others.

Daniel Manrique is a computer engineer and has a PhD in artificial intelligence. He is a professor at the Madrid Technical University's School of Computing. His teaching activities focus on undergraduate and postgraduate computing engineering education in the areas of hard and soft artificial intelligent. Professor Manrique co-directs the artificial neural networks work group at the university and is member of the International Program Committee of the 23rd, 24th, and 25th International Conference on Innovative Techniques and Applications of Artificial Intelligence. He has numerous publications in international journals relating to soft computing, unconventional computation, and e-learning.

Minia Manteiga graduated with a degree in physics from the Faculty of La Laguna (1986). Since then, she entered the postgraduate program at the Spanish Instituto de Astrofísica de Canarias (IAC, Tenerife, Spain) and received a PhD in astrophysics (1990). She spent two years as a postdoctoral researcher at the Italian Istituto di Astrofísica Spaziale (IAS, at Frascati-Rome) and three years at the INTA (Spanish National Institute for Aerospacial Studies, Madrid, Spain). In 1997, she got a position as an associate professor at the Applied Physics Department, University of Vigo. She is currently an associate professor in the Navigation and Earth Sciences Department, University of A Coruña (Spain). She is a member of the IAU (International Astrophysics Union) and president of the GEA (Group of Astrophysics) of the Spanish Royal Society of Physics. Her current research area is stellar evolution and the study of stellar populations.

Eduardo D. Martín completed his undergraduate studies in medicine at the Universidad Nacional de Tucumán, Argentina (1991). He earned his graduate degree in medicine from the Universidad Nacional de Tucumán with Dr. Emilio E. Decima, PhD (1996). He trained as a postdoctoral fellow at Cajal Institute-CSIC, with Dr. Washington Buño (1996-1997 and 1999-2001), and at the Brain Mapping Unit, Instituto Pluridisciplinar, Universidad

Complutense de Madrid, with Dr. Miguel A. Pozo (2001-2003). Since 2003, he has held a research assistant position at the Unidad Asociada Neurodeath, UCLM-CSIC, University of Castilla-La Mancha (Spain), in the laboratory of Dr. Valentín Ceña.

Isabel Martínez Mª earned an MSc from the Polytechnical University of Madrid. From January 1992 to November 1993, Lage was an engineer in charge of the Department of Structures, LABORNOSA. In 1994, Lage was an assistant professor at the Scholl of Civil Engineering, University of A Coruña. From June 1999 to April 2004, Lage was an assistant director at the School of Civil Engineering, University of A Coruña (Spain). Since July 2004, Lage has been a member of the working group "Recycled Concrete"; the purpose of this group is to develop a Spanish code that contains specific recommendations on the utilization of the recycled concrete aggregates in structural concrete.

Mónica Miguélez was born in 1978 in Pontevedra, Spain. She has a research fellowship in the Department of Information and Communications Technologies, University of A Coruña (Spain). She finished her graduate degree in computer science in 2001 and is currently working on medical image and digital image processing for decision support.

José Neves is a full professor in the Computer Science Department, University of Minho (Portugal). He is the head of the Artificial Intelligence Group and coordinates several projects with applications in the areas of law and medicine. His research interests are knowledge representation and computational logic. For more information, visit www.di.uminho.pt/~jneves/.

Tae-Young Paik (typaik@skku.edu) is a full professor of management accounting at Sungkyunkwan University, Seoul, Korea. He received a PhD in accounting from the University of California at Berkeley. He has published papers in leading journals such as *Management Science, Journal of Accounting Research, Review of Quantitative Finance and Accounting,* and *Journal of Accounting, Auditing, and Finance*. His main research interests lie in applying management accounting theories to telecommunications industry issues.

Alejandro Pazos is a professor in the Computer Science Faculty, University of A Coruña (Spain). Born in Padrón, Spain in 1959, he studied medicine at Santiago de Compostela University in 1987. He received a master's degree in knowledge engineering (1989), a PhD in computer science at the Universidad Politécnica de Madrid (1990), and a PhD in medicine from the Universidad Complutense de Madrid (1996). He has worked with research groups at the Georgia Institute of Technology, Harvard Medical School, Stanford University, Computer Science Faculty of the Universidad Politécnica de Madrid, among others. He funded and is director of the research laboratory Artificial Neural Networks and Adaptive Systems at the Computer Science Faculty (A Coruña University).

Nieves Pedreira is an assistant professor in the Department of Information and Communications Technologies, University of A Coruña (Spain). She earned a degree in computer science from the University of A Coruña (1993). This was followed by a master's degree in communications and real time systems. After having worked in private enterprises, she returned to the university in 1997 as a PhD student, and in 2003, she became a doctor of computer science with her thesis "A Model for Virtual Learning to Learn." She is also a tutor at the UNED (Spanish Distance Education National University) since 1998. Her research interests are focused on distance learning and new technologies.

Javier Pereira is a professor in the faculty of health sciences at the University of A Coruña (Spain). He finished his PhD in 1995 and is a member of the research group RNASA-GIB in the Department of Information and Communications Technologies. He has worked in many different fields, including neural network development, evolutionary computation, image acquisition, and processing on medical environments. He is currently working on several projects about computer science applied in medical environments.

Robert Perkins received a bachelor's of commerce degree from University College Dublin (Ireland) and a master's of accounting degree from the Michael Smurfit Graduate School of Business. In 2003, he received the Deloitte and Touche Prize for first place in financial accounting. He is currently working in the Investment Management Group of PricewaterhouseCoopers.

Ana B. Porto is an assistant professor in the Computer Science Faculty, University of A Coruña (Spain). Born in A Coruña 1975, she earned a degree in computer science at A Coruña University (1998), received a PhD in computer science at A Coruña University (2004), and is currently studying for a PhD in neuroscience. She works with research groups in Ramon y Cajal Institute of Neurobiology (CSIC), Madrid, Spain. She is a member of the research laboratory Artificial Neural Networks and Adaptive Systems at the Computer Science Faculty (A Coruña University).

Marcos Gestal Pose is a PhD student in the Department of Information and Communications Technologies, University of A Coruña (Spain). He finished his studies in computer engineering in 2001. Since then, he has participated in several research projects and he has published papers in international journals and books. His actual research interests are focused on evolutionary computation (specifically genetic algorithms) and artificial neural networks and their interaction to perform variable selection.

Jerónimo Puertas is a professor in the area of hydraulics and hydrology at the University of A Coruña (Spain). He has previously worked at the Polytechnical University of Catalonia, Barcelona. He finished his studies in civil engineering in 1989 and became a doctor of civil engineering in 1994. He has published many papers in the main journals related with hydraulics and has been advisor of nine PhD theses. He actually works in sewer systems design, dam engineering, and water supply systems.

Juan Ríos has a PhD in civil engineering and is a full professor at the Madrid Technical University's School of Computing. He was a program advisory committee member of the IV and V International Symposiums on Knowledge Engineering, a member of the Expert Commission, and founding member (1988) of the Centre of Computing and Communications Technology Transfer (CETTICO). He is also a member of the International Association of Knowledge Engineering. Professor Ríos is director of the Artificial Neural Networks Work Group at the university and has much experience in research and development engineering projects in the fields of artificial intelligence and neural networks.

Daniel Rivero was born in 1978 in the city of A Coruña, Spain. In 1996, he began his studies in computer engineering and finished them in 2001. After that, he began his PhD in computer science and artificial intelligence. He has published several papers on international conferences and journals on evolutionary computation, signal processing, image processing, artificial neural networks, and so on. He is working in the RNASA (Artificial Neural Networks and Adaptive Systems) Lab in the University of A Coruña (Spain).

Miguel Rocha was born in Lisbon, Portugal in 1973. He studied at the University of Minho (Portugal), obtaining a systems engineering degree (1995), an MSc (1998), and a PhD (2004) in computer science. Since 1998 he has been a lecturer in the same institution with the Computer Science Department. His research interests include bioinformatics, data mining, machine learning, and optimization. In the last several years, he has authored a number of scientific publications in the fields of neural networks and evolutionary computation.

Alejandra Rodríguez graduated with a degree in computer science from the University of A Coruña, Spain (2000), and received her first postgraduate degree for research in computer science in 2002. She is developing her PhD on applications of artificial intelligence for astrophysics in the Department of Information and Communications Technologies, University of A Coruña. Her main research interests include information systems, techniques of artificial intelligence, and signal processing.

Alfonso Rodríguez-Patón is a physicist (electronic and computation specialty) and has a PhD in computer science. He is professor at the at the Madrid Technical University's School of Computing. He is member of the Artificial Intelligence Lab at the same University. His main research interest lies in the interplay of computer science, biology, and technology. His topics of research are formal languages and automata theory, DNA computing, cellular computing, and any bio-inspired or unconventional model of computation.

J. Sethuraman obtained his BE in electronics and communication engineering in 2001 with distinction from the University of Madras. He is currently a doctoral student at the

Indian Institute of Management, Calcutta, India. In addition to being a fellow with the MIS group of this institute, he has been a EURO-IFORS fellow in 2004. He also specializes in finance & control. His current areas of interest include telecommunications design & planning, business cases involving ICT deployments, artificial intelligence, and data mining and its applications in finance. He is also interested in applying systems concepts to the development sector, especially e-governance. He is one of the founding members of the EURO working group on "Young People for OR in Developing Countries" (YORC). He has represented India at the South Asian Youth Invitation Program 2003 in Japan and the prestigious Copenhagen Consensus 2004.

Miguel Varela is a PhD student at the University of A Coruña (Spain). He finished his degree in 2001 and is working on evolutionary computation and artificial neural networks. He is a member of the research group RNASA-GIB in the Department of Information and Communications Technologies. He has published several papers on genetic programming and distributed systems in different journals and conferences and has worked on several applications of evolutionary computation on civil engineering. He is currently working in evolutionary computation on distributed systems.

Index

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W